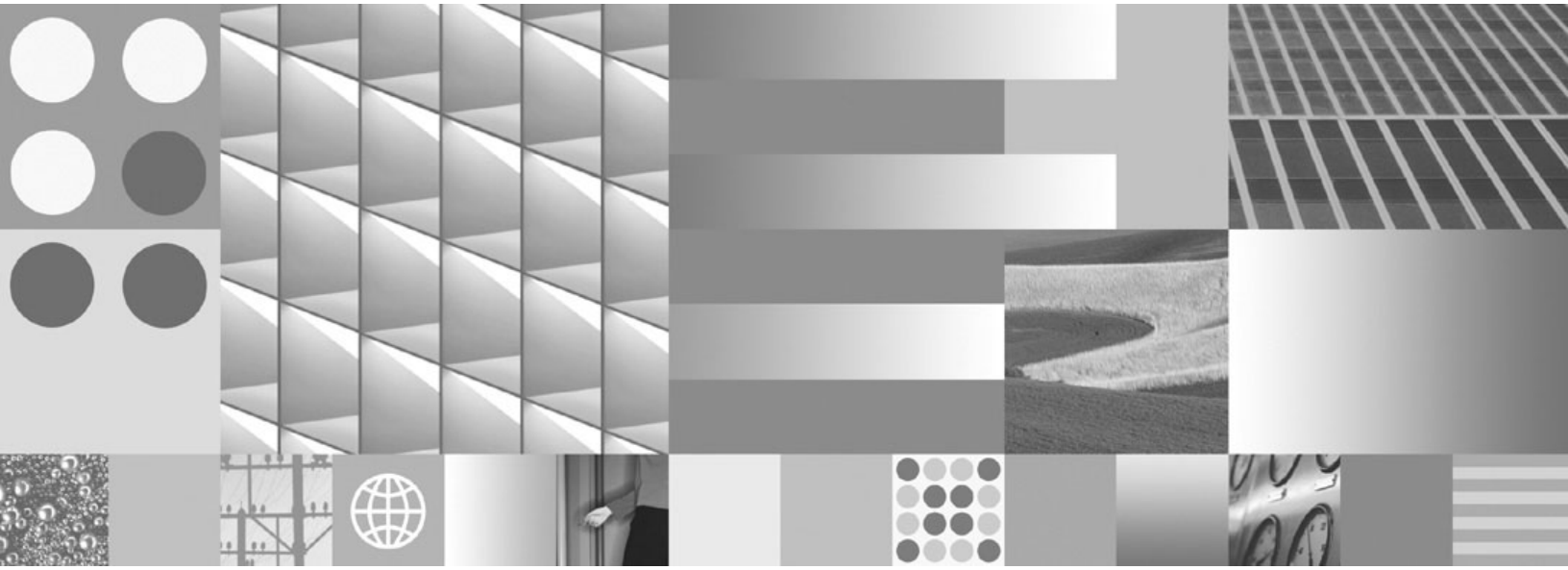
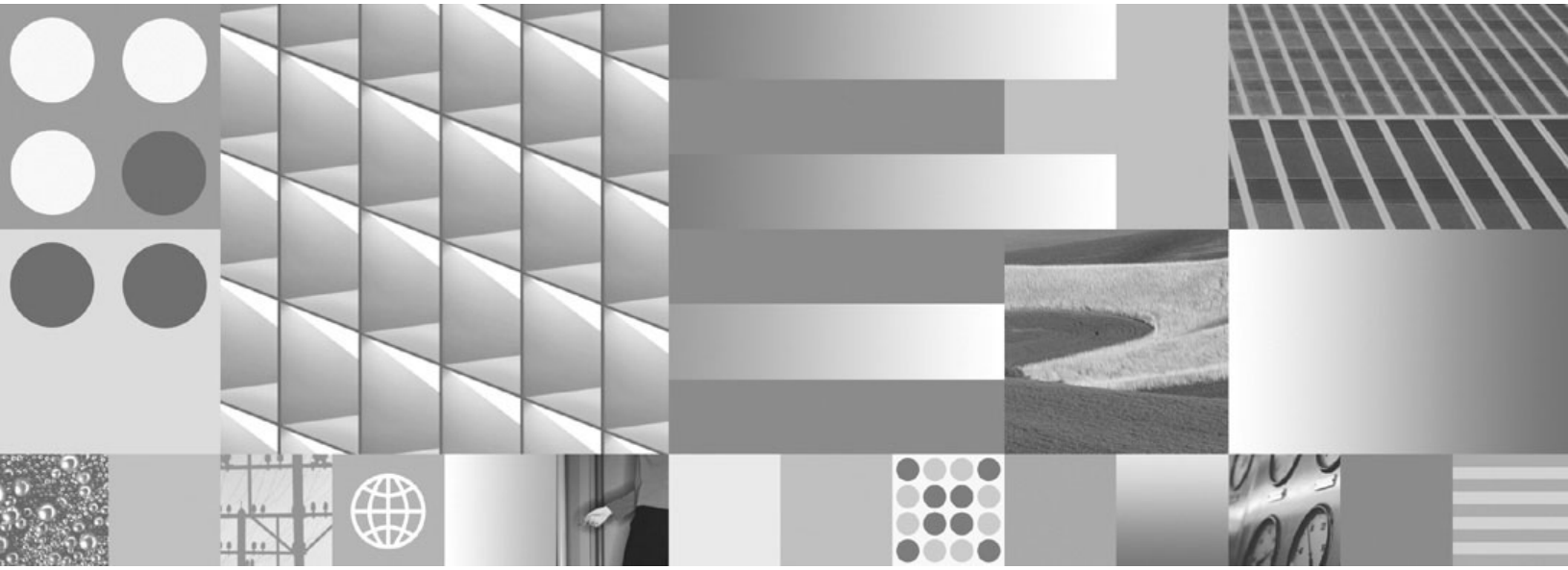
DB2 Version 9.5
für Linux, UNIX und Windows



Datenserver, Datenbanken und Datenbankobjekte

SC12-3912-02

DB2 Version 9.5
für Linux, UNIX und Windows



Datenserver, Datenbanken und Datenbankobjekte

SC12-3912-02

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in Anhang B, „Bemerkungen", auf Seite 703 gelesen werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
DB2 VErsion 9.5 for Linux, UNIX, and Windows, Data Servers, Databases, and Database Objects Guide,
IBM Form SC23-5849-02,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
April 2009

Inhaltsverzeichnis

Zu diesem Handbuch

Das Handbuch 'Datenserver, Datenbanken und Datenbankobjekte' enthält Informationen, die zur Verwendung und Verwaltung der Produkte des DB2- Verwaltungssystems für relationale Datenbanken (RDBMS) erforderlich sind. Es enthält Informationen zur Planung und zum Entwurf von Datenbanken sowie zur Implementierung und Verwaltung von Datenbankobjekten. Darüber hinaus enthält dieses Handbuch Referenzinformationen zur Konfiguration und Optimierung von Datenbanken.

Zielgruppe

Dieses Handbuch ist in erster Linie für Datenbank- und Systemadministratoren vorgesehen, die eine Datenbank für den Zugriff durch lokale und ferne Clients entwerfen, implementieren und verwalten müssen. Es bietet sich auch für Programmierer und andere Benutzer an, die sich Kenntnisse über die Verwaltung und den Betrieb des DB2-Verwaltungssystems für relationale Datenbanken aneignen müssen.

Aufbau des Handbuchs

Dieses Handbuch ist in die folgenden vier Teile untergliedert:

Teil 1. Datenserver
Dieser Teil enthält eine kurze Beschreibung von DB2-Datenservern mit Informationen zur Verwaltung ihrer Kapazität sowie zur Unterstützung großer Seiten in 64-Bit-Umgebungen unter AIX. Darüber hinaus enthält er Informationen zur Ausführung mehrerer DB2-Kopien auf einem einzigen Computer, Informationen zu den automatischen Funktionen, die Sie bei der Verwaltung des Datenbanksystems unterstützten, Informationen zum Entwerfen, Erstellen und Verwenden von Instanzen sowie optionalen Informationen zur Konfiguration von LDAP-Servern (LDAP - Lightweight Directory Access Protocol).

Teil 2. Datenbanken
Dieser Teil beschreibt das Entwerfen, Erstellen und Verwalten von Datenbanken, Pufferpools, Tabellenbereichen und Schemata. Detaillierte Informationen zu Datenbankpartitionen finden Sie in dem neuen Handbuch *Partitionierung und Clustering*.

Teil 3. Datenbankobjekte
Dieser Teil beschreibt das Entwerfen, Erstellen und Verwalten der folgenden Datenbankobjekte: Tabellen, Integritätsbedingungen, Indizes, Trigger, Sequenzen und Sichten.

Teil 4. Referenz
Dieser Teil enthält Referenzinformationen zur Konfiguration und Optimierung des Datenbanksystems durch Umgebungsvariablen, Registrierdatenbankvariablen und Konfigurationsparameter. Darüber hinaus enthält er Informationen zu den verschiedenen Namenskonventionen sowie zu SQL- und XML-Begrenzungen.

Teil 1. Datenserver

Kapitel 1. DB2-Datenserver

Datenserver stellen Software-Services für ein sicheres und effizientes Management für strukturierte Daten bereit. DB2 ist Hybridtyp aus relationalem Datenserver und XML-Datenserver.

Die Bezeichnung Datenserver bezieht sich auf ein System, auf dem die DB2-Datenbanksteuerkomponente installiert ist. Die DB2-Steuerkomponente ist ein umfassend ausgestattetes, leistungsfähiges Datenbankverwaltungssystem, das eine für die tatsächliche Datenbanknutzung optimierte SQL-Unterstützung sowie Tools zur Verwaltung der Daten enthält.

IBM bietet eine Reihe von Datenserverprodukten an, zu denen auch Datenserver-Clients gehören, die auf sämtliche verschiedene Datenserver zugreifen können. Eine vollständige Liste der DB2-Datenserverprodukte und der verfügbaren Zusatzeinrichtungen sowie detaillierte Beschreibungen und Spezifikationen finden Sie unter http://www-306.ibm.com/software/data/db2/9/.

Verwaltung der Datenserverkapazität

Wenn die Kapazität des Datenservers nicht Ihren gegenwärtigen oder zukünftigen Anforderungen genügt, können Sie die Kapazität des Datenservers erweitern, indem Sie Plattenspeicherplatz hinzufügen und weitere Container erstellen oder Speicher hinzufügen. Wenn diese einfachen Strategien nicht die erforderliche Kapazität erbringen, können Sie auch das Hinzufügen von Prozessoren oder physischen Partitionen in Betracht ziehen. Wenn Sie Ihr System durch Änderung der Umgebung skalieren, sollten Sie sich über die Auswirkungen dieser Änderung auf die Prozeduren in Ihrer Datenbank wie zum Beispiel auf das Laden von Daten oder das Backup oder den Restore von Datenbanken im Klaren sein.

Hinzufügen von Prozessoren

Wenn eine Konfiguration mit Einzelpartitionsdatenbanken und einem Einzelprozessor bis zur maximalen Kapazitätsgrenze ausgelastet wird, können Sie entweder Prozessoren oder Datenbankpartitionen hinzufügen. Der Vorteil hinzugefügter Prozessoren liegt in der größeren Verarbeitungskapazität. In einem SMP-System benutzen Prozessoren Hauptspeicher und Speichersystemressourcen gemeinsam. Alle Prozessoren befinden sich in einem System, sodass keine Aufwände zu beachten sind, wie zum Beispiel die Kommunikation oder die Koordination von Tasks zwischen Systemen. Die Dienstprogramme beispielsweise zum Laden, Backup und Restore können die zusätzlichen Prozessoren vorteilhaft nutzen.

Anmerkung: Einige Betriebssysteme, wie zum Beispiel das Solaris-Betriebssystem, können Prozessoren dynamisch in den Online- und den Offlinemodus versetzen.

Wenn Sie Prozessoren hinzufügen, prüfen und modifizieren Sie einige Datenbankkonfigurationsparameter, die die Anzahl der verwendeten Prozessoren bestimmen. Die folgenden Datenbankkonfigurationsparameter bestimmen die Anzahl der verwendeten Prozessoren und müssen möglicherweise aktualisiert werden:

- Grad der Parallelität (dft_degree)
- Maximaler Grad der Parallelität bei Abfragen (max_querydegree)

- Partitionsinterne Parallelität aktivieren (intra_parallel)

Darüber hinaus sollten Sie auch Parameter prüfen, die bestimmen, wie Anwendungen die parallele Verarbeitung ausführen.

In einer Umgebung, in der TCP/IP zur Kommunikation verwendet wird, prüfen Sie den Wert der Registrierdatenbankvariablen DB2TCPCONNMGRS.

Hinzufügen physischer Partitionen

Wenn sich Ihr Datenbankmanager momentan in einer Umgebung mit partitionierten Datenbanken befindet, können Sie sowohl die Datenspeicherkapazitäten als auch die Verarbeitungskapazitäten erhöhen, indem Sie separate physische Einzelprozessor- oder Multiprozessorpartitionen hinzufügen. Die Hauptspeicher- und Speichersystemressourcen jeder Datenbankpartition werden nicht mit den anderen Datenbankpartitionen gemeinsam genutzt. Zwar kann das Hinzufügen von Datenbankpartitionen mit Problemen hinsichtlich der Kommunikation und der Taskkoordination verbunden sein, aber diese Wahl hat den Vorteil, dass Daten und Benutzerzugriffe auf mehr als ein System verteilt werden können. Der Datenbankmanager unterstützt diese Umgebung.

Sie können Datenbankpartitionen hinzufügen, während das Datenbankmanagersystem aktiv bzw. während es gestoppt ist. Wenn Sie Datenbankpartitionen hinzufügen, während das System aktiv ist, müssen Sie das System jedoch stoppen und neu starten, bevor Datenbanken auf die neue Datenbankpartition migriert werden.

Wenn Sie eine neue Datenbankpartition hinzufügen, können Sie eine Datenbank, die die neue Datenbankpartition nutzt, erst dann löschen oder erstellen, wenn die Prozedur abgeschlossen und der neue Server erfolgreich in das System integriert ist.

Aktivieren der Unterstützung großer Seiten in einer 64-Bit-Umgebung (AIX)

Zusätzlich zur traditionellen Seitengröße von 4 KB unterstützen die POWER4-Prozessoren (und höhere Versionen) in den IBM eServer pSeries-Systemen auch eine Seitengröße von 16 MB. Anwendungen, wie zum Beispiel IBM DB2 Version 9.1 für AIX (64-Bit-Edition), die intensiven Zugriff auf den Hauptspeicher erfordern und große Mengen an virtuellem Speicher nutzen, können durch die Verwendung großer Seiten eine Leistungsverbesserung erzielen.

Anmerkung: Eine Aktivierung der Unterstützung großer Seiten verhindert, dass der Speichermanager für die automatische Leistungsoptimierung die Datenbankspeicherbelegung automatisch optimiert. Daher sollte sie nur für klar definierte Auslastungen, die durch einen relativ statischen Datenbankspeicherbedarf gekennzeichnet sind, in Betracht gezogen werden.

1. Detaillierte Informationen zur Ausführung des Befehls vmo finden Sie in Ihren AIX-Handbüchern.
2. Sie müssen äußerst vorsichtig vorgehen, wenn Sie Ihr System für das Fixieren (Pinning) von Speicher und die Unterstützung großer Seiten konfigurieren. Wenn Sie zu viel Speicher fixieren, führt das zu aufwendigen Seitenwechselvorgängen für Hauptspeicherseiten, die nicht im Speicher belassen werden. Wenn Sie zu viel physischen Speicher für große Seiten zuordnen, wird die Systemleistung beeinträchtigt, wenn nicht ausreichend Speicher für die Unterstützung der 4-KB-Seiten vorhanden ist.

3. Die Einstellung der Registrierdatenbankvariablen **DB2_LARGE_PAGE_MEM** impliziert ebenfalls, dass der Speicher fixiert wird.

Sie müssen über die Rootberechtigung verfügen, um mit AIX-Betriebssystembefehlen zu arbeiten.

1. Konfigurieren Sie Ihren AIX-Server zur Unterstützung großer Seiten, indem Sie den Befehl vmo mit den folgenden Markierungen absetzen:

   ```
   vmo -r -o lgpg_size=GroßeSeitengröße -o lgpg_regions=GroßeSeiten
   ```

   Dabei gibt *GroßeSeitengröße* die Größe der hardwareunterstützten großen Seiten in Byte und *GroßeSeiten* die Anzahl der zu reservierenden großen Seiten an. Wenn Sie zum Beispiel 25 GB für die Unterstützung großer Seiten zuordnen müssen, führen Sie den Befehl folgendermaßen aus:

   ```
   vmo -r -o lgpg_size=16777216 -o lgpg_regions=1600
   ```

2. Führen Sie den Befehl bosboot aus, sodass der zuvor ausgeführte Befehl vmo nach dem nächsten Systemstart wirksam wird.
3. Nachdem der Server gestartet wurde, aktivieren Sie ihn für fixierten ('pinned') Speicher:
   - Führen Sie den Befehl vmo mit den folgenden Markierungen aus:

     ```
     vmo -o v_pinshm=1
     ```

   - Setzen Sie mithilfe des Befehls db2set die Registrierdatenbankvariable **DB2_LARGE_PAGE_MEM** auf den Wert DB, und starten Sie anschließend DB2:

     ```
     db2set DB2_LARGE_PAGE_MEM=DB
     db2start
     ```

Kapitel 2. Mehrere DB2-Kopien

Mit Version 9 oder einer späteren Version können Sie mehrere DB2-Kopien auf demselben Computer installieren und ausführen. Eine DB2-Kopie bezieht sich auf eine oder auch mehrere Installationen von DB2-Datenbankprodukten an einer bestimmten Position auf demselben Computer. Die Kopien von DB2 Version 9 können dieselbe oder auch verschiedene Codeversionen besitzen.

Dies bietet folgende Vorteile:

- Die Möglichkeit, Anwendungen auszuführen, die verschiedene DB2-Versionen auf demselben Computer zur gleichen Zeit erfordern.
- Die Möglichkeit, unabhängige Kopien von DB2-Produkten für unterschiedliche Funktionen auszuführen.
- Die Möglichkeit, auf demselben Computer Tests auszuführen, bevor eine Produktionsdatenbank auf eine neuere Version des DB2-Produkts migriert wird.
- Die Möglichkeit für unabhängige Softwareanbieter (ISVs), ein DB2-Serverprodukt in ihr Produkt einzubetten und die DB2-Datenbank den Benutzern gegenüber zu verdecken. Verwenden und verteilen Sie für COM+-Anwendungen den IBM Data Server Driver für ODBC und CLI mit Ihrer Anwendung und nicht den Data Server Runtime Client, da für COM+-Anwendungen nur ein Data Server Runtime Client zur gleichen Zeit verwendet werden kann. Der IBM Data Server Driver für ODBC und CLI unterliegt dieser Einschränkung nicht.

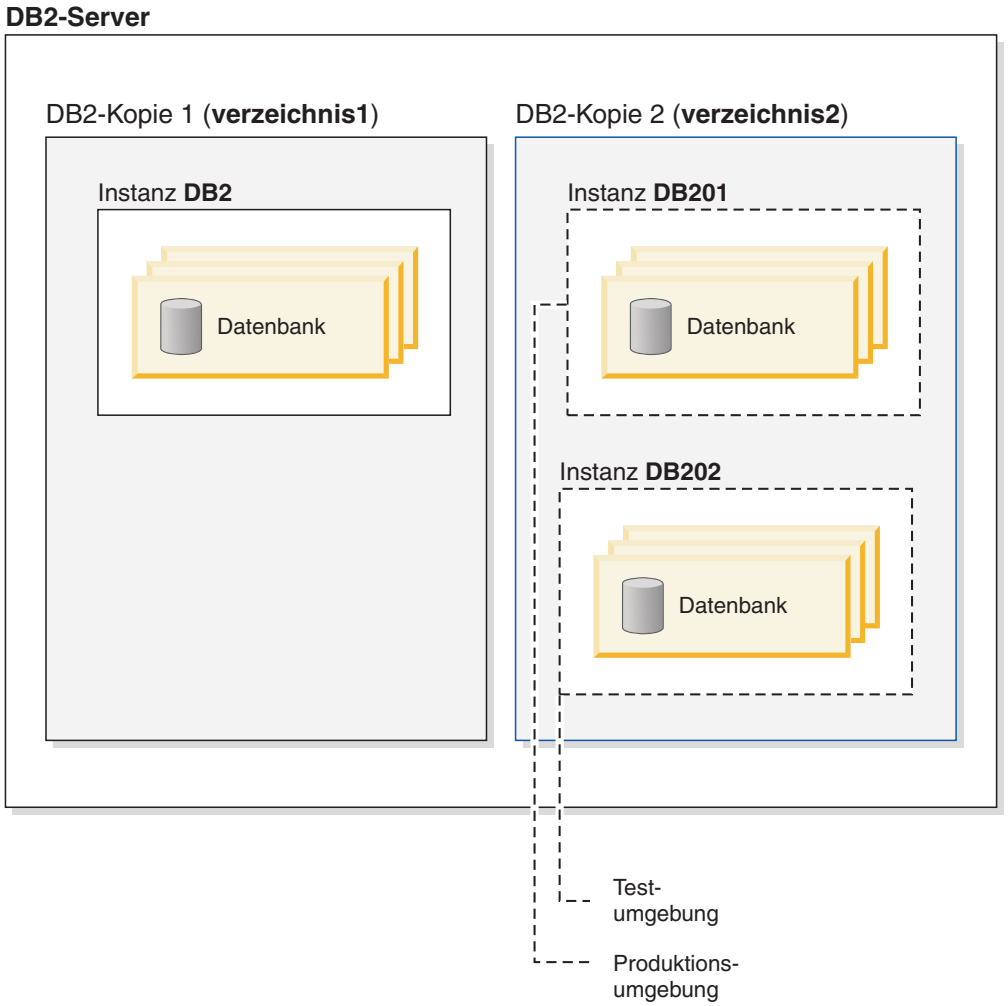
Standardkopie der IBM-Datenbankclientschnittstelle

Sie können über mehrere DB2-Kopien auf einem einzigen Computer sowie über eine Standardkopie der IBM Datenbankclientschnittstelle verfügen. Über diese Schnittstelle erhält eine Clientanwendung den ODBC-Treibercode, den CLI-Treibercode und den .NET-Datenprovidercode, der standardmäßig für eine Kommunikation mit der Datenbank erforderlich ist.

Seit Version 9.1 ist der Code für die Kopie der IBM Datenbankclientschnittstelle in der DB2-Kopie enthalten. Ab Version 9.5 steht ein neues Produkt zur Verfügung, das Sie installieren können und das den erforderlichen Code für die Kommunikation einer Clientanwendung mit einer Datenbank bereitstellt. Dieses Produkt ist ein IBM Data Server Driver Package (DSDRIVER). Mit Version 9.5 (und späteren Versionen) können Sie DSDRIVER auf einer IBM Data Server-Treiberkopie installieren, die von der Installation einer DB2-Kopie getrennt ist.

Ab Version 9.1 können mehrere DB2-Kopien auf Ihrem Computer installiert sein, ab Version 9.5 können mehrere IBM Datenbankclientschnittstellenkopien und mehrere DB2-Kopien auf Ihrem Computer installiert sein. Bei der Installation einer neuen DB2-Kopie oder einer neuen IBM Daten Server-Treiberkopie hätten Sie die Möglichkeit, die DB2-Standardkopie und die Standardkopie der IBM Datenbankclientschnittstelle zu ändern.

Im folgenden Diagramm sind mehrere DB2-Kopien auf einem DB2-Server installiert. Dabei kann es sich um eine beliebige Kombination der DB2-Datenbankprodukte handeln:



Kopien der Version 8 und Version 9 (oder höher) können auf ein und demselben Computer koexistieren. Allerdings muss Version 8 als DB2-Standardkopie und als Standardkopie der IBM Datenbankclientschnittstelle verwendet werden. Es ist nicht möglich, während der Installation von der Kopie der Version 8 zur Kopie der Version 9 (oder höher) als DB2-Standardkopie oder als Standardkopie der IBM Datenbankclientschnittstelle zu wechseln. Es ist außerdem nicht möglich, später den Befehl zum Wechseln der Standardkopie db2swtch auszuführen, es sei denn, Sie führen zuerst eine Migration durch oder deinstallieren die Kopie der Version 8. Wenn Sie den Befehl db2swtch ausführen, solange Version 8 noch auf dem System installiert ist, erhalten Sie eine Fehlernachricht, die Sie darüber informiert, dass die Standardkopie nicht geändert werden kann, weil auf dem System Version 8 gefunden wurde.

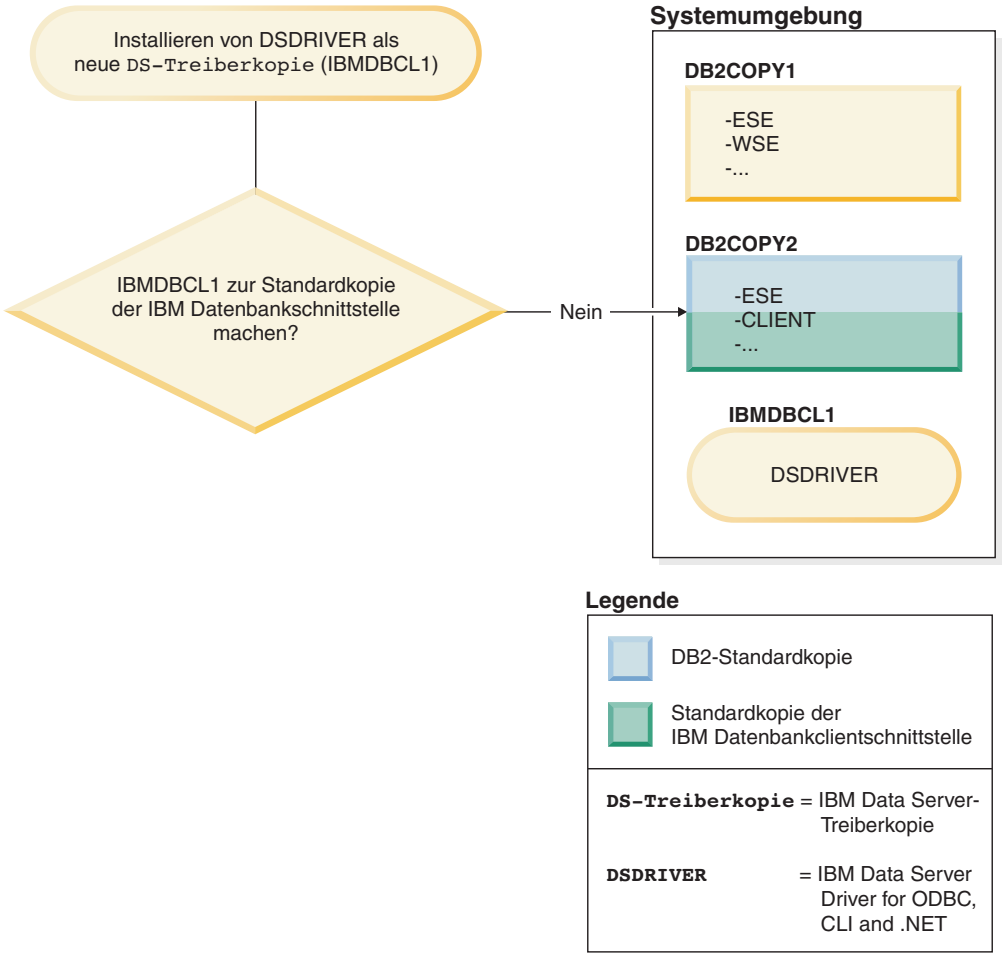
Nach der Installation mehrerer DB2-Kopien oder mehrerer IBM Datenservertreiberkopien können Sie die DB2-Standardkopie bzw. die Standardkopie der IBM® Datenbankclientschnittstelle ändern. Wenn Sie Version 8 installiert haben, müssen Sie das Produkt deinstallieren oder für das Produkt eine Migration auf mindestens Version 9 durchführen, bevor Sie die DB2-Standardkopie ändern können. Sie müssen für das Produkt eine Migration auf mindestens Version 9.5 durchführen, bevor Sie die Standardkopie der IBM Datenbankclientschnittstelle ändern können.

Clientanwendungen können immer direkt an die Speicherposition des Datenservertreibers geleitet werden. Dies ist das Verzeichnis, in dem DSDRIVER installiert ist.

Wenn Sie die DB2-Kopie oder die IBM Daten Server-Treiberkopie, die die Standardkopie der IBM Datenbankclientschnittstelle ist, deinstallieren, werden die Standardkopien für Sie verwaltet. Ausgewählte Standardkopien werden entfernt und neue Standardkopien werden für Sie ausgewählt. Wenn Sie die DB2-Standardkopie deinstallieren, die jedoch nicht die letzte DB2-Kopie auf dem System ist, werden Sie aufgefordert, zunächst eine andere DB2-Kopie als Standardkopie festzulegen.

Auswählen einer Standardkopie beim Installieren einer neuen IBM Datenbankclientschnittstellenkopie

Ab Version 9.5 kommt ein Szenario in Betracht, in dem Sie zwei DB2-Kopien (DB2COPY1 und DB2COPY2) installiert haben können. DB2COPY2 ist die DB2-Standardkopie und die Standardkopie der IBM Datenbankclientschnittstelle.

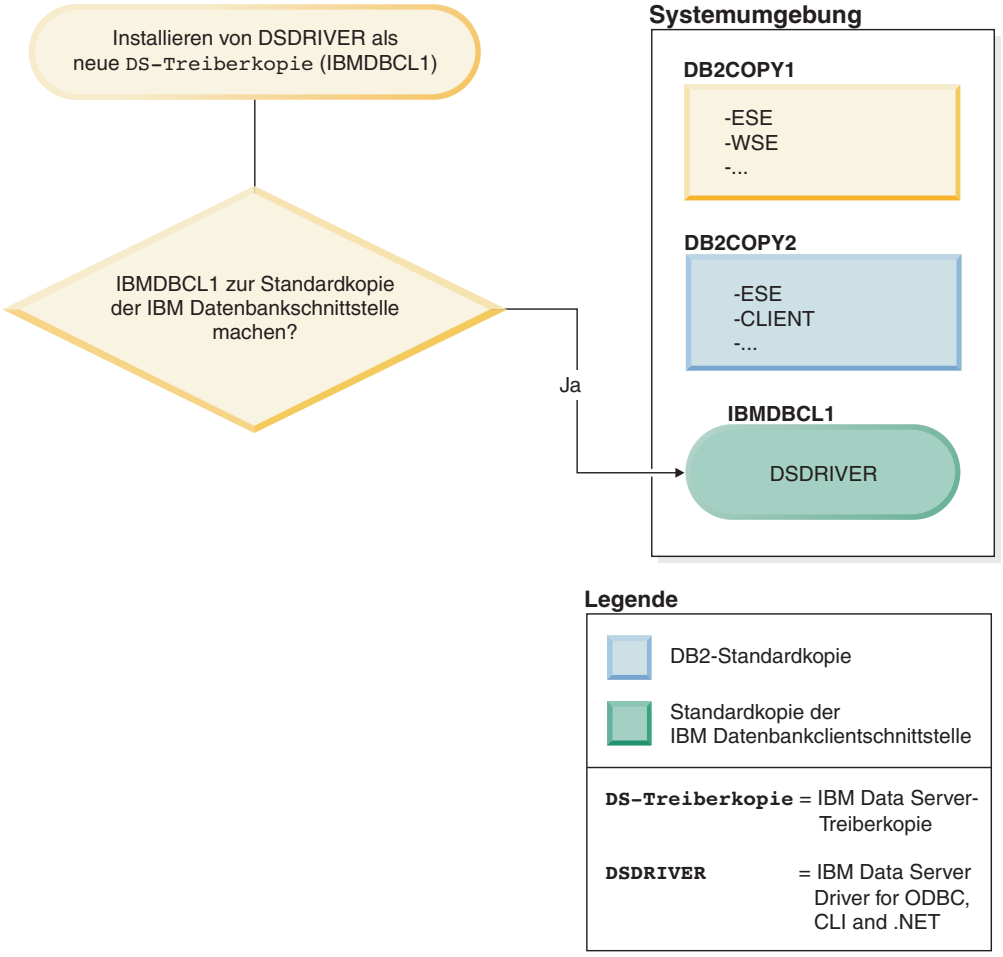


Sie installieren IBM Data Server Driver Package (DSDRIVER) auf einer neuen IBM Data Server-Treiberkopie.

Während der Installation der neuen IBM Data Server-Treiberkopie (IBMDBCL1) werden Sie gefragt, ob die neue IBM Data Server-Treiberkopie zur Standardkopie der IBM Datenbankclientschnittstelle gemacht werden soll.

Wenn Sie „Nein" antworten, wird DB2COPY2 als Standardkopie der IBM Datenbankclientschnittstelle beibehalten. (Außerdem fungiert diese Kopie weiter als DB2-Standardkopie.)

Betrachten Sie jedoch auch den Fall, in dem Sie im gleichen Szenario mit „Ja" auf die Frage antworten, ob die neue IBM Data Server-Treiberkopie zur Standardkopie der IBM Datenbankclientschnittstelle gemacht werden soll.



In diesem Fall wird IBMDBCL1 als Standardkopie der IBM Datenbankclientschnittstelle eingerichtet. (DB2COPY2 bleibt jedoch als DB2-Standardkopie erhalten.)

Einrichten des DB2-Verwaltungsservers bei Verwendung mehrerer DB2-Kopien

Seit Version 9 können Sie mehrere DB2-Kopien haben, die auf demselben Computer ausgeführt werden. Dies wirkt sich auf die Funktionsweise des DB2-Verwaltungsservers (DAS - DB2 Administration Server) aus. Der DAS ist eine einzigartige Komponente innerhalb des Datenbankmanagers, die auf eine aktive Version beschränkt ist, unabhängig davon, wie viele DB2-Kopien auf demselben Computer installiert sind. Aus diesem Grund gelten für ihn die folgenden Einschränkungen und funktionalen Anforderungen.

Auf dem Server darf nur eine DAS-Version installiert sein, die die Instanzen wie folgt verwaltet:

- Wenn der DAS unter Version 9.1 oder Version 9.5 ausgeführt wird, kann er zur Verwaltung von Instanzen der Version 8 und der Version 9.1 oder Version 9.5 verwendet werden.

- Wenn der DAS unter Version 8 ausgeführt wird, kann er nur Instanzen der Version 8 verwalten. Sie können den DAS der Version 8 migrieren oder löschen und anschließend einen neuen DAS der Version 9.5 erstellen, um die Instanzen der Version 8 und Version 9.1 zu verwalten. Dieser Arbeitsschritt ist nur erforderlich, wenn Sie die Steuerzentrale zur Verwaltung der Instanzen verwenden wollen.

Ungeachtet der Anzahl der auf ein und demselben Computer installierten DB2-Kopien kann nur ein DAS auf diesem Computer zu gleicher Zeit erstellt werden. Dieser DAS wird von allen DB2-Kopien verwendet, die auf demselben Computer ausgeführt werden. In Version 9 oder einer späteren Version kann der DAS zu jeder beliebigen DB2-Kopie gehören, die momentan installiert ist.

Zum Versetzen des DAS aus einer Kopie der Version 9.5 in eine andere Kopie der Version 9.5 verwenden Sie den Befehl dasupdt. Zum Versetzen des DAS aus einer Kopie der Version 9.1 in eine Kopie der Version 9.5 können Sie den Befehl dasupdt nicht verwenden, sondern müssen eine Migration von Version 9.1 auf Version 9.5 ausführen.

Unter Windows-Betriebssystemen können Sie den Befehl dasupdt auch verwenden, wenn Sie den DAS in eine neue DB2-Standardkopie derselben Version versetzen müssen.

Anmerkung:

- Der Befehl dasupdt kann nur zum Versetzen des DAS zwischen verschiedenen DB2-Kopien desselben Release (d. h. zwischen verschiedenen Fixpacks) verwendet werden. Er kann nicht zur Konfiguration des DAS verwendet werden.
- Zur Migration des DAS von Version 8 auf Version 9.1 und anschließend auf Version 9.5 verwenden Sie den Befehl dasmigr.
- Wenn der DAS nicht konfiguriert ist, sollte eine reguläre DAS-Konfigurationsprozedur ausgeführt werden, um ihn in einer der DB2-Kopien zu konfigurieren.

Festlegen der Standardinstanz bei Verwendung mehrerer DB2-Kopien (Windows)

Seit Version 9.1 wird die DB2INSTANCE-Umgebung entsprechend der DB2-Kopie festgelegt, auf deren Verwendung Ihre Umgebung momentan eingestellt ist. Wenn Sie sie nicht explizit auf eine Instanz in der aktuellen Kopie einstellen, wird die Standardinstanz verwendet, die durch die Variable DB2INSTDEF der Profilregistrierdatenbank angegeben ist.

DB2INSTDEF ist die Standardinstanzvariable, die für die momentan verwendete DB2-Kopie spezifisch ist. Jede DB2-Kopie besitzt eine eigene Profilregistrierdatenbankvariable DB2INSTDEF. Instanznamen müssen auf dem System eindeutig sein. Wenn eine Instanz erstellt wird, durchsucht der Datenbankmanager die vorhandenen Kopien, um die Eindeutigkeit des Namens sicherzustellen.

Beachten Sie bei der Festlegung der Standardinstanz die folgenden Richtlinien, wenn Sie mit mehreren DB2-Kopien arbeiten:

- Wenn DB2INSTANCE für eine bestimmte DB2-Kopie nicht festgelegt ist, wird der Wert der Variablen DB2INSTDEF für diese DB2-Kopie verwendet. Dies bedeutet Folgendes:
  - Wenn DB2INSTANCE=ABC ist und DB2INSTDEF=XYZ, wird der Wert ABC verwendet.

  - Wenn DB2INSTANCE *nicht definiert ist* und DB2INSTDEF=XYZ ist, wird der Wert XYZ verwendet.
  - Wenn DB2INSTANCE *nicht definiert ist* und DB2INSTDEF *nicht definiert ist*, funktionieren alle Anwendungen bzw. Befehle nicht, die von einem gültigen Wert für DB2INSTANCE abhängig sind.
- Sie können entweder den Befehl db2envar.bat oder die API 'db2SelectDB2Copy' verwenden, um zwischen DB2-Kopien zu wechseln. Es ist auch möglich, alle Umgebungsvariablen (z. B. PATH, INCLUDE, LIB und DB2INSTANCE) richtig zu definieren, jedoch müssen Sie sicherstellen, dass sie ordnungsgemäß definiert werden.

  Anmerkung: Die Verwendung des Befehls db2envar.bat ist nicht ganz identisch mit der Einstellung der Umgebungsvariablen. Der Befehl 'db2envar.bat' stellt fest, zu welcher DB2-Kopie er gehört, und fügt den Pfad dieser DB2-Kopie am Anfang der Umgebungsvariablen PATH an.

  Wenn mehrere DB2-Kopien auf demselben System vorhanden sind, kann die Umgebungsvariable PATH nur auf eine von ihnen verweisen: die Standardkopie (DEFAULT COPY). Zum Beispiel sei DB2COPY1 unter 'c:\sqllib\bin' vorhanden und die Standardkopie, während DB2COPY2 sich unter 'd:\sqllib\bin' befindet. Wenn Sie DB2COPY2 in einem regulären Befehlsfenster verwenden wollten, würden Sie 'd:\sqllib\bin\db2envar.bat' in diesem Befehlsfenster ausführen. Dadurch wird die Umgebungsvariable PATH (und einige andere Umgebungsvariablen) für dieses Befehlsfenster angepasst, sodass es die Binärdateien aus dem Verzeichnis 'd:\sqllib\bin' verwendet.
- DB2INSTANCE ist nur für die Instanzen unter der DB2-Kopie gültig, die Sie verwenden. Wenn Sie jedoch die Kopie wechseln, indem Sie den Befehl db2envar.bat ausführen, wird DB2INSTANCE mit dem Wert von DB2INSTDEF für die DB2-Kopie aktualisiert, zu der Sie anfangs gewechselt haben.
- DB2INSTANCE gibt die aktuelle DB2-Instanz an, die von Anwendungen verwendet wird, die in dieser DB2-Kopie ausgeführt werden. Wenn Sie zwischen Kopien wechseln, wird DB2INSTANCE standardmäßig in den Wert von DB2INSTDEF für die betreffende Kopie geändert. DB2INSTDEF hat in einem System mit nur einer Kopie geringere Bedeutung, weil sich alle Instanzen in der aktuellen Kopie befinden. Jedoch ist sie auch in diesem Fall als Standardinstanz verwendbar, wenn keine andere Instanz definiert ist.
- Alle globalen Variablen der Profilregistrierdatenbank sind für eine DB2-Kopie spezifisch, sofern Sie sie nicht mit dem Befehl SET VARIABLE=<variablenname> angeben.

Mehrere Instanzen des Datenbankmanagers

Auf einem einzigen Server können mehrere Instanzen des Datenbankmanagers erstellt werden. Das heißt, dass verschiedene Instanzen desselben Produkts auf einem physischen Computer erstellt und gleichzeitig ausgeführt werden können. Dadurch haben Sie flexible Möglichkeiten bei der Einrichtung von Umgebungen.

Anmerkung: Ein Instanzname darf nicht in zwei unterschiedlichen DB2-Kopien verwendet werden.

Mehrere Instanzen können z. B. zum Erstellen der folgenden Umgebungen wünschenswert sein:

- Trennen der Entwicklungsumgebung von der Geschäftsumgebung.

- Getrenntes Optimieren jeder Umgebung für bestimmte Anwendungen, die ausgeführt werden sollen.
- Schützen sensibler Informationen vor Administratoren. Es kann zum Beispiel erforderlich sein, die Lohn- und Gehaltsdaten in einer eigenen Instanz geschützt zu speichern, sodass Eigner anderer Instanzen diese Daten nicht einsehen können.

Anmerkung:

- (Nur für UNIX-Betriebssysteme:) Zur Vermeidung von Umgebungskonflikten zwischen zwei oder mehr Instanzen sollten Sie sicherstellen, dass sich die einzelnen Instanzausgangsverzeichnisse auf einem lokalen Dateisystem befinden.
- (Nur auf Windows-Plattformen:) Instanzen werden entweder als lokal (local) oder als fern (remote) im Knotenverzeichnis katalogisiert. Die Standardinstanz wird durch die Umgebungsvariable DB2INSTANCE definiert. Sie können die Verbindung zu anderen Instanzen (Befehl ATTACH) herstellen, um Wartungsaufgaben und Dienstprogramme auszuführen, die nur auf Instanzebene ausgeführt werden können, wie z. B. Erstellen einer Datenbank, Abbrechen von Anwendungen, Monitoraufzeichnungen für eine Datenbank oder Aktualisieren der Konfiguration des Datenbankmanagers. Wenn Sie versuchen, die Verbindung zu einer Instanz herzustellen, die nicht Ihre Standardinstanz ist, wird anhand des Knotenverzeichnisses festgestellt, wie die Kommunikation mit der anderen Instanz herzustellen ist.
- (Auf allen Plattformen:) DB2-Datenbankprogrammdateien werden physisch an einer Position gespeichert, und jede Instanz verweist wieder auf die Kopie, der diese Instanz angehört, sodass die Programmdateien bei der Erstellung der einzelnen erstellten Instanzen nicht kopiert werden. In einer Instanz können mehrere zusammengehörige Datenbanken gespeichert werden.

Mehrere Instanzen (Windows)

Es ist möglich, mehrere Instanzen des Datenbankmanagers auf demselben Computer auszuführen. Jede Instanz des Datenbankmanagers pflegt eigene Datenbanken und verfügt über eigene Konfigurationsparameter für den Datenbankmanager.

Anmerkung: Die Instanzen können auch unterschiedlichen DB2-Kopien auf einem Computer angehören, der unterschiedliche Versionen des Datenbankmanagers aufweisen kann.

Eine Instanz des Datenbankmanagers besteht aus folgenden Komponenten:

- Ein Windows-Dienst (Service), der die Instanz darstellt. Der Name des Service ist mit dem Namen der Instanz identisch. Der Anzeigename des Service (im Fenster 'Dienste') ist der Instanzname, dem die Zeichenfolge „DB2 - " vorangestellt ist. Zum Beispiel gibt es für eine Instanz mit dem Namen „DB2" einen Windows-Dienst mit dem Namen „DB2", der als „DB2 - <name_der_db2-kopie> - DB2" angezeigt wird.

  Anmerkung: Ein Windows-Dienst (Service) wird für Clientinstanzen nicht erstellt.
- Ein Instanzverzeichnis. Dieses Verzeichnis enthält die Konfigurationsdateien des Datenbankmanagers, das Systemdatenbankverzeichnis, das Knotenverzeichnis, das DCS-Verzeichnis, alle Diagnoseprotokolldateien sowie alle Speicherauszugsdateien, die der Instanz zugeordnet sind. Das Instanzverzeichnis ist standardmäßig ein Unterverzeichnis im Verzeichnis SQLLIB und hat den gleichen Namen wie die Instanz. Für die Instanz „DB2" ist dieses Verzeichnis zum Beispiel

C:\SQLLIB\DB2, wobei C:\SQLLIB das Verzeichnis ist, in dem der Datenbankmanager installiert ist. Sie können mithilfe der Registrierdatenbankvariablen DB2INSTPROF die Standardposition des Instanzverzeichnisses ändern. Wenn die Registrierdatenbankvariable DB2INSTPROF eine andere Position angibt, wird das Instanzverzeichnis unter dem Verzeichnis erstellt, auf das die Variable DB2INSTPROF zeigt. Wenn zum Beispiel DB2INSTPROF=D:\DB2PROFS definiert würde, wäre das Instanzverzeichnis D:\DB2PROFS\DB2.

  – Setzen Sie unter Verwendung des Befehls db2set.exe -g die Variable DB2INSTPROF auf c:\DB2PROFS.
  – Führen Sie den Befehl DB2ICRT.exe aus, um die Instanz zu erstellen.
- Wenn Sie unter Windows-Betriebssystemen eine Instanz erstellen, sind die Standardpositionen für Datendateien des Benutzers wie zum Beispiel Instanzverzeichnisse und die Datei 'db2cli.ini' die folgenden Verzeichnisse:
  – Dokumente und Einstellungen\All Users\Application Data\IBM\DB2\name_der_kopie unter den Betriebssystemen Windows XP und Windows 2003
  – ProgramData\IBM\DB2\name_der_kopie unter dem Betriebssystem Windows Vista

Aktualisieren von DB2-Kopien (Windows)

Beim Aktualisieren Ihres DB2-Produkts müssen Sie angeben, ob eine vorhandene DB2-Kopie aktualisiert oder eine neue Kopie installiert werden soll. Zum Aktualisieren einer DB2-Kopie müssen Sie die Option Mit Vorhandenen arbeiten auswählen.

Das gleichzeitige Aktualisieren mehrerer DB2-Kopien ist nicht möglich. Zur Aktualisierung weiterer DB2-Kopien, die auf demselben Computer installiert sind, müssen Sie die Installation erneut ausführen. Die Installation bietet die Option zur Migration einer Kopie von Version 8 - oder Version 9.1 (im gleichen Pfad) - oder zur Installation einer neuen Kopie von Version 9.1 oder Version 9.5 ohne Änderung der Installation von Version 8.

- Wenn Sie eine Migration durchführen wollen, wird die Installation der Version 8 vom System entfernt.
- Wenn Sie eine neue DB2-Kopie installieren, können Sie später Ihre Instanzen mit den Befehlen db2ckmig und db2imigr migrieren.

Mit dem Befehl db2iupdt können Sie eine DB2-Instanz zwischen unterschiedlichen DB2-Kopien der Version 9.1 oder Version 9.5 versetzen. Mit dem Befehl db2imigr können Sie eine Instanz der Version 8 auf Version 9.1 oder Version 9.5 umstellen.

Anmerkung:

- Eine Koexistenz von Version 7 und Version 9.1 oder Version 9.5 wird nicht unterstützt.
- Eine Koexistenz eines DB2-Datenservers (32 Bit) und eines DB2-Datenservers (64 Bit) auf demselben Windows X64-Computer wird nicht unterstützt.

  Es ist nicht möglich, eine Migration von einer 32-Bit-DB2-Installation der Version 8 auf einem X64-System auf eine 64-Bit-Installation der Version 9.1 oder Version 9.5 durchzuführen. Stattdessen müssen Sie eine Migration auf die Version 9.1 oder Version 9.5 mit 32 Bit ausführen, um die DB2-Datenserverinstallation auf dem X64-System für die Migration auf die 64-Bit-Version verwenden zu können. Die 32-Bit-Version wird entfernt. Wenn Sie mehrere 32-Bit-Kopien von DB2 installiert haben, müssen Sie alle Instanzen Ihres System in eine der DB2-Kopien versetzen und die anderen Kopien vom Computer entfernen.

- Wenn Sie mehrere Kopien von Version 9.1 oder Version 9.5 haben, können Sie bei der Installation auswählen, ob Sie eine neue Kopie installieren oder mit einer bereits vorhandenen DB2-Kopie arbeiten wollen, für die ein Upgrade ausgeführt bzw. der neue Funktionen hinzugefügt werden können. Die Migrationsaktion ist verfügbar, wenn Sie neben den Kopien der Version 9.1 oder Version 9.5 auch eine Kopie der Version 8 haben.
- Wenn Version 8 oder Version 9.1 installiert ist, können Sie bei der Installation auswählen, ob die vorhandene Kopie der Version 8 oder Version 9.1 auf eine Kopie der Version 9.5 migriert oder eine neue DB2-Kopie installiert werden soll.
- Wenn auf Ihrem System Version 7 oder eine ältere Version installiert ist, wird während der Installation eine Nachricht angezeigt, die Ihnen mitteilt, dass eine Migration auf Version 9.1 oder Version 9.5 nicht unterstützt wird. Sie können in diesem Fall nur eine neue DB2-Kopie installieren, nachdem Version 7 deinstalliert wurde. Dies bedeutet, dass Version 7 und Version 9.1 bzw. Version 9.5 nicht zusammen auf einem System ausgeführt werden können.
- Zum Versetzen einer Instanz aus einer Kopie der Version 9.1 oder Version 9.5 in eine andere können Sie den Befehl db2iupdt verwenden.
- Wenn Sie den Befehl db2imigr zum Migrieren Ihrer Instanzen von Version 8 verwenden, müssen Sie alle vorhandenen ODBC-Datenquellen rekonfigurieren.

Gleichzeitiges Ausführen mehrerer Instanzen (Windows)

Sie können mehrere Instanzen gleichzeitig ausführen, entweder in derselben DB2-Kopie oder in unterschiedlichen DB2-Kopien.

Gehen Sie unter Verwendung der Befehlszeile wie folgt vor, um mehrere Instanzen gleichzeitig in derselben DB2-Kopie auszuführen:

1. Setzen Sie die Variable DB2INSTANCE auf den Namen der anderen Instanz, die Sie starten wollen, indem Sie folgenden Befehl eingeben:

   ```
   set db2instance=<anderer_instanzname>
   ```

2. Geben Sie den Befehl db2start ein, um die Instanz zu starten.

Verwenden Sie eine der folgenden Methoden, wenn Sie mehrere Instanzen gleichzeitig in unterschiedlichen DB2-Kopien ausführen möchten:

- Mithilfe des DB2-Befehlsfensters über Start → Programme → IBM DB2 → *<name_der_db2-kopie>* → Befehlszeilentools → DB2-Befehlsfenster: Das Befehlsfenster wurde bereits mit den richtigen Umgebungsvariablen für die bestimmte ausgewählte DB2-Kopie eingerichtet.
- Mithilfe der Datei db2envar.bat über ein Befehlsfenster:
  1. Öffnen Sie ein Befehlsfenster.
  2. Führen Sie die Datei db2envar.bat aus, und verwenden Sie dabei den vollständig qualifizierten Pfad für die DB2-Kopie, die die Anwendung verwenden soll:

     ```
     <installationsverzeichnis_der_db2-kopie>\bin\db2envar.bat
     ```

Verwenden Sie die Methode im oben stehenden Abschnitt (″Gleichzeitiges Ausführen mehrerer Instanzen in derselben DB2-Kopie″), wenn Sie zu einer bestimmten DB2-Kopie gewechselt haben, um die Instanzen zu starten.

Arbeiten mit Instanzen in derselben oder in anderen DB2-Kopien

Sie können mehrere Instanzen gleichzeitig in derselben DB2-Kopie oder in verschiedenen DB2-Kopien ausführen.

Wenn Sie mit Instanzen in derselben DB2-Kopie arbeiten möchten, müssen Sie wie folgt vorgehen:

1. Erstellen Sie alle Instanzen in ein und derselben DB2-Kopie, oder migrieren Sie alle Instanzen auf ein und dieselbe DB2-Kopie.
2. Setzen Sie die Umgebungsvariable DB2INSTANCE auf den Namen der Instanz, mit der Sie arbeiten möchten, bevor Sie Befehle für diese Instanz absetzen.

Damit eine Instanz nicht auf die Datenbanken einer anderen Instanz zugreift, werden die Datenbankdateien für eine Instanz unter einem Verzeichnis erstellt, das den gleichen Namen wie die Instanz besitzt. Wenn zum Beispiel eine Datenbank auf dem Laufwerk `C:` für die Instanz „DB2" erstellt wird, werden die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\DB2` erstellt. Analog werden bei der Erstellung einer Datenbank auf Laufwerk `C:` für die Instanz TEST die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\TEST` erstellt. Standardmäßig entspricht der Wert dem Buchstaben des Laufwerks, in dem das DB2-Produkt installiert ist. Weitere Informationen finden Sie im Abschnitt zum Datenbankmanagerkonfigurationsparameter *dftdbpath*.

Verwenden Sie eine der folgenden Methoden, wenn Sie mit einer Instanz in einem System mit mehreren DB2-Kopien arbeiten möchten:

- Mithilfe des Befehlsfensters über Start → Programme → IBM DB2 → *<name_der_DB2-kopie>* → Befehlszeilentools → Befehlsfenster: Das Befehlsfenster ist bereits mit den richtigen Umgebungsvariablen für die jeweils ausgewählte DB2-Kopie eingerichtet.
- Mithilfe der Datei db2envar.bat über ein Befehlsfenster:
  1. Öffnen Sie ein Befehlsfenster.
  2. Führen Sie die Datei db2envar.bat aus, und verwenden Sie dabei den vollständig qualifizierten Pfad für die DB2-Kopie, die die Anwendung verwenden soll:

     ```
     <installationsverzeichnis_der_db2-kopie>\bin\db2envar.bat
     ```

Kapitel 3. Autonomic Computing

Die Autonomic Computing-Umgebung von DB2 bietet automatische Funktionen zur Selbstkonfiguration, zur Fehlerbehebung, Selbstoptimierung sowie zum Selbstschutz. Durch die Fähigkeit zur Erkennung eintretender Situationen und die Einleitung entsprechender Reaktionen verlagert Autonomic Computing die Verwaltungslast für eine IT-Umgebung von Datenbankadministratoren auf die Technologie.

Die folgenden automatischen Funktionen können Sie bei der Verwaltung Ihres Datenbanksystems unterstützen:

- Speicher mit automatischer Leistungsoptimierung
- Dynamischer Speicher
- Automatische Erstellung von Komprimierungswörterverzeichnissen (ADC)
- Automatische Datenbankbackups
- Automatische Statistikerfassung
- Konfigurationsadvisor
- Diagnosemonitor
- Drosselung von Dienstprogrammen

Automatische Funktionen

Automatische Funktionen unterstützen Sie bei der Verwaltung Ihres Datenbanksystems. Sie statten Ihr System mit der Möglichkeit aus, Selbstdiagnosen durchzuführen und Probleme vor dem Eintreten im Voraus zu erkennen. Zu diesem Zweck werden Echtzeitdaten analysiert und mit Langzeitproblemdaten verglichen. Einige automatische Tools können so konfiguriert werden, dass sie Änderungen an Ihrem System ohne Bedienereingriff durchführen und auf diese Weise Serviceunterbrechungen vermeiden helfen.

Beim Erstellen einer Datenbank werden einige der folgenden automatischen Funktionen standardmäßig aktiviert, während Sie andere manuell aktivieren müssen:

Speicher mit automatischer Leistungsoptimierung (nur Einzelpartitionsdatenbanken)

Die Funktion zur automatischen Speicherleistungsoptimierung vereinfacht die Aufgabe der Speicherkonfiguration. Diese Funktion reagiert auf signifikante Änderungen in der Auslastung, indem sie die Werte verschiedener Speicherkonfigurationsparameter sowie die Größen der Pufferpools automatisch und iterativ anpasst, um die Leistung zu optimieren. Die Speicheroptimierungsfunktion verteilt verfügbare Speicherressourcen dynamisch unter verschiedenen Speicherkonsumenten, zu denen die Sortierfunktion, der Paketcache, die Sperrenliste und Pufferpools gehören. Sie können die automatische Speicheroptimierung nach der Erstellung einer Datenbank inaktivieren, indem Sie den Datenbankkonfigurationsparameter **self_tuning_mem** auf den Wert OFF setzen.

Dynamischer Speicher

Die Funktion für dynamischen Speicher vereinfacht die Speicherverwaltung für Tabellenbereiche. Bei der Erstellung einer Datenbank geben Sie die Speicherpfade an, in denen der Datenbankmanager Ihre Tabellenbereichsdaten speichern soll. Anschließend verwaltet der Datenbank-

manager die Container und die Speicherzuordnung für die Tabellenbereiche, je nachdem, wie sie von Ihnen erstellt und mit Daten gefüllt werden.

Automatische Erstellung von Komprimierungswörterverzeichnissen (ADC)
Komprimierungswörterverzeichnisse (Compression Dictionaries) werden automatisch beim Einfügen von Daten in Tabellen, für die Sie das Attribut COMPRESS auf den Wert YES gesetzt haben, erstellt, wenn in der physischen Tabelle oder Partition noch kein Komprimierungswörterverzeichnis vorhanden ist und nachdem eine Tabelle eine Größe von ungefähr 1 MB infolge hinzugefügter Daten (z. B. durch INSERT- oder LOAD-Verarbeitung) erreicht hat. Das Wörterverzeichnis wird erstellt und in die Tabelle eingefügt. Vorausgesetzt, das Tabellenattribut COMPRESS bleibt aktiviert, unterliegen alle Daten der Komprimierung, die nach Erstellung des Komprimierungswörterverzeichnisses in die Tabelle eingefügt werden.

Automatische Datenbankbackups
Eine Datenbank kann durch eine Vielzahl möglicher Hard- und Softwarefehler unbrauchbar werden. Die Bereithaltung eines möglichst aktuellen Gesamtbackups Ihrer Datenbank ist ein integraler Bestandteil der Planung und Implementierung einer Strategie zur Wiederherstellung eines Systems nach einem Katastrophenfall. Verwenden Sie automatische Datenbankbackups im Rahmen Ihrer Wiederherstellungsstrategie, um den Datenbankmanager zu veranlassen, ordnungsgemäß und regelmäßig Backups Ihrer Datenbank durchzuführen.

Automatische Statistikerfassung
Die automatische Statistikerfassung unterstützt eine Verbesserung der Datenbankleistung, indem sie sicherstellt, dass aktuelle Tabellenstatistiken verfügbar sind. Der Datenbankmanager ermittelt, welche Statistiken für Ihre Auslastung erforderlich sind und welche Statistiken aktualisiert werden müssen. Statistiken können asynchron (im Hintergrund) oder synchron, durch Sammeln von Laufzeitstatistikdaten während der Kompilierung von SQL-Anweisungen, erfasst werden. Anschließend kann das DB2-Optimierungsprogramm einen Zugriffsplan auswählen, der auf präzisen Statistiken beruht. Sie können die automatische Statistikerfassung nach der Erstellung einer Datenbank inaktivieren, indem Sie den Datenbankkonfigurationsparameter **auto_runstats** auf den Wert OFF setzen. Die Echtzeitstatistikerfassung kann nur aktiviert werden, wenn die automatische Statistikerfassung aktiviert ist. Die Echtzeitstatistikerfassung wird durch den Konfigurationsparameter **auto_stmt_stats** gesteuert.

Konfigurationsadvisor
Wenn Sie eine Datenbank erstellen, wird dieses Tool automatisch ausgeführt, um die Datenbankkonfigurationsparameter und die Größe des Standardpufferpools (IBMDEFAULTBP) zu bestimmen und festzulegen. Die Werte werden auf der Basis der Systemressourcen und des für das System vorgesehenen Gebrauchs ausgewählt. Diese erste automatische Optimierung sorgt dafür, dass Ihre Datenbank eine bessere Leistung erreicht als eine äquivalente Datenbank, die Sie mit den Standardwerten erstellen könnten. Sie sorgt außerdem dafür, dass Sie im Anschluss an die Erstellung der Datenbank weniger Zeit für die Optimierung Ihres Systems aufwenden müssen. Sie können den Konfigurationsadvisor jederzeit (auch nachdem die Datenbanken mit Daten gefüllt wurden) ausführen, um das Tool empfohlene Werte für eine Gruppe von Konfigurationsparametern berechnen und optional anwenden zu lassen und die Leistung auf der Basis der aktuellen Kenndaten des Systems zu optimieren.

Diagnosemonitor

Der Diagnosemonitor ist ein serverseitiges Tool, das Situationen oder Änderungen in Ihrer Datenbankumgebung proaktiv überwacht, die zu Leistungseinbußen oder potenziellen Ausfällen führen können. Eine Reihe von Statusinformationen wird ohne jede Form aktiver Überwachung Ihrerseits erfasst. Wenn ein Risiko für den ordnungsgemäßen Betrieb festgestellt wird, werden Sie vom Datenbankmanager informiert und Sie erhalten Empfehlungen zur weiteren Verfahrensweise. Der Diagnosemonitor erfasst Informationen über das System mithilfe des Snapshot Monitors und verursacht keinerlei Leistungsbeeinträchtigung. Außerdem aktiviert er keine Snapshot Monitor-Schalter zur Erfassung von Informationen.

Drosselung von Dienstprogrammen

Diese Funktion reguliert die Leistungsbeeinträchtigung durch Wartungsdienstprogramme, sodass sie während der Produktionszeiten gleichzeitig ausgeführt werden können. Obwohl die *Richtlinie für die Auslastungswirkung* für gedrosselte Dienstprogramme standardmäßig definiert wird, müssen Sie die *Priorität der Auslastungswirkung* festlegen, wenn Sie ein Dienstprogramm gedrosselt ausführen wollen. Das Drosselungssystem stellt sicher, dass die gedrosselten Dienstprogramme so häufig wie möglich ausgeführt werden, ohne gegen die Richtlinie für die Auslastungswirkung zu verstoßen. Gegenwärtig können Operationen zur Statistikerfassung, Backup-Operationen, Datenumverteilungen und asynchrone Indexbereinigungen gedrosselt werden.

Automatische Verwaltung

Der Datenbankmanager stellt Funktionen zur automatischen Verwaltung zur Verfügung, um bei Bedarf Datenbankbackups, Statistikaktualisierungen und Reorganisationen von Tabellen und Indizes auszuführen. Die Ausführung von Verwaltungsaktivitäten an Ihren Datenbanken ist wichtig, um sicherzustellen, dass sie im Hinblick auf Leistung und Wiederherstellbarkeit optimal gepflegt werden.

Zur Verwaltung Ihrer Datenbank gehören einige oder alle der folgenden Aktivitäten:

- **Backups.** Wenn Sie ein Backup einer Datenbank durchführen, erstellt der Datenbankmanager eine Kopie der Daten in der Datenbank und speichert diese auf einem anderen Speichermedium für den Fall, dass die Originaldaten verloren gehen oder beschädigt werden. Automatische Datenbankbackups helfen bei der Gewährleistung, dass Ihre Datenbank ordnungsgemäß und regelmäßig gesichert wird, ohne dass Sie sich um den Zeitpunkt des Backups kümmern oder die Syntax des Befehls BACKUP kennen müssen.
- **Datendefragmentierung (Tabellen- oder Indexreorganisation).** Diese Verwaltungsaktivität kann die Effizienz erhöhen, mit der der Datenbankmanager auf Ihre Tabellen zugreift. Die automatische Reorganisation verwaltet eine Offlinereorganisation von Tabellen und Indizes, ohne dass Sie sich darum zu kümmern brauchen, wann und wie Ihre Daten zu reorganisieren sind.
- **Datenzugriffsoptimierung (Statistikerfassung).** Der Datenbankmanager aktualisiert die Statistikdaten in den Systemkatalogen für die Daten in einer Tabelle, für die Daten in Indizes oder für beide Arten von Daten (in einer Tabelle und den zugehörigen Indizes). Mit diesen Statistiken bestimmt das Optimierungsprogramm, welcher Pfad für den Zugriff auf die Daten verwendet wird. Die automatische Statistikerfassung versucht, die Leistung der Datenbank durch die Pflege aktueller Tabellenstatistiken zu verbessern. Dies dient dazu, dem Optimierungsprogramm die Möglichkeit zu geben, einen Zugriffsplan auf der Grundlage exakter Statistikdaten auszuwählen.

- **Statistikprofilerstellung.** Die automatische Statistikprofilerstellung gibt Empfehlungen, wann und wie Tabellenstatistiken zu erfassen sind, indem sie veraltete, fehlende oder falsche Statistikdaten erkennt und Statistikprofile auf der Basis des Abfragefeedbacks generiert.

Die Feststellung, ob und wann Verwaltungsaktivitäten auszuführen sind, kann zeitaufwendig sein. Die automatischen Verwaltungsfunktionen nehmen Ihnen diese Aufgabe jedoch ab. Sie können die Aktivierung dieser automatischen Verwaltungsfunktionen einfach und flexibel mithilfe der Datenbankkonfigurationsparameter für die automatische Verwaltung steuern. Über den Assistenten 'Automatische Verwaltung konfigurieren' können Sie Ihre Verwaltungsziele angeben. Der Datenbankmanager bestimmt anhand dieser Verwaltungsziele, ob die Verwaltungsaktivitäten ausgeführt werden müssen, und führt während des nächsten verfügbaren Verwaltungsfensters (ein von Ihnen definierter Zeitraum) nur die erforderlichen Verwaltungsaktivitäten aus.

Verwaltungsfenster

Ein Verwaltungsfenster ist ein von Ihnen definierter Zeitraum für die Ausführung automatischer Verwaltungsaktivitäten. Solche Verwaltungsaktivitäten sind Backups, Statistikerfassungen, Statistikprofilerstellungen und Reorganisationen. Ein Offlinefenster kann der Zeitraum sein, in dem der Zugriff auf eine Datenbank nicht verfügbar ist. Ein Onlinefenster kann der Zeitraum sein, in dem Benutzer die Verbindung zu einer Datenbank herstellen können.

Ein Verwaltungsfenster ist nicht dasselbe wie eine Taskzeitplanung. Während eines Verwaltungsfensters wird nicht unbedingt jede automatische Verwaltungsaktivität ausgeführt. Vielmehr bewertet der Datenbankmanager das System, um den Ausführungsbedarf für die einzelnen Verwaltungsaktivitäten zu bestimmen. Wenn die Verwaltungsanforderungen nicht erfüllt sind, wird die Verwaltungsaktivität ausgeführt. Befindet sich die Datenbank in einem guten Verwaltungszustand, wird die Verwaltungsaktivität nicht ausgeführt.

Sie müssen sich überlegen, wann die automatischen Verwaltungsaktivitäten ausgeführt werden sollen. Die automatischen Verwaltungsaktivitäten beanspruchen Ressourcen auf Ihrem System und können bei der Ausführung die Leistung Ihrer Datenbank beeinträchtigen. Einige dieser Aktivitäten schränken zudem den Zugriff auf Tabellen, Indizes und Datenbanken ein. Daher müssen Sie geeignete Zeitfenster angeben, während deren der Datenbankmanager Verwaltungsaktivitäten ausführen kann. Sie können diese Zeiträume als Offline- und Onlineverwaltungsfenster angeben, indem Sie den Assistenten 'Automatische Verwaltung konfigurieren' über die Steuerzentrale oder die Diagnosezentrale verwenden.

Offlineverwaltungsaktivitäten

Offlineverwaltungsaktivitäten (Offline-Datenbank-Backups sowie Tabellen- und Indexreorganisationen) sind Verwaltungsaktivitäten, die nur im Offlineverwaltungsfenster stattfinden können. Das Ausmaß, in dem der Benutzerzugriff eingeschränkt wird, ist von der jeweils ausgeführten Verwaltungsaktivität abhängig:

- Während eines Offline-Backups können Anwendungen keine Verbindung zur Datenbank herstellen. Alle gerade verbundenen Anwendungen werden zwangsweise getrennt.
- Während einer offline ausgeführten Tabellen- oder Indexreorganisation (Datendefragmentierung) können Anwendungen auf die Daten in den Tabellen zugreifen, diese jedoch nicht aktualisieren.

Offlineverwaltungsaktivitäten werden bis zum Ende ausgeführt, selbst wenn sie dabei über das angegebene Zeitfenster hinausgehen. Mit der Zeit erfasst der interne Zeitplanmechanismus, wie sich Jobausführungszeiten am besten abschätzen lassen. Wenn das Offlineverwaltungsfenster für eine bestimmte Datenbankbackup- oder Reorganisationsaktivität zu klein ist, startet die Planungsfunktion (Scheduler) den Job beim nächsten Mal nicht wieder und überlässt es dem Diagnosemonitor, eine Benachrichtigung über eine erforderliche Erweiterung des Offlineverwaltungsfensters auszugeben.

Onlineverwaltungsaktivitäten

Onlineverwaltungsaktivitäten (automatische Statistikerfassung und -profilerstellung, Onlineindexreorganisationen und Online-Datenbank-Backups) sind Verwaltungsaktivitäten, die nur im Onlineverwaltungsfenster stattfinden können. Während der Ausführung von Onlineverwaltungsaktivitäten können alle bereits verbundenen Anwendungen verbunden bleiben und auch neue Verbindungen hergestellt werden. Zur Minimierung der Auswirkungen auf das System werden Online-Datenbank-Backups sowie die automatische Statistikerfassung und Statistikprofilerstellung durch einen adaptiven Drosselmechanismus für Dienstprogramme gedrosselt.

Onlineverwaltungsaktivitäten werden bis zum Ende ausgeführt, selbst wenn sie dabei über das angegebene Zeitfenster hinausgehen.

Speicher mit automatischer Leistungsoptimierung

Mit DB2 Version 9 wird eine neue Speicheroptimierungsfunktion eingeführt, welche die Speicherkonfiguration vereinfacht, indem sie automatisch Werte für verschiedene Speicherkonfigurationsparameter einstellt. Wenn sie aktiviert ist, verteilt die Speicheroptimierungsfunktion verfügbare Speicherressourcen dynamisch unter folgenden Speicherkonsumenten: Pufferpools, Paketcache, Sperrenspeicher und Sortierspeicher.

Die Optimierungsfunktion arbeitet innerhalb der Speicherbegrenzungen, die durch den Konfigurationsparameter **database_memory** definiert sind. Der Wert des Parameters **database_memory** selbst kann ebenfalls automatisch optimiert werden. Wenn die automatische Optimierung für den Parameter **database_memory** aktiviert wird (d. h., wenn Sie ihn auf den Wert AUTOMATIC setzen), bestimmt die Optimierungsfunktion den Gesamtspeicherbedarf für die Datenbank und erhöht bzw. verringert die Menge an Speicher, die für den gemeinsam genutzten Datenbankspeicher zugeordnet wird, entsprechend den aktuellen Anforderungen der Datenbank. Wenn zum Beispiel der aktuelle Bedarf der Datenbank hoch ist und ausreichend freier Speicher auf dem System zur Verfügung steht, wird mehr Speicher für den gemeinsam genutzten Datenbankspeicher in Anspruch genommen. Wenn der Bedarf an Datenbankspeicher sinkt oder die Größe des freien Speichers auf dem System auf einen zu niedrigen Wert zurückgeht, wird ein Teil des gemeinsam genutzten Datenbankspeichers freigegeben.

Wenn Sie den Parameter **database_memory** nicht für die automatische Optimierung aktivieren, (d. h., wenn Sie ihn nicht auf AUTOMATIC setzen), verwendet die gesamte Datenbank die für den Parameter angegebene Größe an Speicher und verteilt sie nach Bedarf an die Konsumenten des Datenbankspeichers. Sie können die von der Datenbank verwendete Speichergröße auf zwei Arten angeben: durch Setzen des Parameters **database_memory** entweder auf einen numerischen Wert oder auf den Wert COMPUTED. Im zweiten Fall wird der Gesamtspeicher auf der Basis der Summe der Anfangswerte der Datenbankzwischenspeicher beim Starten der Datenbank berechnet.

Neben der Optimierung des gemeinsam genutzten Datenbankspeichers durch eine entsprechende Definition des Konfigurationsparameters **database_memory** können Sie auch andere Speicherkonsumenten wie folgt für die automatische Optimierung aktivieren:

- Für Pufferpools verwenden Sie die Anweisungen ALTER BUFFERPOOL und CREATE BUFFERPOOL.
- Für den Paketcache verwenden Sie den Konfigurationsparameter **pckcachesz**.
- Für den Sperrenspeicher verwenden Sie die Konfigurationsparameter **locklist** und **maxlocks**.
- Für den Sortierspeicher verwenden Sie die Konfigurationsparameter **sheapthres_shr** und **sortheap**.

Hauptspeicherzuordnung in DB2

Die Hauptspeicherzuordnung bzw. die Aufhebung der Hauptspeicherzuordnung tritt zu unterschiedlichen Zeiten in DB2 auf. Hauptspeicher kann einem bestimmten Hauptspeicherbereich zugeordnet werden, wenn ein angegebenes Ereignis auftritt, wie z. B. die Herstellung einer Verbindung zu einer Anwendung. Er kann darüber hinaus infolge einer geänderten Konfigurationsparametereinstellung neu zugeordnet werden.

In der nachfolgenden Abbildung werden die unterschiedlichen Bereiche des Hauptspeichers gezeigt, die der Datenbankmanager für unterschiedliche Zwecke zuordnet, sowie die Konfigurationsparameter angegeben, mit denen Sie die Größe dieses Speichers steuern können. Beachten Sie, dass in einer Enterprise Server Edition-Umgebung, die mehrere logische Datenbankpartitionen umfasst, jede Datenbankpartition über einen eigenen Bereich für den gemeinsam genutzten Speicher des Datenbankmanagers verfügt.

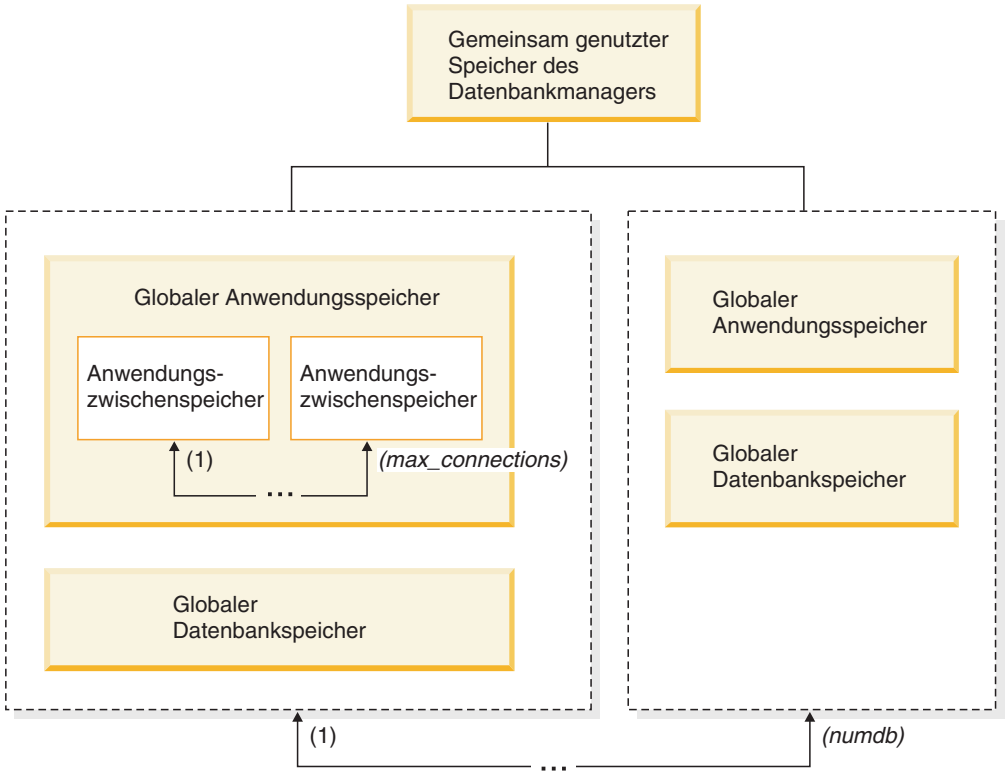


Abbildung 1. Vom Datenbankmanager verwendete Speichertypen

Der Speicher wird für jede Instanz des Datenbankmanagers zugeordnet, wenn die folgenden Ereignisse eintreten:

- **Wenn der Datenbankmanager gestartet wird (`db2start`):** Der globale gemeinsam genutzte Speicher (gemeinsam genutzter Speicher der Instanz) des Datenbankmanagers wird zugeordnet und bleibt so lange zugeordnet, bis der Datenbankmanager gestoppt wird (`db2stop`). Dieser Bereich enthält Informationen, die der Datenbankmanager zur Verwaltung von Aktivitäten für alle Datenbankverbindungen verwendet. Die Größe des globalen gemeinsam genutzten Speichers des Datenbankmanagers wird von DB2 automatisch gesteuert.
- **Wenn eine Datenbank aktiviert oder zum ersten Mal eine Verbindung zu ihr hergestellt wird:** Der globale Datenbankspeicher wird zugeordnet. Der globale Datenbankspeicher wird für alle Anwendungen verwendet, die eine Verbindung zur Datenbank herstellen. Die Größe des globalen Datenbankspeichers wird durch den Konfigurationsparameter **database_memory** festgelegt. Standardmäßig ist dieser Parameter auf automatic gesetzt, sodass DB2 die Anfangsgröße des Speichers, der der Datenbank zugeordnet wird, berechnen und die Größe des Datenbankspeichers während der Laufzeit automatisch je nach Bedarf der Datenbank konfigurieren kann. Sie können **database_memory** definieren, damit zu Anfang mehr Speicher als benötigt zugeordnet wird, sodass der zusätzliche Speicher später dynamisch verteilt werden kann.

  Die folgenden Speicherbereiche können dynamisch angepasst werden, um beispielsweise Speicher zu verkleinern, der einem Bereich zugeordnet ist, und Speicher zu vergrößern, der einem anderen Bereich zugeordnet ist.
  - Pufferpools (unter Verwendung der DDL-Anweisung ALTER BUFFERPOOL)
  - Datenbankzwischenspeicher (einschließlich Protokollpuffer)
  - Zwischenspeicher für Dienstprogramme
  - Paketcache

- Katalogcache
- Sperrenliste (Dieser Speicherbereich kann nur dynamisch vergrößert und nicht verkleinert werden.)

In einer Umgebung, in der der Konfigurationsparameter für partitionsinterne Parallelität (intra_parallel) des Datenbankmanagers aktiviert ist, in einer Umgebung, in der der Verbindungskonzentrator aktiviert ist, oder in einer Umgebung, in der die Datenbankpartitionierungsfunktion (DPF) aktiviert ist, wird auch der gemeinsam genutzte Sortierspeicher als Teil des globalen Datenbankspeichers zugeordnet. Wenn der Konfigurationsparameter **sheapthres** des Datenbankmanagers auf den Wert 0 (Standardwert) gesetzt ist, verwenden zudem alle Sortiervorgänge den globalen Datenbankspeicher.

- **Wenn eine Anwendung die Verbindung zu einer Datenbank herstellt:** Ein Anwendungszwischenspeicher wird zugeordnet. Jede Anwendung verfügt über einen eigenen Anwendungszwischenspeicher. Falls erwünscht, können Sie die Speicherkapazität, die jede einzelne Anwendung zuordnen kann, mithilfe des Konfigurationsparameters **applheapsz** begrenzen, oder die Kapazität des Anwendungsspeichers insgesamt mit dem Konfigurationsparameter **appl_memory** begrenzen.

  Der Konfigurationsparameter **max_connections** des Datenbankmanagers legt eine obere Begrenzung für die Anzahl der Anwendungen fest, die eine Verbindung zur Instanz (ATTACH) oder zu beliebigen Datenbanken, die in der Instanz vorhanden sind, herstellen können. Da jede Anwendung, die die Verbindung zu einer Datenbank herstellt, die Zuordnung von Speicher erforderlich macht, entsteht durch das Zulassen einer höheren Anzahl gleichzeitiger Anwendungen potenziell ein höherer Speicherbedarf.
- **Bei der Erstellung eines Agenten:** Der private Agentenspeicher wird für einen Agenten zugeordnet, wenn der Agent infolge einer Verbindungsanforderung oder einer neuen SQL-Anforderung in einer parallelen Umgebung zugeordnet wird. Der private Agentenspeicher wird für den Agenten zugeordnet und enthält Speicher, der nur von diesem bestimmten Agenten verwendet wird, wie zum Beispiel der private Sortierspeicher.

Die Abbildung gibt außerdem die folgenden Konfigurationsparametereinstellungen an, die die Größe des Speichers begrenzen, der für die einzelnen Speicherbereichstypen zugeordnet wird. Beachten Sie, dass dieser Speicher in einer partitionierten Datenbankumgebung in jeder Datenbankpartition zugeordnet wird.

- **numdb**

  Dieser Parameter gibt die maximale Anzahl gleichzeitig aktiver Datenbanken an, die von verschiedenen Anwendungen verwendet werden können. Da jede Datenbank über einen eigenen globalen Speicherbereich verfügt, wächst die Menge an Speicher, die möglicherweise zugeordnet wird, wenn Sie den Wert dieses Parameters erhöhen.
- **maxappls**

  Dieser Parameter definiert die maximale Anzahl von Anwendungen, die gleichzeitig mit einer einzigen Datenbank verbunden sein können. Er wirkt sich auf die Menge an Speicher aus, die möglicherweise für privaten Agentenspeicher und globalen Anwendungsspeicher für diese Datenbank zugeordnet wird. Beachten Sie, dass dieser Parameter für jede Datenbank unterschiedlich eingestellt werden kann.

  Zwei weitere Parameter, die berücksichtigt werden müssen, sind **max_coordagents** und **max_connections**, die sich beide auf die Instanzebene (in einer DPF-Instanz pro Knoten) beziehen.
- **max_connections**

Dieser Parameter begrenzt die Anzahl von Verbindungen (CONNECT) oder Instanzverbindungen (ATTACH), die auf den DB2-Server gleichzeitig zugreifen können (in einer DPF-Instanz pro Knoten).

- **max_coordagents**

  Dieser Parameter begrenzt die Anzahl von koordinierenden Agenten des Datenbankmanagers, die gleichzeitig für alle aktiven Datenbanken in einer Instanz (in einer DPF-Instanz pro Knoten) vorhanden sein können. Zusammen mit den Parametern **maxappls** und **max_connections** begrenzt dieser Parameter die Größe des Speichers, der für den privaten Agentenspeicher und den globalen Anwendungsspeicher zugeordnet wird.

Mit dem Speichertracker, der über den Befehl db2mtrk aufgerufen wird, können Sie die momentane Speicherzuordnung in der Instanz anzeigen, einschließlich der folgenden Informationstypen für die einzelnen Speicherpools:

- Aktuelle Größe
- Maximalgröße (fester Grenzwert)
- Größte erreichte Größe (obere Grenze)

Betriebsmerkmale und Einschränkungen von Speicher mit automatischer Leistungsoptimierung

Ermitteln der Optimierungsanforderungen

Zur Sicherstellung einer sachgerechten und relevanten Vergleichbarkeit von Speicherkonsumenten wurde eine neue einheitliche Metrik entwickelt. Jeder optimierte Speicherkonsument berechnet eine Voraussage über den durch zusätzlichen Speicher erzielbaren Nutzen und meldet diesen Wert an den Prozess der automatischen Speicheroptimierung. Der Speicher mit automatischer Leistungsoptimierung verwendet diese Werte als Basis für die Speicheroptimierung, indem er Speicherressourcen von Konsumenten mit dem geringsten Bedarf abzieht und Speicher den Speicherkonsumenten zuordnet, die am meisten davon profitieren.

Häufigkeit der Speicheroptimierung

Wenn die automatische Leistungsoptimierung aktiviert ist, überprüft sie in regelmäßigen Abständen Veränderungen der Datenbankauslastung. Wenn die Auslastung nicht konstant ist (d. h., wenn die ausgeführten Abfragen keine ähnlichen Speichermerkmale zeigen), ordnet die Speicheroptimierung Speicher weniger häufig (in Abständen von bis zu zehn Minuten zwischen Optimierungszyklen) zu, um stabilere Voraussagen von Trends zu erzielen. Bei Auslastungen mit konstanteren Speichernutzungsprofilen führt die Speicheroptimierung häufigere Optimierungen durch (in auf bis zu 30 Sekunden verkürzten Abständen zwischen Optimierungszyklen), um die beste Speicherkonfiguration rascher zu erreichen.

Verfolgen des Prozesses der automatischen Speicheroptimierung

Die aktuelle Hauptspeicherkonfiguration kann mithilfe des Befehls GET DATABASE CONFIGURATION oder durch eine Momentaufnahme (Snapshot) abgerufen werden. Änderungen, die durch die automatische Leistungsoptimierung erfolgen, werden in den Protokolldateien der Speicheroptimierung im Verzeichnis 'stmmlog' aufgezeichnet. Die Protokolldateien der Speicheroptimierung enthalten Zusammenfassungen der Ressourcenbedarfsdaten für jeden Speicherkonsumenten in jedem Optimierungsintervall. Diese Intervalle können anhand der Zeitmarken in den Protokolleinträgen ermittelt werden.

Erwartete Zeit für das Erreichen der besten Konfiguration

Wenn diese Funktion aktiviert bleibt, sollte dies rasch zu einer optimalen Einstellung von Parametern zur Optimierung der Speichernutzung führen. Ein System kann ausgehend von einer Erstkonfiguration bereits innerhalb von einer Stunde optimiert werden. In den meisten Fällen wird die Optimierung in der Regel in maximal 10 Stunden abgeschlossen. Dieser Extremfall tritt auf, wenn Abfragen, die an der Datenbank ausgeführt werden, deutliche Unterschiede in ihren Speicherbedarfsmerkmalen aufweisen.

Einschränkungen der automatischen Speicheroptimierung

In Fällen, in denen nur kleine Speichergrößen verfügbar sind (z. B., weil der Wert des Parameters *database_memory* sehr niedrig eingestellt ist oder weil mehrere Datenbanken, Instanzen oder andere Anwendungen auf dem Server ausgeführt werden), sind die durch die automatische Speicheroptimierung erzielbaren Vorteile begrenzt.

Da bei der automatischen Speicheroptimierung Optimierungsentscheidungen auf der Basis der Datenbankauslastung erfolgen, schränken Auslastungen mit wechselnden Speichernutzungsmerkmalen die Möglichkeiten für eine effektive Optimierung durch die Funktion der automatischen Leistungsoptimierung ein. Wenn sich die Speichernutzungsmerkmale Ihrer Auslastung beständig ändern, führt die automatische Speicheroptimierung weniger häufige Optimierungen durch und richtet diese Optimierungen wiederholt auf wechselnde Zielbedingungen aus. In diesem Fall kann die automatische Speicheroptimierung keine absolute Konvergenz auf eine optimale Speicherkonfiguration erzielen, sondern versucht stattdessen eine Speicherkonfiguration zu erhalten, die auf die aktuelle Auslastung optimal zugeschnitten ist.

Operative Details, Einschränkungen und Interaktionen zwischen Speicherparametern

Obwohl Sie den Speicher mit automatischer Leistungsoptimierung aktivieren und die Standardeinstellung AUTOMATIC für die meisten auf Speicher bezogenen Konfigurationsparameter verwenden können, kann es sich als nützlich erweisen, die operativen Details, Einschränkungen und Interaktionen zwischen den Speicherparametern zu kennen. Hierbei sind vor allem die Interaktionen zwischen den unterschiedlichen Speicherparametern **instance_memory**, **database_memory** und **appl_memory** zu nennen, damit Sie ihre Einstellung besser steuern können und auch verstehen, warum Fehler aufgrund von „Speicherknappheit" unter bestimmten Bedingungen noch immer möglich sind.

Zweck

Der DB2-Datenbankmanager verwendet hauptsächlich zwei Speichertypen:

- *Cachebasierter Speicher* wird vom Speichermanager mit automatischer Leistungsoptimierung (STMM - Self-Tuning Memory Manager) gesteuert und auf die verschiedenen Zwischenspeicher für die Leistung verteilt. Der Konfigurationsparameter **database_memory** kann verwendet werden, um den Maximalwert des verwendbaren cachebasierten Speichers einzuschränken. Alternativ kann dieser Konfigurationsparameter auf AUTOMATIC gesetzt werden, um dem Speichermanager mit automatischer Leistungsoptimierung die Verwaltung des gesamten cachebasierten Speichers zu überlassen.
- *Funktionaler Speicher* wird von Anwendungsprogrammen verwendet. Der Konfigurationsparameter **appl_memory** wird verwendet, um die maximale

Menge des Anwendungsspeichers bzw. funktionalen Speichers zu steuern, die Serviceanwendungsanforderungen von DB2-Datenbankagenten zugeordnet wird. Standardmäßig ist der Wert dieses Parameters auf AUTOMATIC gesetzt, dies bedeutet, dass Anforderungen für Anwendungsspeicher zulässig sind, wenn die gesamte von der Datenbankpartition zugeordnete Speichermenge sich innerhalb der Grenzwerte des Parameters **instance_memory** bewegt.

Prozess

In früheren Releases waren verschiedene Betriebssystem- und DB2-Tools verfügbar, um unterschiedliche Speicherabschnitte anzuzeigen, wie z. B. gemeinsam genutzten Speicher, privaten Speicher, Pufferpoolspeicher, Sperrenlisten, Sortierspeicher und so weiter, aber es war nahezu unmöglich, den Gesamtspeicher anzuzeigen, der vom DB2-Datenbankmanager verwendet wurde. Hatte einer der Zwischenspeicher den Speichergrenzwert erreicht, würde eine Anweisung in einer Anwendung mit der Fehlernachricht über „Speicherknappheit" fehlschlagen. Der Datenbankadministrator konnte den Speicher für diesen Zwischenspeicher vergrößern und die Anwendung erneut ausführen, um daraufhin einen Fehler über „Speicherknappheit" für einen anderen Zwischenspeicher zu erhalten. Jetzt können einzelne feste obere Grenzwerte für funktionale Zwischenspeicher unter Verwendung der Standardeinstellung des Konfigurationsparameters AUTOMATIC entfernt werden.

Bei Bedarf (z. B. um Szenarios zu vermeiden, in denen eine Datenbankanwendung mit schlechter Leistung extrem große Speichermengen benötigt) kann ein Grenzwert für den gesamten Anwendungsspeicher auf Datenbankebene mit dem Konfigurationsparameter **appl_memory** angewendet werden. Es können bei Bedarf auch einzelne Zwischenspeichergrenzwerte angewendet werden, indem die Einstellung AUTOMATIC des entsprechenden Datenbankkonfigurationsparameters für den Zwischenspeicher in einen festen Wert geändert wird. Wenn für alle funktionalen Zwischenspeicher die Standardeinstellung AUTOMATIC beibehalten wird, und für den Parameter **appl_memory** ebenfalls die Standardeinstellung AUTOMATIC ausgewählt ist, dann ist die Einstellung **instance_memory** der einzige Grenzwert für die Anwendungsspeicherbelegung. Wenn der Parameter **instance_memory** auch auf AUTOMATIC gesetzt ist, dann ermittelt DB2 automatisch eine Obergrenze für die Speicherbelegung. Datenbankadministratoren können problemlos die Gesamtsumme, die von **instance_memory** belegt wird, wie auch den aktuellen Grenzwert von **instance_memory** anzeigen, indem Sie die Tabellenfunktion 'admin_get_dbp_mem_usage' verwenden.

Interaktion zwischen den Konfigurationsparametern 'self_tuning_mem', 'instance_memory', 'database_memory' und 'appl_memory'

Wenn der Speicher mit automatischer Leistungsoptimierung vollständig aktiviert ist (**self_tuning_mem** ist auf ON und alle Speicherparameter sind auf AUTOMATIC gesetzt), überprüft der Speichermanager mit automatischer Leistungsoptimierung den freien Speicher, der auf dem System verfügbar ist, und stellt automatisch fest, wie viel Speicher für cachebasierte Zwischenspeicher dediziert sein sollte, um eine optimale Leistung zu erzielen. Alle cachebasierten Zwischenspeicher tragen zur Gesamtgröße von **database_memory** bei. Neben dem cachebasierten Speicherbedarf wird ferner etwas Speicher benötigt, um den Betrieb und die Integrität des DB2-Datenbankmanagers sicherzustellen. Die Differenz zwischen **instance_memory** und diesen zwei Speicherkonsumenten ist der Speicheranteil, der für die Verwendung als Anwendungsspeicher (**appl_memory**) zur Verfügung steht. Der funktionale Speicher für Anwendungsprogramme wird anschließend nach Bedarf zugeordnet, so lange er den Grenzwert von **instance_memory** nicht

überschreitet, sind keine weiteren Einschränkungen dafür vorhanden, wie viel Speicher eine einzelne Anwendung zuordnen kann.

Der Speichermanager mit automatischer Leistungsoptimierung fragt ebenfalls regelmäßig ab, wie viel freier Systemspeicher übrig ist und wie viel freier Speicher des Typs **instance_memory** übrig ist. Der Speichermanager mit automatischer Leistungsoptimierung misst Anwendungsanforderungen mehr Gewicht bei als Leistungskriterien (um Anwendungsfehler zu verhindern), er wird daher Leistungseinbußen durch das Verkleinern der cachebasierten Zwischenspeicher in Kauf nehmen, um sicherzustellen, dass genügend freier Systemspeicher und **instance_memory** für Anwendungsspeicheranforderungen zur Verfügung stehen. Nach Beendigung der Anwendungen wird der belegte Speicher freigegeben und kann sofort wieder durch andere Anwendungen erneut belegt werden oder zur Verwendung als **database_memory** vom Speichermanager mit automatischer Leistungsoptimierung freigegeben werden. Sinkt die Leistung des Datenbanksystems in Zeiträumen mit starker Anwendungsaktivität unter ein akzeptables Maß, kann es ratsam sein, entweder verfügbare Angaben dazu zu machen, wie viele Anwendungen im Datenbankmanager zulässig sind (z. B. unter Verwendung des Verbindungskonzentrators oder der neuen Workload-Manager-Komponente von DB2 9.5), oder das Hinzufügen zusätzlicher Speicherressourcen zum System in Betracht zu ziehen.

Einschränkungen (Fälle, in denen Fehler aufgrund von “Speicherknappheit” noch möglich sind)

In einigen Fällen können Sie noch Fehler aufgrund von „Speicherknappheit” erhalten, wenn dem Speichermanager mit automatischer Leistungsoptimierung nicht genügend Zeit zur Verfügung steht, um auf plötzlich auftretende Spitzen bei der Speicherbelegung zu reagieren, wenn z. B. eine Anwendung plötzlich eine große Speichermenge benötigt oder wenn es zu einer plötzlichen Spitze bei Ihrer Datenbankauslastung kommt (d. h. viele neue Anwendungen, die gleichzeitig eine Verbindung zu Ihrer Datenbank herstellen). In diesem Fall oder in den Fällen, von denen ein Datenbankadministrator weiß, dass die meisten Anwendungen eine festgelegte Speichermenge belegen, kann es besser sein, einen fest codierten Wert für den Parameter **appl_memory** anstelle der Einstellung AUTOMATIC zu verwenden. Wenn der Parameter **appl_memory** auf einen festen Wert, z. B. 2 GB, gesetzt wird, wird DB2 anschließend nicht zulassen, dass die gesamte Anwendungsspeicherbelegung diese Menge überschreitet. Jeder Anwendung wird dann gestattet, so viel Speicher wie nötig zu belegen, so lange die gesamte Anwendungsspeicherbelegung unterhalb des Grenzwerts für den Parameter **appl_memory** liegt. Wenn der Grenzwert von **appl_memory** oder der Grenzwert von **instance_memory** erreicht wird, schlägt die Anwendungsanforderung fehl, durch die der Datenbankmanager den Grenzwert erreicht hat. Die Anforderung gibt einen entsprechenden SQL-Code zurück (der tatsächlich zurückgegebene Fehlercode hängt davon ab, wo genau in der Operation der Anwendung der Fehler aufgrund von „Speicherknappheit” aufgetreten ist). Wenn ein Fehler aufgrund von „Speicherknappheit” aufgetreten ist, kann der Datenbankadministrator die Datei db2diag.log anzeigen, um festzustellen, wie viel Speicher belegt wurde, als der Fehler auftrat. Dies kann bei der Feststellung hilfreich sein, ob ein Speicherparameter angepasst werden muss.

Aktivieren des Speichers mit automatischer Leistungsoptimierung

Der Speicher mit automatischer Leistungsoptimierung vereinfacht die Speicherkonfiguration, indem automatisch Werte für Speicherkonfigurationsparameter eingestellt und die Größe von Pufferpools gesteuert werden. Wenn sie aktiviert ist, verteilt die Speicheroptimierungsfunktion verfügbare Speicherressourcen unter mehreren Speicherkonsumenten, zu denen Bereiche für Sortierspeicher, Paketcache, Sperrenliste und Pufferpools zählen.

1. Die automatische Speicheroptimierungsfunktion für die Datenbank wird dadurch aktiviert, dass der Parameter **self_tuning_mem** auf den Wert ON gesetzt wird. Sie können den Parameter **self_tuning_mem** mithilfe des Befehls UPDATE DATABASE CONFIGURATION, mit der API sqlfupd oder über das Fenster 'Datenbankkonfigurationsparameter ändern' der Steuerzentrale auf ON setzen.
2. Zur Aktivierung der automatischen Optimierungsfunktion für Speicherbereiche, die durch Speicherkonfigurationsparameter gesteuert werden, setzen Sie die relevanten Konfigurationsparameter mithilfe des Befehls UPDATE DATABASE CONFIGURATION, mit der API sqlfupd oder über das Fenster Datenbankkonfigurationsparameter ändern der Steuerzentrale auf den Wert AUTOMATIC.
3. Zur Aktivierung der automatischen Optimierung für Pufferpools setzen Sie die Größe (SIZE) des Pufferpools auf den Wert AUTOMATIC. Dies können Sie mithilfe der Anweisung ALTER BUFFERPOOL für vorhandene Pufferpools und der Anweisung CREATE BUFFERPOOL für neue Pufferpools erreichen. Wenn die Größe eines Pufferpools in einer Umgebung mit partitionierten Datenbanken auf AUTOMATIC gesetzt wird, darf dieser Pufferpool keine Einträge in SYSIBM.SYSBUFFERPOOLNODES haben.

Anmerkung:

1. Da die automatische Optimierungsfunktion Speicherressourcen unter verschiedenen Speicherbereichen umverteilt, müssen mindestens zwei Speicherkonsumenten, zum Beispiel der Sperrenspeicherbereich und der gemeinsam genutzte Datenbankspeicher, für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann. Die einzige Ausnahme in dieser Hinsicht ist der Speicher, der durch den Konfigurationsparameter **sortheap** gesteuert wird. Wenn **sortheap** allein auf AUTOMATIC gesetzt ist, ist die automatische Optimierung von **sortheap** aktiviert.
2. Zur Aktivierung des Konfigurationsparameters **locklist** für die automatische Optimierung muss auch der Parameter **maxlocks** für die automatische Optimierung aktiviert werden. Daher wird **maxlocks** auf AUTOMATIC gesetzt, wenn **locklist** auf AUTOMATIC gesetzt wird. Zur Aktivierung des Konfigurationsparameters **sheapthres_shr** für die automatische Optimierung muss auch der Parameter **sortheap** für die automatische Optimierung aktiviert werden. Daher wird **sortheap** auf AUTOMATIC gesetzt, wenn **sheapthres_shr** auf AUTOMATIC gesetzt wird.
3. Die automatische Optimierung von **sheapthres_shr** oder **sortheap** ist nur zulässig, wenn der Datenbankkonfigurationsparameter **sheapthres** auf den Wert 0 gesetzt ist.
4. Speicher mit automatischer Leistungsoptimierung wird nur auf dem primären HADR-Server verwendet. Wenn die Funktion für die automatische Speicheroptimierung auf einem HADR-System aktiviert wird, wird sie nie auf dem sekundären Server verwendet, und auch nur dann auf dem primären Server, wenn die Konfiguration ordnungsgemäß eingestellt ist. Wenn ein Befehl ausgeführt wird, der die Rollen der HADR-Datenbanken vertauscht, wird die Nutzung der Funktion für die automatische Speicheroptimierung ebenfalls übertra-

gen, sodass sie auf dem neuen primären Server erfolgt. Nach dem Starten der Primärdatenbank oder dem Konvertieren einer Bereitschaftsdatenbank in eine Primärdatenbank durch Übernahme wird die STMM-EDU (Self Tuning Memory Manager Engine-Dispatchable-Unit) möglicherweise erst bei der ersten ankommenden Clientverbindungsanforderung gestartet.

Inaktivieren des Speichers mit automatischer Leistungsoptimierung

Die automatische Leistungsoptimierung kann für die gesamte Datenbank inaktiviert werden, indem der Parameter **self_tuning_mem** auf den Wert OFF gesetzt wird. Wenn der Parameter **self_tuning_mem** auf den Wert OFF gesetzt wird, behalten die Parameter für die Speicherkonfiguration und die Pufferpools, die den Wert AUTOMATIC haben, den Wert AUTOMATIC, und die Speicherbereiche behalten ihre aktuelle Größe bei.

Sie können den Parameter **self_tuning_mem** mithilfe des Befehls UPDATE DATABASE CONFIGURATION, mit der API sqlfupd oder über das Fenster 'Datenbankkonfigurationsparameter ändern' der Steuerzentrale auf OFF setzen.

Die automatische Leistungsoptimierung kann für die gesamte Datenbank effektiv außerdem dadurch inaktiviert werden, dass nur ein einziger Speicherkonsument für die automatische Leistungsoptimierung aktiviert wird. Dies liegt daran, dass kein Speicher umverteilt werden kann, wenn nur ein Hauptspeicherbereich aktiviert ist.

Zur Inaktivierung der automatischen Optimierung des Konfigurationsparameters **sortheap** könnten Sie zum Beispiel den folgenden Befehl eingeben:

```
UPDATE DATABASE CONFIGURATION USING SORTHEAP MANUAL
```

Zur Inaktivierung der automatischen Optimierung des Konfigurationsparameters **sortheap** und gleichzeitigen Änderung des aktuellen Werts von **sortheap** in den Wert 2000 geben Sie den folgenden Befehl ein:

```
UPDATE DATABASE CONFIGURATION USING SORTHEAP 2000
```

In einigen Fällen kann ein Speicherkonfigurationsparameter für die automatische Optimierungsfunktion nur dann aktiviert werden, wenn ein anderer Speicherkonfigurationsparameter ebenfalls aktiviert wird. Zum Beispiel kann die automatische Optimierung für den Konfigurationsparameter **maxlocks** nur aktiviert werden, wenn der Konfigurationsparameter **locklist** ebenfalls für diese Funktion aktiviert wird. Desgleichen kann die automatische Optimierungsfunktion für den Konfigurationsparameter **sheapthres_shr** nur aktiviert werden, wenn sie auch für den Konfigurationsparameter **sortheap** aktiviert wird. Dies bedeutet, dass eine Inaktivierung der automatischen Optimierung des Parameters **locklist** oder **sortheap** auch die automatische Optimierung für den Parameter **maxlocks** bzw. **sheapthres_shr** inaktiviert.

Die automatische Optimierung kann für einen Pufferpool inaktiviert werden, indem der Pufferpool auf eine bestimmte Größe gesetzt wird. Zum Beispiel inaktiviert die folgende Anweisung die automatische Optimierung für den Pufferpool `bufferpool1`:

```
ALTER BUFFERPOOL bufferpool1 SIZE 1000
```

Ermitteln der Speicherkonsumenten mit aktivierter automatischer Leistungsoptimierung

Zum Anzeigen der Einstellungen für die automatische Optimierungsfunktion für Speicherkonsumenten, die durch Konfigurationsparameter gesteuert werden, können Sie eine der folgenden Methoden verwenden.

- Zum Anzeigen der Einstellungen der automatischen Optimierungsfunktion für Konfigurationsparameter über die Befehlszeile verwenden Sie den Befehl GET DATABASE CONFIGURATION mit dem Parameter SHOW DETAIL.

  Die Speicherkonsumenten, die für die automatische Optimierung aktiviert werden können, werden in der Ausgabe wie folgt zusammengruppiert:

```
Beschreibung                                                    Parameter    Aktueller Wert     Verzögerter Wert
-------------------------------------------------------------------------------------------------------------------
Speicher mit automatischer Leistungsoptimierung          (SELF_TUNING_MEM) = ON (Aktiv)         ON
Größe des gemeinsamen Datenbankspeichers (4 KB)          (DATABASE_MEMORY) = AUTOMATIC(37200)   AUTOMATIC(37200)
Max. Speicher für Sperrenliste (4 KB)                           (LOCKLIST) = AUTOMATIC(7456)    AUTOMATIC(7456)
Anzahl der Sperrenlisten pro Anwend. (in %)                     (MAXLOCKS) = AUTOMATIC(98)      AUTOMATIC(98)
Größe des Paketcache (4 KB)                                  (PCKCACHESZ) = AUTOMATIC(5600)    AUTOMATIC(5600)
Sortierspeicherschwelle für gemeinsame Sortierungen (4 KB)(SHEAPTHRES_SHR) = AUTOMATIC(5000)    AUTOMATIC(5000)
Zwischenspeicher für Sortierlisten (4 KB)                       (SORTHEAP) = AUTOMATIC(256)     AUTOMATIC(256)
```

- Sie können auch die API 'db2CfgGet' verwenden, um zu ermitteln, ob die Optimierung aktiviert ist oder nicht. Die folgenden Werte werden zurückgegeben:

```
SQLF_OFF                  0
SQLF_ON_ACTIVE            2
SQLF_ON_INACTIVE          3
```

  Der Wert SQLF_ON_ACTIVE beschreibt eine Situation, in der die automatische Optimierung aktiviert und aktiv ist, während der Wert SQLF_ON_INACTIVE anzeigt, dass die automatische Optimierung zwar aktiviert, jedoch zurzeit nicht aktiv ist.

- Sie können die Konfigurationseinstellungen auch im Fenster **Datenbankkonfiguration** der Steuerzentrale prüfen.

Zum Anzeigen der Einstellungen für die automatische Optimierungsfunktion für Pufferpools können Sie eine der folgenden Methoden verwenden.

- Zum Abrufen einer Liste von Pufferpools, für die die automatische Optimierung aktiviert ist, geben Sie den folgenden Befehl in die Befehlszeile ein:

```
db2 "select BPNAME, NPAGES from sysibm.sysbufferpools"
```

  Wenn die automatische Optimierung für einen Pufferpool aktiviert ist, ist das Feld NPAGES in der Tabelle SYSIBM.SYSBUFFERPOOLS für den betreffenden Pufferpool auf den Wert -2 gesetzt. Wenn die automatische Optimierung inaktiviert ist, enthält das Feld NPAGES die aktuelle Größe des Pufferpools.

- Zur Bestimmung der aktuellen Größe von Pufferpools, die für die automatische Optimierung aktiviert wurden, verwenden Sie den Snapshot Monitor wie folgt und prüfen die aktuelle Größe des Pufferpools (den Wert des Monitorelements 'bp_cur_buffsz'):

```
db2 get snapshot for bufferpools on db_name
```

- Zum Anzeigen der Einstellungen für die automatische Optimierung Ihrer Pufferpools über die Steuerzentrale klicken Sie einen Pufferpool mit der rechten Maustaste an und prüfen die Attribute der Pufferpools im Teilfenster mit den Objektdetails.

Es ist wichtig zu beachten, dass die Reaktionsfähigkeit der Speicheroptimierungsfunktion durch die Zeit eingeschränkt wird, die zur Änderung der Größe eines Speicherkonsumenten erforderlich ist. Zum Beispiel kann die Verringerung der

Größe eines Pufferpools ein längerer Prozess sein, weshalb die Vorteile der Verkleinerung des Pufferpoolspeichers zu Gunsten einer Erweiterung des Sortierspeichers vielleicht nicht sofort realisiert werden.

Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken

Wenn die automatische Speicheroptimierungsfunktion in Umgebungen mit partitionierten Datenbanken verwendet wird, bestimmen einige wenige Faktoren, ob die Funktion das System geeignet optimiert.

Wenn der Speicher mit automatischer Leistungsoptimierung in partitionierten Datenbanken aktiviert wird, wird eine Datenbankpartition als Optimierungspartition bestimmt. Alle Entscheidungen bezüglich der Speicheroptimierung werden auf der Basis der Speicher- und Auslastungsmerkmale dieser Datenbankpartition getroffen. Wenn die Optimierungsentscheidungen in der Optimierungspartition getroffen sind, werden die Speicheranpassungen an alle anderen Datenbankpartitionen verteilt, um sicherzustellen, dass alle Datenbankpartitionen ähnliche Konfigurationen behalten.

Das auf einer Optimierungspartition basierende Modell setzt voraus, dass die Funktion nur in Datenbankpartitionen mit ähnlichen Speicheranforderungen verwendet wird. Die folgenden Richtlinien sind bei der Entscheidung zu beachten, ob die automatische Speicheroptimierung für eine partitionierte Datenbank aktiviert werden sollte.

Fälle, in denen die automatische Optimierung in partitionierten Datenbanken empfohlen wird

Wenn alle Datenbankpartitionen ähnliche Speicheranforderungen haben und auf ähnlicher Hardware betrieben werden, kann die automatische Speicheroptimierung ohne Modifikationen aktiviert werden. Solche Typen von Umgebungen haben die folgenden gemeinsamen Merkmale:

- Alle Datenbankpartitionen befinden sich auf identischer Hardware, und mehrere logische Knoten sind gleichmäßig auf mehrere physische Knoten verteilt.
- Sie weisen eine perfekte oder nahezu perfekte Verteilung der Daten auf.
- Die Auslastung, die in den Datenbankpartitionen verarbeitet wird, wird gleichmäßig auf die Datenbankpartitionen verteilt. Dies bedeutet, dass keine einzelne Datenbankpartition erhöhte Speicheranforderungen für einen oder mehrere Zwischenspeicher hat.

In einer solchen Umgebung ist für alle Datenbankpartitionen die gleiche Konfiguration wünschenswert, und die automatische Speicheroptimierung kann das System geeignet konfigurieren.

Fälle, in denen die automatische Optimierung in partitionierten Datenbanken nur mit Vorsicht empfohlen wird

In Fällen, in denen die meisten Datenbankpartitionen in einer Umgebung ähnliche Speicheranforderungen haben und auf ähnlicher Hardware betrieben werden, kann die automatische Speicheroptimierung eingesetzt werden, solange die Anfangskonfiguration mit Vorsicht erfolgt. Solche Systeme verfügen möglicherweise über eine Gruppe von Datenbankpartitionen für Daten und über eine wesentlich kleinere Gruppe von Koordinatorpartitionen und Katalogpartitionen. In solchen Umgebungen kann es von Vorteil sein, die Koordinatorpartitionen und Katalogpartitionen anders zu konfigurieren als die Datenbankpartitionen mit den Daten.

Auch eine solche Umgebung kann von der automatischen Speicheroptimierungsfunktion profitieren, wenn einige kleinere Konfigurationsschritte ausgeführt werden. Da die Datenbankpartitionen mit den Daten den Hauptteil der Datenbankpartitionen ausmachen, sollte die automatische Optimierung für alle diese Datenbankpartitionen aktiviert und eine dieser Datenbankpartitionen als Optimierungspartition angegeben werden. Außerdem sollte die automatische Speicheroptimierung für die Katalog- und Koordinatorpartitionen inaktiviert werden, da sie möglicherweise eine andere Konfiguration aufweisen. Zur Inaktivierung der automatischen Optimierung in den Katalog- und Koordinatorpartitionen aktualisieren Sie den Datenbankkonfigurationsparameter *self_tuning_mem* in diesen Partitionen mit dem Wert OFF.

Fälle, in denen die automatische Optimierung in partitionierten Datenbanken nicht empfohlen wird

In Umgebungen, in denen sich die Speicheranforderungen jeder Datenbankpartition unterscheiden oder verschiedene Datenbankpartitionen auf völlig anderer Hardware betrieben werden, ist es empfehlenswert, die Funktion der automatischen Speicheroptimierung zu inaktivieren. Sie können dies erreichen, indem Sie in allen Partitionen den Datenbankkonfigurationsparameter *self_tuning_mem* auf den Wert OFF setzen.

Vergleich der Speicheranforderungen verschiedener Datenbankpartitionen

Die beste Methode zur Bestimmung, ob die Speicheranforderungen verschiedener Datenbankpartitionen ausreichend ähnlich sind, ist die Verwendung des Snapshot Monitors. Wenn die folgenden Monitorelemente in allen Partitionen ähnliche Werte liefern (mit Abweichungen von unter 20 %), können die Partitionen als ähnlich betrachtet werden.

Erfassen Sie die folgenden Daten, indem Sie den Befehl `get snapshot for database on <dbname>` ausführen.

```
Gesamter zugeordneter gemeinsamer Sortierspeicher                               = 0
Obere Grenze für gemeinsamen Sortierspeicher                                    = 0
Sortiervorgang nach Schwellenwertüberschreitung (gemeinsam genutzter Speicher) = 0
Überläufe bei Sortierung                                                        = 0

Suchoperationen im Paket-Cache                                                  = 13
Einfügungen im Paket-Cache                                                      = 1
Überläufe des Paket-Caches                                                      = 0
Obere Grenze für Paket-Cache (Byte)                                             = 655360

Anzahl Hash-Joins                                                               = 0
Anzahl Hash-Schleifen                                                           = 0
Anzahl Überläufe von Hash-Joins                                                 = 0
Anzahl kleiner Überläufe von Hash-Joins                                         = 0
Hash-Joins nach Schwellenwertüberschreitung (gemeinsam genutzter Speicher)      = 0

Aktuelle Sperren                                                                = 0
Warten bei Sperren                                                              = 0
Wartezeit der Datenbank bei Sperren (ms)                                        = 0
Verwendeter Speicher für Sperrenlisten (Byte)                                   = 4968
Sperreneskalationen                                                             = 0
Exklusive Sperreneskalationen                                                   = 0
```

Erfassen Sie die folgenden Daten, indem Sie den Befehl `get snapshot for bufferpools on <dbname>` ausführen:

```
Logische Lesevorgänge im Pufferpool                                    = 0
Physische Lesevorgänge im Pufferpool                                   = 0
Logische Lesevorgänge im Pufferpoolindex                               = 0
Physische Lesevorgänge im Pufferpoolindex                              = 0
Gesamtzeit der Lesevorgänge im Pufferpool  (ms)                        = 0
Gesamtzeit der Schreibvorgänge im Pufferpool (ms)                      = 0
```

Verwenden von Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken

Wenn die Funktion für die automatische Optimierung in Umgebungen mit partitionierten Datenbanken aktiviert wird, überwacht eine einzige Datenbankpartition, die als *Optimierungspartition* bezeichnet wird, die Speicherkonfiguration und gibt alle Konfigurationsänderungen an alle anderen Datenbankpartitionen weiter, um eine konsistente Konfiguration über alle beteiligten Datenbankpartitionen hinweg sicherzustellen.

Die Optimierungspartition wird nach einer Reihe von Merkmalen ausgewählt, wie zum Beispiel der Anzahl von Datenbankpartitionen in der Partitionsgruppe und der Anzahl der definierten Pufferpools.

- Wenn Sie ermitteln möchten, welche Datenbankpartition zurzeit als Optimierungspartition angegeben ist, verwenden Sie die Prozedur ADMIN_CMD wie folgt:

  ```
  CALL SYSPROC.ADMIN_CMD( 'get stmm tuning dbpartitionnum' )
  ```

- Zum Ändern der Optimierungspartition verwenden Sie die Prozedur ADMIN_CMD wie folgt:

  ```
  CALL SYSPROC.ADMIN_CMD( 'update stmm tuning dbpartitionnum <db_partitionsnummer>')
  ```

  Wenn Sie diesen Befehl absetzen, wird die Optimierungspartition asynchron oder beim nächsten Starten der Datenbank aktualisiert.

- Wenn die Optimierungspartition von der Speicheroptimierungsfunktion automatisch neu ausgewählt werden soll, geben Sie für <db_partitionsnummer> den Wert -1 ein.

Starten der Speicheroptimierungsfunktion auf DPF-Systemen

In einer DPF-Umgebung wird die Speicheroptimierungsfunktion nur gestartet, wenn die Datenbank explizit mit dem Befehl ACTIVATE DATABASE aktiviert wird, weil die automatische Optimierung voraussetzt, dass alle Partitionen aktiv sind, bevor sie den Speicher in einem Mehrpartitionssystem richtig optimieren kann.

Inaktivieren der automatischen Optimierung für eine bestimmte Datenbankpartition

- Zur Inaktivierung der automatischen Optimierung für eine Teilgruppe von Datenbankpartitionen setzen Sie für die Datenbankpartitionen, die nicht optimiert werden sollen, den Konfigurationsparameter *self_tuning_mem* auf den Wert OFF.
-

  Zur Inaktivierung der automatischen Optimierung für eine Teilgruppe von Speicherkonsumenten, die durch Konfigurationsparameter gesteuert werden, in einer bestimmten Datenbankpartition, setzen Sie den Wert des relevanten Konfigurationsparameters oder die Pufferpoolgröße auf MANUAL bzw. einen bestimmten Wert in dieser Datenbankpartition. Es wird jedoch empfohlen, die

Werte der Konfigurationsparameter für die automatische Optimierungsfunktion über alle aktiven Partitionen hinweg einheitlich zu definieren.

- Wenn Sie die Optimierung für einen bestimmten Pufferpool in einer Datenbankpartition inaktivieren, führen Sie den Befehl ALTER BUFFERPOOL mit der Angabe eines Größenwerts und eines Werts für den Parameter PARTITIONNUM für die Partition aus, in der die automatische Optimierung inaktiviert werden soll.

  Eine Anweisung ALTER BUFFERPOOL, die die Größe mithilfe der Klausel PARTITIONNUM in einer bestimmten Datenbankpartition angibt, erstellt einen Ausnahmeeintrag für den angegebenen Pufferpool im Katalog SYSCAT.SYSBUFFERPOOLNODES oder aktualisiert den Ausnahmeeintrag, wenn bereits einer vorhanden ist. Wenn ein Ausnahmeeintrag für einen Pufferpool in diesem Katalog vorhanden ist, ist dieser Pufferpool an der automatischen Optimierung nicht beteiligt, wenn die Standardpufferpoolgröße auf den Wert AUTOMATIC gesetzt ist. Gehen Sie daher wie folgt vor, wenn Sie einen Ausnahmeeintrag entfernen möchten, sodass ein Pufferpool für die automatische Optimierung wieder aktiviert werden kann:

  1. Inaktivieren Sie die automatische Optimierung für diesen Pufferpool, indem Sie eine Anweisung ALTER BUFFERPOOL ausführen, die die Pufferpoolgröße auf einen bestimmten Wert setzt.
  2. Führen Sie eine weitere Anweisung ALTER BUFFERPOOL unter Angabe der Klausel PARTITIONNUM aus, um den Pufferpool in dieser Datenbankpartition auf die Standardpufferpoolgröße zu setzen.
  3. Aktivieren Sie die automatische Optimierung, indem Sie eine weitere Anweisung ALTER BUFFERPOOL ausführen, in der die Größe (SIZE) auf den Wert AUTOMATIC gesetzt ist.

Aktivieren des Speichers mit automatischer Leistungsoptimierung in nicht einheitlichen Umgebungen

Im Idealfall sollten Ihre Daten gleichmäßig auf alle Datenbankpartitionen verteilt und die Auslastung in jeder Partition durch ähnliche Speicheranforderungen gekennzeichnet sein. Wenn die Datenverteilung ungleichmäßig ist, sodass mindestens eine Datenbankpartition erheblich mehr oder weniger Daten als andere Datenbankpartitionen enthält, sollten solche anomalen Datenbankpartitionen nicht für die automatische Optimierung aktiviert werden. Dasselbe gilt, wenn die Speicheranforderungen in den Datenbankpartitionen unterschiedlich sind. Dies kann geschehen, wenn zum Beispiel ressourcenintensive Sortiervorgänge nur in einer Partition ausgeführt werden oder wenn einige Datenbankpartitionen mit anderer Hardware und mehr verfügbarem Speicher als andere Partitionen ausgestattet sind. Die automatische Optimierung kann dennoch in einigen Datenbankpartitionen in diesem Typ von Umgebung aktiviert werden. Zur Nutzung der Vorteile der automatischen Speicheroptimierung in Umgebungen mit ungleich verteilten Anforderungen, geben Sie eine Gruppe von Datenbankpartitionen an, die ähnliche Daten- und Speicheranforderungen haben, und aktivieren sie für die automatisch Optimierung. Die Speicherkonfiguration in den übrigen Partitionen muss in diesem Fall manuell erfolgen.

Konfigurieren von Speicher und der Zwischenspeicher

Mit der Funktion zur vereinfachten Speicherkonfiguration können Sie Speicher und Zwischenspeicher konfigurieren, die für den DB2-Datenserver erforderlich sind, indem Sie die Standardeinstellung AUTOMATIC für die meisten speicherbezogenen Konfigurationsparameter verwenden. Dies ermöglicht gleichzeitig eine erhebliche Verringerung des Optimierungsaufwands.

Die vereinfachte Speicherkonfigurationsfunktion hat die folgenden Vorteile:

- Sie können einen einzigen Parameter (**instance_memory**) verwenden, um den gesamten Speicher anzugeben, den der Datenbankmanager aus seinen privaten und gemeinsam genutzten Zwischenspeichern zuordnen darf. Darüber hinaus können Sie auch den Konfigurationsparameter **appl_memory** verwenden, um die maximale Menge an Anwendungsspeicher zu steuern, die Serviceanwendungsanforderungen durch DB2-Datenbankagenten zugeordnet wird.
- Sie brauchen Parameter, die nur für den funktionalen Speicher verwendet werden, nicht manuell zu optimieren.
- Sie können abfragen, wie viel Gesamtspeicher momentan von privaten und gemeinsam genutzten Zwischenspeichern des Datenbankmanagers belegt wird, indem Sie Memory Visualizer verwenden. Sie können auch den Befehl db2mtrk zum Überwachen der Zwischenspeicherbelegung und die Tabellenfunktion ADMIN_GET_DBP_MEM_USAGE() zum Abfragen der Gesamtspeicherbelegung verwenden.
- Für die DB2-Standardkonfiguration ist viel weniger Optimierungsaufwand erforderlich. Dies ist ein Vorteil für neue Instanzen, die Sie erstellen.

In der folgenden Tabelle werden die Speicherkonfigurationsparameter aufgeführt, die standardmäßig die Einstellung AUTOMATIC haben. Diese Parameter können bei Bedarf auch dynamisch konfiguriert werden. Beachten Sie, dass sich die Bedeutung der Einstellung AUTOMATIC für die einzelnen Parameter unterscheidet. Dies wird in der ganz rechten Spalte beschrieben.

Tabelle 1. Speicherkonfigurationsparameter mit dem Standardwert AUTOMATIC

Name des Konfigurationsparameters	Beschreibung	Bedeutung der Einstellung AUTOMATIC
appl_memory	Steuert die maximale Menge an Anwendungsspeicher, die Serviceanwendungsanforderungen von DB2-Datenbankagenten zugeordnet wird.	Die Einstellung AUTOMATIC lässt alle Anwendungsspeicheranforderungen zu, solange sich die gesamte von der Datenbankpartition zugeordnete Speicherkapazität innerhalb der Grenzwerte des Parameters **instance_memory** bewegt.

Tabelle 1. Speicherkonfigurationsparameter mit dem Standardwert AUTOMATIC (Forts.)

Name des Konfigurationsparameters	Beschreibung	Bedeutung der Einstellung AUTOMATIC
applheapsz	Vor Version 9.5 bezog sich dieser Parameter auf die Größe des Anwendungsspeichers, die jeder Datenbankagent, der für eine Anwendung arbeitet, in Anspruch nehmen konnte. In Version 9.5 bezieht sich dieser Parameter auf die gesamte Anwendungsspeichergröße, die von der gesamten Anwendung verwendet werden kann. Für DPF-, Konzentrator- oder SMP-Konfigurationen bedeutet dies, dass der in früheren Releases benutzte Wert für den Parameter **applheapsz** eventuell erhöht werden muss, sofern nicht die Einstellung AUTOMATIC verwendet werden.	Mit der Einstellung AUTOMATIC kann die Größe des Anwendungszwischenspeichers nach Bedarf anwachsen, bis der Grenzwert entweder des Parameters **appl_memory** oder des Parameters **instance_memory** erreicht ist.
database_memory (Vor Version 9.5 galt die Standardeinstellung AUTOMATIC nur für Windows- und AIX-Plattformen. Mit Version 9.5 gilt die Standardeinstellung AUTOMATIC für alle DB2-Serverprodukte.)	Gibt die Menge des gemeinsam genutzten Speichers an, die für den gemeinsam genutzten Speicherbereich einer Datenbank reserviert ist.	Wenn sie aktiviert ist, bestimmt die Speicheroptimierungsfunktion den Gesamtspeicherbedarf für die Datenbank und erhöht oder verringert auf der Basis des aktuellen Bedarfs der Datenbank die Speichergröße, die für den gemeinsam genutzten Datenbankspeicher zugeordnet ist.
dbheap	Legt die maximale Speicherkapazität fest, die vom Zwischenspeicher für die Datenbank verwendet wird.	Mit der Einstellung AUTOMATIC kann die Größe des Datenbankzwischenspeichers nach Bedarf anwachsen, bis der Grenzwert entweder des Parameters **database_memory** oder des Parameters **instance_memory** erreicht ist.
instance_memory	Gibt die maximale Speichermenge an, die für eine Datenbankpartition zugeordnet werden kann.	Durch die Einstellung AUTOMATIC kann der von der gesamten Datenbankmanagerinstanz belegte Gesamtspeicher bis zu einem Grenzwert von 75 - 95 % des physischen Arbeitsspeichers (RAM) des Systems anwachsen. Dieser Grenzwert wird während der Verarbeitung des Befehls 'db2start' berechnet.

Tabelle 1. Speicherkonfigurationsparameter mit dem Standardwert AUTOMATIC (Forts.)

Name des Konfigurations-parameters	Beschreibung	Bedeutung der Einstellung AUTOMATIC
mon_heap_sz	Legt den Speicherbereich in Seiten fest, der für Daten des Datenbanksystemmonitors zugeordnet werden soll.	Mit der Einstellung AUTOMATIC kann die Größe des Monitorzwischen-speichers nach Bedarf anwachsen, bis der Grenz-wert des Parameters **instance_memory** erreicht ist.
stat_heap_sz	Gibt die maximale Größe des Zwischenspeichers an, der bei der Erfassung statisti-scher Daten mit dem Befehl RUNSTATS verwendet wird.	Mit der Einstellung AUTOMATIC kann die Größe des Statistikzwischen-speichers nach Bedarf anwachsen, bis der Grenz-wert entweder des Parame-ters **appl_memory** oder des Parameters **instance_memory** erreicht ist.
stmtheap	Legt die Größe des Anweisungszwischen-speichers fest, der als Arbeitsbereich für den SQL- oder XQuery-Compiler zur Kompilierung einer SQL- oder XQuery-Anweisung verwendet wird.	Mit der Einstellung AUTOMATIC kann die Größe des Anweisungs-zwischenspeichers nach Bedarf anwachsen, bis der Grenzwert entweder des Parameters **appl_memory** oder des Parameters **instance_memory** erreicht ist.

Anmerkung: Die Verwaltungssichten DBMCFG und DBCFG rufen Informationen zu Konfigurationsparametern des Datenbankmanagers für die momentan verbundene Datenbank für alle Datenbankpartitionen ab. Für die Konfigurationsparameter **mon_heap_sz**, **stmtheap** und **stat_heap_sz** ist die Spalte DEFERRED_VALUE in diesen Sichten über Datenbankaktivierungen hinweg nicht persistent. Das heißt, dass die Ausgabe der Abfrage, wenn Sie den Befehl get dbm cfg show detail oder get db cfg show detail absetzen, aktualisierte Werte (im Speicher) anzeigt.

Der folgenden Tabelle ist zu entnehmen, ob Konfigurationsparameter bei der Migration oder Erstellung einer Instanz und bei der Migration oder Erstellung einer Datenbank auf den Standardwert AUTOMATIC gesetzt werden.

Tabelle 2. Konfigurationsparameter, die bei der Migration und Erstellung einer Instanz oder Datenbank auf AUTOMATIC gesetzt werden

Konfigurations-parameter	Bei Migration oder Erstellung einer Instanz auf AUTOMATIC gesetzt	Bei Migration einer Datenbank auf AUTOMATIC gesetzt	Bei Erstellung einer Datenbank auf AUTOMATIC gesetzt
applheapsz[1]		X	X
dbheap		X	X
instance_memory	X		

Tabelle 2. Konfigurationsparameter, die bei der Migration und Erstellung einer Instanz oder Datenbank auf AUTOMATIC gesetzt werden (Forts.)

Konfigurations-parameter	Bei Migration oder Erstellung einer Instanz auf AUTOMATIC gesetzt	Bei Migration einer Datenbank auf AUTOMATIC gesetzt	Bei Erstellung einer Datenbank auf AUTOMATIC gesetzt
mon_heap_sz[1]	X		
stat_heap_sz[1]		X	X
stmtheap[1]			X

Die folgenden Elemente werden für die vereinfachte Speicherkonfiguration nicht weiter unterstützt:

- Die Konfigurationsparameter **appgroup_mem_sz**, **groupheap_ratio** und **app_ctl-_heap_sz**. Diese Konfigurationsparameter wurden durch den neuen Konfigurationsparameter **appl_memory** ersetzt.
- Der Parameter **-p** des Speichertrackerbefehls db2mtrk. Diese Option, mit der private Agentenzwischenspeicher aufgelistet werden, wurde durch den Parameter **-a** ersetzt, mit dem die gesamte Anwendungsspeicherbelegung aufgeführt wird.

Memory Visualizer zeigt die maximale Anwendungsspeicherbelegung durch eine Datenbank mit dem neuen Konfigurationsparameter **appl_memory** sowie die maximale Speicherbelegung durch eine Instanz mit dem aktualisierten Konfigurationsparameter **instance_memory** an. Memory Visualizer zeigt auch die Werte für alle Konfigurationsparameter an, die die Einstellung AUTOMATIC zulassen. Werte für die veralteten Konfigurationsparameter werden in Memory Visualizer für Datenbanken der Version 9.5 nicht angezeigt. Sie werden jedoch für frühere Versionen der Datenbanken angezeigt.

Versuche, den Parameter **instance_memory** auf Werte zu aktualisieren, die größer als die in der folgenden Liste angegebenen sind, schlagen mit dem Rückkehrcode SQL5130N fehl:

- 4 GB (1.048.576 * 4-KB-Seiten) für DB2 Express Edition und DB2 Express-C
- 16 GB (4.194.304 * 4-KB-Seiten) für DB2 Workgroup Server Edition

Wenn der gemeinsam genutzte FCM-Speicher (FCM - Fast Communication Manager) zugeordnet wird, wird der Anteil der einzelnen lokalen Datenbankpartitionen an der gesamten Größe des gemeinsam genutzten FCM-Speichers für das System im Grenzwert **instance_memory** der jeweiligen Datenbankpartition berücksichtigt. Aufgrund der Spezifik von FCM-Speicher (fehlschlagende Zuordnung von FCM-Puffern kann die Instanz außer Gefecht setzen) schlagen FCM-Speicheranforderungen nie wegen des Grenzwerts in **instance_memory** fehl. Sie können jedoch fehlschlagen, wenn aus dem Betriebssystem kein Speicher zugeordnet werden kann. Wenn eine FCM-Speicheranforderung dazu führt, dass eine Datenbankpartition ihren Grenzwert in **instance_memory** überschreitet, schlagen andere Speicheranforderungen so lange fehl, bis die Speicherbelegung der Partition wieder einen Stand unterhalb des Grenzwerts von **instance_memory** erreicht.

Konfiguration des Agenten und des Prozessmodells

Version 9.5 stellt einen auf Vereinfachung und größere Flexibilität ausgerichteten Mechanismus zur Konfiguration der Prozessmodellparameter zur Verfügung. Dank dieser vereinfachten Konfiguration brauchen diese Parameter nicht regelmäßig

angepasst zu werden, sodass sich der Zeit- und Arbeitsaufwand für ihre Konfiguration verringert. Darüber hinaus müssen DB2-Instanzen auch nicht mehr beendet und erneut gestartet werden, um die neuen Werte in Kraft zu setzen.

Für die dynamische und automatische Agenten- und Speicherkonfiguration sind geringfügig mehr Speicherressourcen erforderlich, wenn eine Instanz aktiviert wird.

Konfiguration von Agenten, des Prozessmodells und des Speichers

DB2-Datenserver nutzen eine Multithreadarchitektur auf 32- und 64-Bit-Plattformen, um eine Reihe von Vorteilen, wie zum Beispiel besseren Bedienungskomfort, bessere gemeinsame Ressourcennutzung, geringeren Speicherbedarf und eine konsistente Threading-Architektur über alle Betriebssysteme hinweg zu realisieren.

Konfigurieren von Datenbanken über mehrere Partitionen

Der Datenbankmanager stellt in einer Sicht alle Datenbankkonfigurationselemente über mehrere Partitionen hinweg dar. Dies bedeutet, dass Sie eine Datenbankkonfiguration über alle Datenbankpartitionen hinweg aktualisieren oder zurücksetzen können, ohne den Befehl db2_all für jede einzelne Datenbankpartition aufrufen zu müssen.

Sie können eine Datenbankkonfiguration über Partitionen hinweg aktualisieren, indem Sie nur eine SQL-Anweisung bzw. einen Verwaltungsbefehl von einer beliebigen Partition aus ausführen, in der sich die Datenbank befindet. Die Methode zum Aktualisieren und Zurücksetzen einer Datenbankkonfiguration gilt standardmäßig *für alle Datenbankpartitionen*.

Aus Gründen der Abwärtskompatibilität von Befehlsscripts und Anwendungen sind drei Optionen verfügbar:

- Sie können den Befehl db2set wie folgt verwenden, um die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** auf den Wert TRUE zu setzen:

  ```
  DB2_UPDDBCFG_SINGLE_DBPARTITION=TRUE
  ```

  Anmerkung: Die Einstellung der Registrierdatenbankvariablen hat keine Relevanz für Anforderungen mit dem Befehl UPDATE DATABASE CONFIGURATION oder RESET DATABASE CONFIGURATION, die Sie mithilfe der Prozedur ADMIN_CMD ausführen.
- Sie können den Parameter **DBPARTITIONNUM** entweder mit dem Befehl UPDATE DATABASE CONFIGURATION oder mit dem Befehl RESET DATABASE CONFIGURATION oder mit der Prozedur ADMIN_CMD verwenden. Wenn Sie zum Beispiel die Datenbankkonfigurationen in allen Datenbankpartitionen aktualisieren möchten, rufen Sie die Prozedur ADMIN_CMD wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG USING sortheap 1000')
  ```

  Zur Aktualisierung nur einer Datenbankpartition rufen Sie die Prozedur ADMIN_CMD wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG DBPARTITIONNUM 10 USING sortheap 1000')
  ```
- Sie können den Parameter **DBPARTITIONNUM** mit der API db2CfgSet verwenden. Die Markierungen (Flags) in der Struktur **db2Cfg** geben an, ob der Wert für

die Datenbankkonfiguration auf nur eine Datenbankpartition angewendet werden soll. Wenn Sie eine Markierung setzen, müssen Sie auch den Wert für **DBPARTITIONNUM** angeben. Beispiel:

```
#define db2CfgSingleDbpartition          256
```

Wenn Sie den Wert für db2CfgSingleDbpartition nicht definieren, gilt der Wert für die Datenbankkonfiguration für alle Datenbankpartitionen, sofern Sie nicht die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** auf den Wert TRUE setzen oder für die API 'db2CfgSet', die die Konfigurationsparameter des Datenbankmanagers oder der Datenbank einstellt, das Feld *versionNumber* auf einen beliebigen Wert setzen, der kleiner als die Versionsnummer für Version 9.5 ist.

Bei der Migration der Datenbanken auf Version 9.5 behalten vorhandene Konfigurationsparameter im Allgemeinen Ihre Werte auch nach der Migration. Es werden jedoch neue Parameter mit den entsprechenden Standardwerten hinzugefügt, und einige vorhandene Parameter werden auf ihre neuen Standardwerte der Version 9.5 gesetzt. Der Abschnitt "Änderungen am Verhalten des DB2-Servers" in der Veröffentlichung *Migration* enthält Details zu den Änderungen an vorhandenen Datenbankkonfigurationsparametern. Für alle nachfolgenden Anforderungen zum Aktualisieren oder Zurücksetzen der Datenbankkonfiguration für die migrierten Datenbanken wird die Methode von Version 9.5 zum Aktualisieren oder Zurücksetzen der Konfiguration verwendet.

Für vorhandene Befehlsscripts zum Aktualisieren oder Zurücksetzen gelten ebenfalls die zuvor genannten Regeln: Sie können die Methode vor Version 9.5 verwenden, Sie können Ihre Scripts ändern, um die Option **DBPARTITIONNUM** des Befehls UPDATE DATABASE CONFIGURATION bzw. RESET DATABASE CONFIGURATION einzufügen oder Sie können die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren.

Für vorhandene Anwendungen, die die API db2CfgSet aufrufen, müssen Sie die Methode von Version 9.5 verwenden. Wenn Sie die Methode vor Version 9.5 verwenden wollen, können Sie die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren oder Ihre Anwendungen in der Weise ändern, dass sie die API mit der Versionsnummer von Version 9.5, einschließlich der neuen Markierung (Flag) db2CfgSingleDbpartition und des neuen Felds **dbpartitionnum**, aufrufen, um Datenbankkonfigurationen für eine bestimmte Datenbankpartition zu aktualisieren bzw. zurückzusetzen.

Anmerkung: Wenn Sie feststellen, dass Datenbankkonfigurationswerte inkonsistent sind, können Sie jede Datenbankpartition einzeln aktualisieren oder zurücksetzen.

Gemeinsam genutzte Tabelle für Dateikennungen

Der mit Threads arbeitende Datenbankmanager verwaltet nur eine gemeinsam genutzte Dateikennungstabelle für jede Datenbank und alle Agenten, die für die einzelnen Datenbanken aktiv sind, sodass E/A-Anforderungen, die sich auf dieselbe Datei beziehen, kein erneutes Öffnen und Schließen der Datei erfordern.

Vor Version 9.5 wurde die Dateikennungstabelle von jedem DB2-Agenten separat verwaltet, und die Größe der Dateikennungstabelle pro Agent wurde über den Konfigurationsparameter **maxfilop** gesteuert. Ab Version 9.5 verwaltet der Datenbankmanager eine einzige gemeinsam genutzte Tabelle für Dateikennungen für die gesamte Datenbank, sodass dieselbe Dateikennung von allen Agenten, die für dieselbe Datenbankdatei aktiv sind, gemeinsam genutzt werden kann. Infolgedessen wird der Konfigurationsparameter **maxfilop** zum Steuern der Größe der gemeinsam genutzten Dateikennungstabelle verwendet.

Aufgrund dieser Änderung hat der Konfigurationsparameter **maxfilop** einen neuen Standardwert sowie einen neuen Mindest- und Höchstwert. Bei einer Datenbankmigration wird der Konfigurationsparameter **maxfilop** automatisch auf die neuen Standardwerte gesetzt.

Ausführen von Bibliotheksfunktionen anderer Anbieter in Prozessen im abgeschirmten Modus

Der Datenbankmanager unterstützt Bibliotheksfunktionen anderer Anbieter in Prozessen des abgeschirmten Modus, die solche Aufgaben ausführen, wie Datenkomprimierung, TSM-Backups und Protokolldatenarchivierung.

Vor Version 9.5 wurden Bibliotheksfunktionen, Dienstprogramme oder Routinen anderer Anbieter innerhalb von Agentenprozessen ausgeführt. Da ab Version 9.5 der DB2-Datenbankmanager selbst eine Multithreading-Anwendung ist, können Bibliotheksfunktionen anderer Anbieter, die nicht mehr threadsicher sind, Speicher- oder Stackdatenverluste oder, noch schwer wiegender, Datenbeschädigungen in DB2-Datenbanken verursachen. Aus diesem Grund wird für jeden Aufruf eines Dienstprogramms eines anderen Anbieters ein neuer Prozess im abgeschirmten Modus erstellt, und die Bibliotheksfunktionen oder Routinen des Anbieters werden in diesem Prozess im abgeschirmten Modus ausgeführt. Dies wird nicht zu wesentlichen Leistungseinbußen führen.

Anmerkung: Die Funktionalität des abgeschirmten Modus ist für Windows-Plattformen nicht verfügbar.

Dynamischer Speicher

Der dynamische Speicher vereinfacht die Speicherverwaltung für Tabellenbereiche. Bei der Erstellung einer Datenbank geben Sie die Speicherpfade an, in denen der Datenbankmanager Ihre Tabellenbereichsdaten speichern soll. Anschließend verwaltet der Datenbankmanager die Container und die Speicherzuordnung für die Tabellenbereiche, je nachdem, wie sie von Ihnen erstellt und mit Daten gefüllt werden.

Tabellenbereiche mit dynamischem Speicher

Bei der Erstellung eines Tabellenbereichs in einer Datenbank, für die der dynamische Speicher nicht aktiviert ist, müssen Sie eine der Klauseln MANAGED BY SYSTEM oder MANAGED BY DATABASE angeben. . Durch die Verwendung dieser Klauseln wird ein vom System verwalteter SMS-Tabellenbereich bzw. ein von der Datenbank verwalteter DMS-Tabellenbereich erstellt. In beiden Fällen müssen Sie eine explizite Liste von Containern angeben.

Wenn für eine Datenbank der dynamische Speicher aktiviert ist, sind weitere Möglichkeiten verfügbar: Sie können die Klausel MANAGED BY AUTOMATIC STORAGE angeben oder die MANAGED BY-Klausel weglassen (was die Verwendung des dynamischen Speichers impliziert). . In diesem Fall brauchen Sie keine Containerdefinitionen anzugeben, weil der Datenbankmanager die Container automatisch zuweist.

Die folgenden Beispiele zeigen einige Anweisungen zur Erstellung von Tabellenbereichen mit dynamischem Speicher:

```
CREATE TABLESPACE TS1
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
CREATE TEMPORARY TABLESPACE TEMPTS
CREATE USER TEMPORARY TABLESPACE USRTMP MANAGED BY AUTOMATIC STORAGE
CREATE LONG TABLESPACE LONGTS
```

Obwohl der Tabellenbereichstyp mit dynamischem Speicher ein anderer Tabellenbereichstyp zu sein scheint, handelt es sich tatsächlich nur um eine Erweiterung der bereits vorhandenen Tabellenbereichstypen SMS und DMS. Wenn Sie einen regulären (REGULAR) oder großen (LARGE) Tabellenbereich erstellen, wird er als DMS-Tabellenbereich mit Dateicontainern erstellt. Wenn Sie einen Tabellenbereich für temporäre Systemtabellen erstellen, wird er als SMS-Tabellenbereich mit Verzeichniscontainern erstellt.

Anmerkung: In zukünftigen Versionen des Datenbankmanagers wird dieses Verhalten möglicherweise geändert.

Die diesen Containern zugeordneten Namen haben folgendes Format:

speicherpfad/*instanzname*/NODE####/*datenbankname*/T#######/C#######.*ERW*

Dabei gilt:

speicherpfad
: Ein der Datenbank zugeordneter Speicherpfad

instanzname
: Die Instanz, unter der die Datenbank erstellt wurde

datenbankname
: Der Name der Datenbank

NODE####
: Die Datenbankpartitionsnummer (z. B. NODE0000)

T#######
: Die Tabellenbereichs-ID (z. B. T0000003)

C#######
: Die Container-ID (z. B. C0000012)

ERW
: Eine Erweiterung basierend auf dem Typ der zu speichernden Daten:

  CAT Systemkatalogtabellenbereich

  TMP Tabellenbereich für temporäre Systemtabellen

  UTM Tabellenbereich für temporäre Benutzertabellen

  USR Benutzertabellenbereich oder regulärer Tabellenbereich

  LRG LOB-Tabellenbereich

Unterschiede zwischen regulären und großen Tabellenbereichen mit dynamischem Speicher und DMS-Tabellenbereichen

Reguläre und große (LARGE) Tabellenbereiche mit dynamischem Speicher werden als DMS-Tabellenbereiche erstellt, für die alle Regeln und Funktionsweisen von DMS-Tabellenbereichen gelten. In der Verwaltung des Speichers gibt es jedoch Unterschiede, die in der folgenden Tabelle erläutert werden:

Tabelle 3. Verwaltungsunterschiede zwischen Tabellenbereichen ohne dynamischen Speicher und mit dynamischem Speicher

Nicht dynamischer Speicher	Dynamischer Speicher
Sie müssen beim Erstellen des Tabellenbereichs explizit eine Liste von Containern angeben.	Sie können keine Liste von Containern beim Erstellen des Tabellenbereichs angeben, da der Datenbankmanager Container automatisch zuweist und zuordnet.

Tabelle 3. Verwaltungsunterschiede zwischen Tabellenbereichen ohne dynamischen Speicher und mit dynamischem Speicher (Forts.)

Nicht dynamischer Speicher	Dynamischer Speicher
Die Funktion zur automatischen Größenänderung ist standardmäßig inaktiviert (AUTORESIZE ist NO).	Die Funktion zur automatischen Größenänderung ist standardmäßig aktiviert (AUTORESIZE ist YES).
Sie können die Klausel INITIALSIZE nicht verwenden, um die Anfangsgröße für den Tabellenbereich anzugeben.	Sie können die Klausel INITIALSIZE verwenden, um die Anfangsgröße für den Tabellenbereich anzugeben.
Sie können Containeroperationen mit der Anweisung ALTER TABLESPACE (unter Angabe von ADD, DROP, BEGIN NEW STRIPE SET usw.) ausführen.	Sie können keine Containeroperationen ausführen, da der Datenbankmanager den Speicherbereich verwaltet.
Sie können eine umgeleitete Restoreoperation verwenden, um die dem Tabellenbereich zugeordneten Container erneut zu definieren.	Sie können keine umgeleitete Restoreoperation dazu verwenden, die dem Tabellenbereich zugeordneten Container erneut zu definieren, da der Datenbankmanager den Speicherbereich verwaltet.

Wie in obiger Tabelle aufgeführt, können Sie beim Erstellen eines regulären oder großen Tabellenbereichs mit dynamischem Speicher die Anfangsgröße in der Anweisung CREATE TABLESPACE angeben. Beispiel:

```
CREATE TABLESPACE TS1 INITIALSIZE 100 M
```

Wenn Sie keine Anfangsgröße angeben, verwendet der Datenbankmanager den Standardwert von 32 MB.

Zur Erstellung eines Tabellenbereichs mit einer bestimmten Größe erstellt der Datenbankmanager Dateicontainer in den Speicherpfaden. Wenn die Verteilung des Speicherplatzes in den Pfaden ungleichmäßig ist, werden Container möglicherweise mit unterschiedlichen Größen erstellt. Daher ist es wichtig, dass in allen Speicherpfaden eine annähernd gleiche Menge an freiem Speicherplatz zur Verfügung steht.

Wenn Sie die Funktion zur automatischen Größenänderung für den Tabellenbereich aktivieren, erweitert der Datenbankmanager entsprechend dem in diesem Tabellenbereich belegten Speicherplatz automatisch die vorhandenen Container und fügt neue hinzu (durch Stripe-Sets). Weder beim Erweitern noch beim Hinzufügen von Containern findet eine Umverteilung der Daten statt.

Automatische Änderung der Größe von Tabellenbereichen

Durch die Aktivierung von Tabellenbereichen mit dynamischem Speicher zur automatischen Größenänderung kann der Datenbankmanager die Problembedingung eines vollen Dateisystems automatisch durch Hinzufügen eines neuen Stripe-Sets mit Containern beheben.

In einem Datenbanksystem können zwei Typen von Tabellenbereichen enthalten sein: vom System verwaltete Bereiche (SMS) und von der Datenbank verwaltete Bereiche (DMS). Die Container, die SMS-Tabellenbereichen zugeordnet sind, sind Dateisystemverzeichnisse und die Dateien in diesen Verzeichnissen werden in gleichem Maße größer wie die Objekte im entsprechenden Tabellenbereich. Die Dateien wachsen, bis für einen der Container eine Dateisystembegrenzung oder die Größenbegrenzung der Datenbank für Tabellenbereiche erreicht wird (siehe SQL- und XML-Begrenzungen).

DMS-Tabellenbereiche bestehen aus Dateicontainern oder Containern für (unformatierte und unpartitionierte) Roheinheiten, und ihre Größe wird definiert, wenn die Container dem Tabellenbereich zugeordnet werden. Der Tabellenbereich gilt dann als voll, wenn der gesamte Speicherplatz des Containers belegt ist. Im Gegensatz zu SMS-Tabellenbereichen können hier jedoch Container mit der Anweisung ALTER TABLESPACE hinzugefügt oder erweitert werden, wodurch der Tabellenbereich mehr Speicherplatz erhält. DMS-Tabellenbereiche verfügen darüber hinaus über eine *Funktion zur automatischen Größenänderung (AUTORESIZE)*: Wenn in einem DMS-Tabellenbereich, für den die automatische Größenänderung aktiviert ist, der Speicherplatz erschöpft ist, kann das Datenbanksystem den Tabellenbereich um einen oder mehrere Dateicontainer erweitern. SMS-Tabellenbereiche verfügen über ähnliche Funktionen zur automatischen Erweiterung, doch der Begriff der Funktion zur automatischen Größenänderung (AUTORESIZE) wird ausschließlich für DMS verwendet.

Die automatische Größenänderung von Tabellenbereichen hat die folgenden Auswirkungen:

- Tabellenbereichen, für die die automatische Größenänderung aktiviert ist, sind Metadaten zugeordnet, die von Version 8.2.1 oder früheren Releases nicht erkannt werden. Jeder Versuch, eine Datenbank mit Tabellenbereichen, für die die automatische Größenänderung aktiviert ist, mit diesen Versionen zu verwenden, führt zu einem Fehler (normalerweise SQL0980C oder SQL0902C). Ein Fehler kann bei dem Versuch, eine Verbindung zu einer Datenbank herzustellen, oder bei dem Versuch, einen Restore der Datenbank auszuführen, zurückgegeben werden. Wenn Sie die automatische Größenänderung für Tabellenbereiche aktiviert haben, werden durch die Inaktivierung der Funktion zur automatischen Größenänderung für diese Tabellenbereiche die Metadaten entfernt, sodass die Datenbank mit Version 8.2.1 oder früheren Releases verwendet werden kann.
- Wenn Sie die Funktion zur automatischen Größenänderung inaktivieren, gehen die Werte, die INCREASESIZE und MAXSIZE zugeordnet sind, verloren und stehen bei einer späteren Aktivierung der Funktion nicht mehr zur Verfügung.
- Sie können diese Funktion nicht für Tabellenbereiche aktivieren, die mit Containern für Roheinheiten arbeiten, und Sie können einem Tabellenbereich, für den die automatische Größenänderung aktiviert ist, keine Container für Roheinheiten hinzufügen. Versuche, solche Operationen auszuführen, schlagen mit Fehlern (SQL0109N) fehl. Wenn Sie Container für Roheinheiten hinzufügen müssen, müssen Sie diese Funktion zuvor inaktivieren.
- Eine umgeleitete Restoreoperation kann die Containerdefinitionen nicht so ändern, dass ein Container für eine Roheinheit verwendet werden kann. Ein Versuch, diese Art von Operation auszuführen, verursacht einen Fehler (SQL0109N).
- Da der Wert für die Maximalgröße die automatische Vergrößerung eines Tabellenbereichs durch den Datenbankmanager begrenzt, legt dieser Maximalwert auch fest, bis zu welcher Größe Sie einen Tabellenbereich vergrößern können. Dies bedeutet, dass bei einer von Ihnen ausgeführten Operation, die einem Tabellenbereich Speicherbereich hinzufügt, die resultierende Größe kleiner oder gleich der Maximalgröße sein muss. Speicherbereich kann mit der Klausel ADD, EXTEND, RESIZE oder BEGIN NEW STRIPE SET der Anweisung ALTER TABLESPACE hinzugefügt werden.

Aktivieren und Inaktivieren der Funktion zur automatischen Größenänderung (AUTORESIZE)

Standardmäßig ist die Funktion zur automatischen Größenänderung für einen DMS-Tabellenbereich nicht aktiviert. Mit der folgenden Anweisung wird ein DMS-Tabellenbereich ohne Aktivierung der automatischen Größenänderung erstellt:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
```

Zum Aktivieren der Funktion zur automatischen Größenänderung geben Sie die Klausel AUTORESIZE YES in der Anweisung CREATE TABLESPACE an:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M) AUTORESIZE YES
```

Sie können die Funktion zur automatischen Größenänderung auch nach der Erstellung eines DMS-Tabellenbereichs aktivieren oder inaktivieren, indem Sie die Klausel AUTORESIZE in der Anweisung ALTER TABLESPACE verwenden:

```
ALTER TABLESPACE DMS1 AUTORESIZE YES
ALTER TABLESPACE DMS1 AUTORESIZE NO
```

Darüber hinaus sind zwei weitere Attribute, MAXSIZE und INCREASESIZE, für die automatische Größenänderung von Tabellenbereichen relevant:

Maximalgröße (MAXSIZE)

Mit der Klausel MAXSIZE der Anweisung CREATE TABLESPACE wird die maximal mögliche Größe für den Tabellenbereich definiert. So wird beispielsweise durch die folgende Anweisung ein Tabellenbereich erstellt, der bis zu einem Maximalwert von 100 MB vergrößert werden kann (pro Datenbankpartition, falls die Datenbank mehrere Datenbankpartitionen besitzt):

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES MAXSIZE 100 M
```

Die Klausel MAXSIZE NONE gibt an, dass für den Tabellenbereich kein Maximalwert festgelegt ist. Der Tabellenbereich kann wachsen, bis eine Dateisystembegrenzung oder die Größenbegrenzung für Tabellenbereiche erreicht wird (siehe SQL- und XML-Begrenzungen). Wenn Sie die Klausel MAXSIZE nicht angeben, gibt es keine Maximalgröße, wenn die Funktion zur automatischen Größenänderung aktiviert wird.

Mit der Anweisung ALTER TABLESPACE können Sie den Wert von MAXSIZE für einen Tabellenbereich ändern, für den die Funktion zur automatischen Größenänderung bereits aktiviert ist, wie in den folgenden Beispielen gezeigt:

```
ALTER TABLESPACE DMS1 MAXSIZE 1 G
ALTER TABLESPACE DMS1 MAXSIZE NONE
```

Wenn Sie eine Maximalgröße angeben, kann der tatsächlich vom Datenbankmanager umgesetzte Wert geringfügig niedriger sein als der angegebene Wert, da der Datenbankmanager versucht, die Vergrößerung von Containern konsistent zu halten.

Vergrößerungsvolumen (INCREASESIZE)

Mit der Klausel INCREASESIZE der Anweisung CREATE TABLESPACE wird die Menge an Speicherplatz definiert, um die der Tabellenbereich vergrößert wird, wenn keine freien Speicherbereiche im Tabellenbereich mehr vorhanden sind, jedoch eine Anforderung für einen oder mehrere Speicherbereiche erfolgt ist. Sie können den Wert als explizite Größe oder als Prozentsatz wie in den folgenden Beispielen angeben:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 5 M
```

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 50 PERCENT
```

Ein Prozentsatz bedeutet, dass das Volumen, um das entsprechend dem Wert von INCREASESIZE zu vergrößern ist, jedes Mal, wenn der Tabellenbereich vergrößert werden muss, neu berechnet wird. Dabei basiert das jeweilige Vergrößerungsvolumen auf einem Prozentsatz der Tabellenbereichsgröße zu dem betreffenden Zeitpunkt. Beispiel: Wenn der Tabellenbereich 20 MB groß ist und für INCREASESIZE ein Wert von 50 % angegeben wird, wird der Tabellenbereich bei der ersten Vergrößerung um 10 MB erweitert (auf eine Größe von 30 MB) und bei der nächsten Vergrößerung um 15 MB.

Wenn Sie die Klausel INCREASESIZE bei der Aktivierung der Funktion zur automatischen Größenänderung nicht angegeben, ermittelt der Datenbankmanager einen angemessenen Wert, der sich während des Lebenszyklus des Tabellenbereichs ändern kann. Wie bei AUTORESIZE und MAXSIZE können Sie den Wert für INCREASESIZE mithilfe der Anweisung ALTER TABLESPACE ändern.

Wenn Sie einen Wert für INCREASESIZE angeben, kann der tatsächlich vom Datenbankmanager verwendete Wert geringfügig vom angegebenen Wert abweichen. Diese Anpassung des verwendeten Wertes dient dazu, die Vergrößerungen für alle Container im Tabellenbereich konsistent zu halten.

Erweiterung von Tabellenbereichen

Bei Tabellenbereichen, deren Größe automatisch geändert werden kann, versucht der Datenbankmanager, den Tabellenbereich zu vergrößern, wenn der gesamte vorhandene Speicherbereich belegt ist und eine Anforderung für zusätzlichen Speicherbereich abgesetzt wurde. Der Datenbankmanager bestimmt, welche Container im Tabellenbereich erweitert werden können, sodass kein Neuausgleich stattfindet. Der Datenbankmanager erweitert nur die Container, die sich im letzten Bereich der Tabellenbereichszuordnung befinden (diese Zuordnung beschreibt die Speicherbelegung für den Tabellenbereich), und erweitert sie um dasselbe Volumen.

Betrachten Sie z. B. die folgende Anweisung:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
  USING (FILE 'C:\TS1CONT' 1000, FILE 'D:\TS1CONT' 1000,
         FILE 'E:\TS1CONT' 2000, FILE 'F:\TS1CONT' 2000)
  EXTENTSIZE 4
  AUTORESIZE YES
```

Unter Berücksichtigung der Tatsache, dass der Datenbankmanager einen kleinen Teil jedes Containers (einen EXTENTSIZE großen Speicherbereich) für Metadaten verwendet, ist nachfolgend die Tabellenbereichszuordnung dargestellt, die für den Tabellenbereich auf der Basis der Anweisung CREATE TABLESPACE erstellt wird. (Die Tabellenbereichszuordnung ist Teil der Ausgabe einer Momentaufnahme für einen Tabellenbereich.)

```
Tabellenbereichszuordnung:

 Range  Stripe Stripe  Max         Max  Start  End    Adj.   Containers
 Number Set    Offset  Extent      Page Stripe Stripe
 [   0] [   0]      0     995      3983      0    248   0    4 (0,1,2,3)
 [   1] [   0]      0    1495      5983    249    498   0    2 (2,3)
```

Die Tabellenbereichszuordnung zeigt, dass die Container mit der ID 2 und 3 (`E:\TS1CONT` und `F:\TS1CONT`) die einzigen Container im letzten Bereich (Range) der Zuordnung sind. Wenn der Datenbankmanager die Container in diesem Tabellenbereich automatisch erweitert, werden daher nur diese beiden Container erweitert.

Anmerkung: Wenn Sie einen Tabellenbereich erstellen, bei dem alle Container dieselbe Größe haben, enthält die Zuordnung nur einen Bereich (Range). In diesem Fall erweitert der Datenbankmanager jeden Container. Um zu verhindern, dass die Erweiterung nur auf eine Untermenge der Container begrenzt wird, müssen Sie einen Tabellenbereich mit Containern gleicher Größe erstellen.

Wie zuvor erläutert, können Sie eine Begrenzung für die Maximalgröße des Tabellenbereichs angeben. Sie können auch den Wert NONE angeben, sodass keine Größenbegrenzung besteht. Wenn Sie NONE oder keinen Grenzwert angeben, wird die Obergrenze durch den Grenzwert des Dateisystems oder des Tabellenbereichs definiert. Der Datenbankmanager versucht nicht, die Tabellenbereichsgröße über die Obergrenze hinaus zu erweitern. Es kann jedoch vorkommen, dass ein Versuch, einen Container zu erweitern, fehlschlägt, bevor dieser Grenzwert erreicht ist, wenn das Dateisystem voll ist. In diesem Fall vergrößert der Datenbankmanager den Tabellenbereich nicht weiter und gibt eine Fehlerbedingung wegen nicht ausreichenden Speicherplatzes an die Anwendung zurück. In dieser Situation haben Sie zwei Möglichkeiten zur Lösung des Problems:

- Stellen Sie mehr verfügbaren Speicherplatz in dem vollen Dateisystem bereit.
- Führen Sie Containeroperationen für den Tabellenbereich aus, die bewirken, dass sich der betreffende Container nicht mehr im letzten Bereich (Range) der Tabellenbereichszuordnung befindet. Die einfachste Methode, dies zu erreichen, besteht darin, dem Tabellenbereich ein neues Stripe-Set mit einer neuen Gruppe von Containern hinzuzufügen, wobei nach Möglichkeit sichergestellt werden sollte, dass alle Container dieselbe Größe haben. Sie können neue Stripe-Sets mit der Klausel BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE hinzufügen. Durch das Hinzufügen eines neuen Stripe-Sets wird der Tabellenbereichszuordnung ein neuer Bereich hinzugefügt. Durch einen neu hinzugefügten Bereich befinden sich die Container, die der Datenbankmanager automatisch zu erweitern versucht, in diesem neuen Stripe-Set, und die älteren Container bleiben unverändert.

  Anmerkung: Wenn eine vom Benutzer eingeleitete Containeroperation ansteht oder gerade ein nachfolgender Neuausgleich ausgeführt wird, wird die Funktion zur automatischen Größenänderung so lange inaktiviert, bis die Operation festgeschrieben bzw. der Neuausgleich abgeschlossen ist.

Ein Beispiel für DMS-Tabellenbereiche: Nehmen Sie einmal an, dass ein Tabellenbereich drei Container umfasst, die dieselbe Größe haben und sich jeweils in einem eigenen Dateisystem befinden. Während der Ausführung von Operationen im Tabellenbereich erweitert der Datenbankmanager diese drei Container automatisch. Schließlich ist eines der Dateisysteme voll, und der zugehörige Container kann nicht mehr erweitert werden. Wenn es nicht möglich ist, im Dateisystem zusätzlichen freien Speicherplatz verfügbar zu machen, müssen Sie Containeroperationen im Tabellenbereich durchführen, die bewirken, dass sich der betreffende Container nicht mehr im letzten Bereich (Range) der Tabellenbereichszuordnung befindet. In diesem Fall könnten Sie beispielsweise ein neues Stripe-Set hinzufügen und dabei zwei Container angeben (jeweils einen für jedes Dateisystem, in dem noch Speicherplatz zur Verfügung steht); oder Sie könnten eine größere oder geringere Anzahl von Containern angeben (und dabei sicherstellen, dass jeder der hinzugefügten Container dieselbe Größe hat und in jedem der verwendeten Dateisysteme

ausreichend Speicherplatz für mögliche Erweiterungen verfügbar ist). Beim Vergrößern des Tabellenbereichs versucht der Datenbankmanager nun, anstelle der älteren Container die Container in diesem neuen Stripe-Set zu erweitern.

Überwachung

Informationen zur automatischen Größenänderung von DMS-Tabellenbereichen sind Teil der Momentaufnahmenausgabe des Tabellenbereichsmonitors. Die Werte für das Vergrößerungsvolumen (INCREASESIZE) und für die Maximalgröße (MAXSIZE) werden ebenfalls in der Ausgabe wie im folgenden Beispiel angezeigt:

```
Autom. Größenänderung aktiviert               = Ja oder Nein
Aktuelle Größe des Tabellenbereichs (Byte)    = ###
Maximale Größe des Tabellenbereichs (Byte)    = ### oder NONE
Vergrößerungsvolumen (Byte)                   = ###
Vergrößerungsvolumen (Prozent)                = ###
Zeit der letzten erfolgreichen Größenänderung = TT/MM/JJJJ HH:MM:SS.SSSSSS
Letzte fehlgeschlagene Größenänderung         = Ja oder Nein
```

Datenbanken mit dynamischem Speicher

Der Datenbankmanager erstellt standardmäßig alle Datenbanken *mit dynamischem Speicher*. Wenn Sie eine Datenbank „ohne dynamischen Speicher" erstellen wollen, müssen Sie im Befehl CREATE DATABASE die Klausel AUTOMATIC STORAGE NO angeben.

Datenbanken, für die der dynamische Speicher aktiviert ist, besitzen einen zugeordneten Speicherpfad bzw. eine Gruppe zugeordneter Speicherpfade. Auf der Grundlage dieser Speicherpfade können Sie Tabellenbereiche als *vom dynamischen Speicher verwaltet* und die zugehörigen Container als durch den Datenbankmanager zugeordnet definieren.

Sie können den dynamischen Speicher für eine Datenbank nur bei der Erstellung der Datenbank aktivieren. Ebenso können Sie den dynamischen Speicher für eine Datenbank nicht inaktivieren, für die dessen Verwendung ursprünglich definiert wurde.

Alle Datenbanken werden standardmäßig mit dynamischem Speicher erstellt. Wenn Sie eine Datenbank ohne dynamischen Speicher erstellen wollen, müssen Sie die Option **AUTOMATIC STORAGE NO** im Befehl CREATE DATABASE angeben.

Beispielbefehle für die Erstellung von Datenbanken ohne dynamischen Speicher:

```
CREATE DATABASE ASNODB1 AUTOMATIC STORAGE NO
CREATE DATABASE ASNODB2 AUTOMATIC STORAGE NO ON X:
```

Beispielbefehle für die Erstellung von Datenbanken mit expliziter oder impliziter Aktivierung des dynamischen Speichers:

```
CREATE DATABASE DB1
CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:
CREATE DATABASE DB3 ON /data/pfad1, /data/pfad2
CREATE DATABASE DB4 ON D:\SpeicherPfad DBPATH ON C:
```

Auf der Basis der verwendeten Syntax extrahiert der Datenbankmanager die beiden folgenden Informationen, die sich auf Speicherpositionen beziehen:

- Den Datenbankpfad (in dem der Datenbankmanager eine Reihe von Steuerdateien für die Datenbank speichert):

  - Wenn Sie **DBPATH ON** angeben, geben Sie damit den Datenbankpfad an.
  - Wenn Sie **DBPATH ON** nicht angeben, gibt der erste Pfad in **ON** den Datenbankpfad (und den Speicherpfad) an.
  - Wenn Sie weder **DBPATH ON** noch **ON** angeben, wird der Konfigurationsparameter **dftdbpath** des Datenbankmanagers zur Bestimmung des Datenbankpfades verwendet.
- Die Speicherpfade (in denen der Datenbankmanager Tabellenbereichscontainer mit dynamischem Speicher erstellt):
  - Wenn Sie **ON** angeben, sind alle aufgeführten Pfade Speicherpfade.
  - Wenn Sie **ON** nicht angeben, wird nur ein Speicherpfad verwendet, der dem Wert des Konfigurationsparameters **dftdbpath** des Datenbankmanagers entspricht.

In der folgenden Tabelle sind die verwendeten Datenbank- und Speicherpfade für die oben gezeigten Beispiele zusammengefasst:

Tabelle 4. Dynamischer Speicher - Datenbankpfade und Speicherpfade

Befehl CREATE DATABASE	Datenbankpfad	Speicherpfade
`CREATE DATABASE DB1 AUTOMATIC STORAGE YES`	Wert des Konfigurationsparameters **dftdbpath**	Wert des Konfigurationsparameters **dftdbpath**
`CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:`	X:	X:
`CREATE DATABASE DB3 ON /data/pfad1, /data/pfad2`	/data/pfad1	/data/pfad1, /data/pfad2
`CREATE DATABASE DB4 ON D:\SpeicherPfad DBPATH ON C:`	C:	D:\SpeicherPfad

Die angegebenen Speicherpfade müssen vorhanden und zugänglich sein. In einer Umgebung mit partitionierten Datenbanken werden in jeder Datenbankpartition die gleichen Speicherpfade verwendet. Sie können keine eindeutige Gruppe von Speicherpfaden für eine bestimmte Datenbankpartition angeben, sofern Sie nicht Datenbankpartitionsausdrücke als Teil der Speicherpfadnamen verwenden. Dieses Verfahren bietet die Möglichkeit, die Datenbankpartitionsnummer in die Speicherpfade einzufügen, sodass die sich ergebenden Pfadnamen für jede Datenbankpartition unterschiedlich ist.

Verwenden Sie das Argument `$N` (mit einem Leerzeichen vor `$N`!), um einen Datenbankpartitionsausdruck anzugeben. Sie können einen Datenbankpartitionsausdruck an einer beliebigen Stelle im Speicherpfad verwenden, und Sie können mehrere Datenbankpartitionsausdrücke angeben. Schließen Sie den Datenbankpartitionsausdruck mit einem Leerzeichen ab. Alle weiteren Zeichen nach dem Leerzeichen werden an den Speicherpfad angehängt, nachdem der Datenbankpartitionsausdruck ausgewertet wurde. Wenn nach dem Datenbankpartitionsausdruck im Speicherpfad kein Leerzeichen folgt, wird der Rest der Zeichenfolge als Teil des Ausdrucks interpretiert. In der folgenden Tabelle sind die einzigen gültigen Formen des Arguments `$N` aufgeführt. Operatoren werden von links nach rechts ausgewertet, und das Zeichen % stellt den Modulusoperator dar. Die Datenbankpartitionsnummer in den Beispielen ist 10.

Tabelle 5. Datenbankpartitionsausdrücke

Syntax	Beispiel	Wert
[Leerzeichen]$N	`" $N"`	10

Tabelle 5. Datenbankpartitionsausdrücke (Forts.)

Syntax	Beispiel	Wert
[Leerzeichen]$N+[zahl]	" $N+100"	110
[Leerzeichen]$N%[zahl]	" $N%5"	0
[Leerzeichen]$N+[zahl]%[zahl]	" $N+1%5"	1
[Leerzeichen]$N%[zahl]+[zahl]	" $N%4+2"	4

Das folgende Beispiel zeigt die Verwendung von Datenbankpartitionsausdrücken:

```
CREATE DATABASE TESTDB ON "/pfad1FuerKnoten $N",
  "/pfad2FuerKnoten $N" DBPATH ON "/dbpfadFuerKnoten"
```

Das folgende Beispiel zeigt einen in der Mitte eines Pfads eingebetteten Datenbankpartitionsausdruck:

```
CREATE DATABASE TESTDB ON "/pfad1FuerKnoten $N",
  "/pfad2FuerKnoten $N suffix" DBPATH ON "/dbpfadFuerKnoten"
```

Anmerkung: Datenbankpartitionsausdrücke sind in Datenbankpfaden nicht gültig, unabhängig davon, ob sie explizit mit **DBPATH ON** oder implizit durch einen Datenbankpartitionsausdruck im ersten Speicherpfad angegeben werden.

Bei der Berechnung des freien Speicherbereichs für einen Speicherpfad für eine gegebene Datenbankpartition prüft der Datenbankmanager, ob die folgenden Verzeichnisse (bzw. Mountpunkte) innerhalb des Speicherpfads vorhanden sind, und verwendet das erste Verzeichnis, das er findet:

speicherpfad/*instanzname*/NODE####/*datenbankname*
speicherpfad/*instanzname*/NODE####
speicherpfad/*instanzname*
speicherpfad

Dabei gilt Folgendes:

speicherpfad
: Ein der Datenbank zugeordneter Speicherpfad

instanzname
: Die Instanz, unter der sich die Datenbank befindet

NODE####
: Die Datenbankpartitionsnummer (z. B. NODE0000 oder NODE0001)

datenbankname
: Der Name der Datenbank

Dateisysteme können an einem Punkt unterhalb des Speicherpfads angehängt sein. In diesem Fall erkennt der Datenbankmanager, dass die tatsächliche Kapazität an freiem Speicherbereich, die für Tabellenbereichscontainer verfügbar ist, möglicherweise nicht die gleiche Kapazität ist, die dem Speicherpfadverzeichnis selbst zugeordnet ist.

Betrachten Sie ein Beispiel, in dem zwei logische Datenbankpartitionen auf einem physischen Computer angelegt sind und nur ein Speicherpfad vorhanden ist: /db2data. Beide Datenbankpartitionen können diesen einen Speicherpfad verwenden. Sie möchten jedoch möglicherweise die Daten jeder Datenbankpartition isolieren, indem Sie für jede jeweils ein eigenes Dateisystem erstellen. Das Dateisystem wird an /db2data/*instanz*/NODE#### angehängt. Beim Erstellen von Containern im Speicherpfad und beim Ermitteln des freien Speicherbereichs ruft der

Datenbankmanager nicht die Informationen über den freien Speicherbereich im Pfad /db2data, sondern die Informationen für das entsprechende Verzeichnis /db2data/*instanz*/NODE#### ab.

Jedes Mal, wenn Sie eine Datenbank erstellen, werden drei Standardtabellenbereiche erstellt. Wenn Sie im Befehl CREATE DATABASE keine expliziten Tabellenbereichsdefinitionen angeben, werden die Tabellenbereiche als Tabellenbereiche mit dynamischem Speicher erstellt.

Nach der Erstellung einer Datenbank können Sie ihr neue Speicherpfade hinzufügen, indem Sie die Klausel ADD STORAGE der Anweisung ALTER DATABASE wie im folgenden Beispiel verwenden:

```
ALTER DATABASE ADD STORAGE ON '/data/pfad3', '/data/pfad4'
```

Hinzufügen von Pfaden für den dynamischen Speicher in Datenbanken mit aktiviertem dynamischen Speicher

Mit der Anweisung ALTER DATABASE können Sie einer Datenbank, die für den dynamischen Speicher eingerichtet ist, einen Pfad für den dynamischen Speicher hinzufügen. Sie können den dynamischen Speicher für eine Datenbank nur bei der Erstellung der Datenbank aktivieren.

Wenn Sie einen Speicherpfad für eine Umgebung mit mehreren Datenbankpartitionen hinzufügen, muss der Speicherpfad in jeder Datenbankpartition vorhanden sein. Wenn der angegebene Pfad nicht in jeder Datenbankpartition vorhanden ist, wird die Anweisung rückgängig gemacht.

Geben Sie die Anweisung ALTER DATABASE wie folgt ein, um einer vorhandenen Datenbank einen Speicherpfad hinzuzufügen:

```
ALTER DATABASE PATH pfadname
```

Einschränkungen für dynamischen Speicher

Bei der Entscheidung, ob eine Datenbank mit dynamischem Speicher erstellt werden soll, sind einige Einschränkungen zu berücksichtigen.

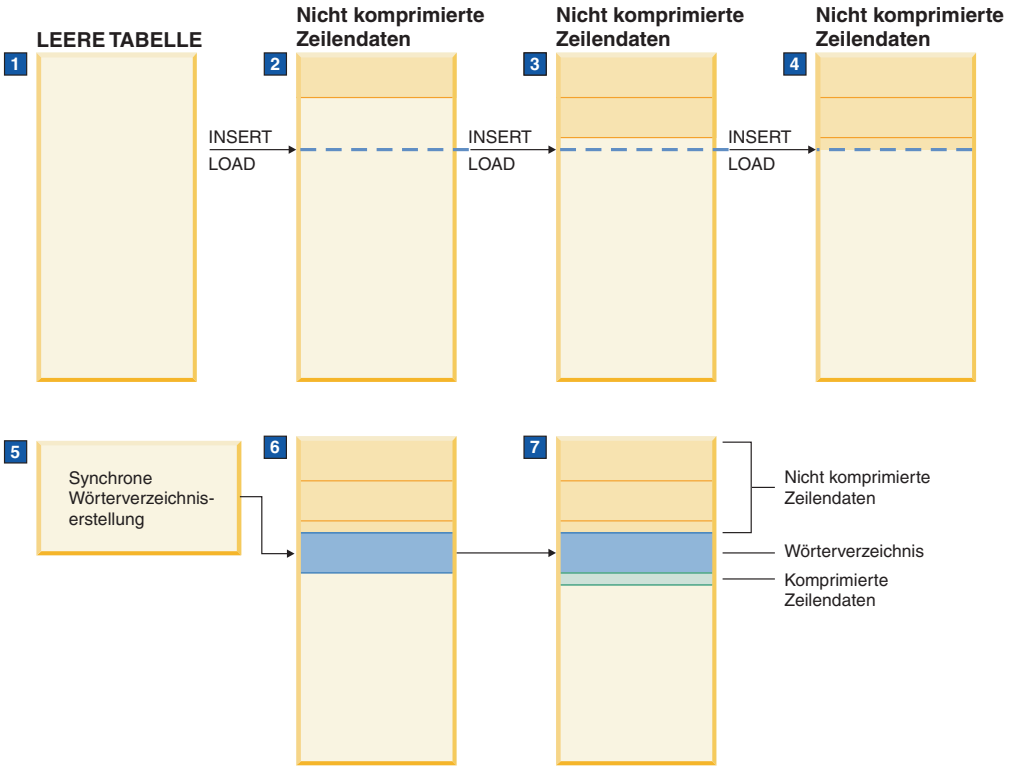
- Sie können den dynamischen Speicher für eine Datenbank nach ihrer Erstellung nicht inaktivieren oder aktivieren.
- Speicherpfade müssen absolute Pfadnamen sein. Unter dem Windows-Betriebssystem sind Pfade oder Laufwerkbuchstaben möglich. Der Datenbankpfad muss ein Laufwerkbuchstabe sein. Die maximal zulässige Pfadlänge beträgt 175 Zeichen.
- Für partitionierte Datenbanken müssen Sie in jeder Datenbankpartition die gleiche Gruppe von Speicherpfaden verwenden (sofern Sie keine Datenbankpartitionsausdrücke verwenden).
- Datenbankpartitionsausdrücke sind in Datenbankpfaden nicht gültig, unabhängig davon, ob Sie sie explizit mit der Option **DBPATH ON** des Befehls CREATE DATABASE oder implizit durch einen Datenbankpartitionsausdruck im ersten Speicherpfad angeben.

Automatische Erstellung von Komprimierungswörterverzeichnissen (ADC)

Ein Komprimierungswörterverzeichnis (Compression Dictionary) dient zur Komprimierung von Daten, die in eine Tabelle versetzt werden, um Speicherplatz freizugeben, sodass mehr Daten in die Tabelle eingefügt werden können. Ein Komprimierungswörterverzeichnis wird automatisch während einer Dateneinfügeoperation (z. B. einer LOAD- oder INSERT-Operation) erstellt und in eine Tabelle eingefügt bzw. an eine Tabelle angehängt.

Die automatische Komprimierungswörterverzeichniserstellung (ADC - Automatic (Compression) Dictionary Creation) findet für eine Tabelle statt, wenn Sie das Attribut COMPRESS für die Tabelle definiert haben, wenn noch kein Komprimierungswörterverzeichnis innerhalb der physischen Tabelle oder Partition vorhanden ist und wenn die Tabelle genügend Daten enthält. Nachfolgend in die Tabelle versetzte Daten werden mithilfe des Komprimierungswörterverzeichnisses komprimiert (sofern das Attribut COMPRESS der Tabelle aktiviert bleibt).

Das folgende Diagramm zeigt den Prozess der automatischen Erstellung des Komprimierungswörterverzeichnisses:



1. Es wird kein Komprimierungswörterverzeichnis erstellt, da die Tabelle leer ist.
2. Daten werden durch INSERT- oder LOAD-Operationen in die Tabelle eingefügt und bleiben unkomprimiert.
3. Wenn weitere Daten in die Tabelle eingefügt oder geladen werden, bleiben diese unkomprimiert.
4. Nach Erreichen eines Schwellenwerts wird die Wörterverzeichniserstellung automatisch ausgelöst, wenn das Attribut COMPRESS auf den Wert YES gesetzt ist.

5. Das Wörterverzeichnis wird erstellt.
6. Das Wörterverzeichnis wird an die Tabelle angehängt.
7. Von diesem Punkt an werden die Daten komprimiert.

In der folgenden Tabelle sind die Unterschiede in der Erstellung von Komprimierungswörterverzeichnissen nach Release aufgeführt:

Tabelle 6. Unterschiede in der Erstellung von Komprimierungswörterverzeichnissen nach Release

Befehle und Attribute	**Version 9.1**	**Version 9.5**
Befehl LOAD REPLACE mit der Option **RESETDICTIONARY**	Nicht gültig.	Wenn Sie das Attribut COMPRESS der Tabelle auf YES setzen, wird jedes vorhandene Komprimierungswörterverzeichnis entfernt und ein neues generiert, sofern mindestens eine Zeile mit Daten in die Tabelle geladen oder eingefügt wird.
CREATE oder ALTER TABLE mit dem Attribut COMPRESS auf YES	Die Wörterverzeichniserstellung war nicht automatisch. Zum Komprimieren von Tabellendaten mussten Sie explizit ein Komprimierungswörterverzeichnis durch einen Tabellenreorganisationsprozess erstellen.	Durch Setzen des Attributs COMPRESS einer Tabelle auf den Wert YES wird die Tabelle für die automatische Wörterverzeichniserstellung (ADC) auswählbar, wenn mindestens eine Zeile von Daten in die Tabelle geladen oder eingefügt wird.
Befehl INSERT, LOAD INSERT, IMPORT INSERT oder REDISTRIBUTE	Nicht gültig.	Wenn Sie das Attribut COMPRESS der Tabelle auf YES setzen, unterliegt eine Tabelle, die noch kein Komprimierungswörterverzeichnis besitzt, der automatischen Wörterverzeichniserstellung (ADC), sofern die Tabelle genügend Daten enthält (d. h., wenn der entsprechende Schwellenwert überschritten ist). **Anmerkung:** Der Befehl REDISTRIBUTE löst die automatische Wörterverzeichniserstellung nur für neu hinzugefügte Datenbankpartitionen aus.

Tabelle 6. Unterschiede in der Erstellung von Komprimierungswörterverzeichnissen nach Release (Forts.)

Befehle und Attribute	Version 9.1	Version 9.5
Befehl REORG TABLE mit der Option **KEEPDICTIONARY**	Wenn Sie das Attribut COMPRESS der Tabelle auf YES gesetzt haben und noch kein Komprimierungswörterverzeichnis in der Tabelle vorhanden war, wurde versucht, unabhängig vom Volumen der Daten in der Tabelle ein Komprimierungswörterverzeichnis zu erstellen und in die Tabelle einzufügen bzw. an die Tabelle anzuhängen.	Es wird nur dann ein Wörterverzeichnis in eine Tabelle eingefügt, wenn die Größe der Tabelle dem Schwellenwert der automatischen Wörterverzeichniserstellung für die Tabellengröße entspricht und genügend Daten in der Tabelle vorhanden sind, wenn sie den Schwellenwert überschreitet.

Datenzeilenkomprimierung

Der Zweck der Datenzeilenkomprimierung besteht darin, Einsparungen beim benötigten Plattenspeicherplatz zu realisieren. Zudem kann sie auch zu Einsparungen beim Platten-E/A-Aufwand führen. Darüber hinaus können mehr Daten im Pufferpool zwischengespeichert werden, wodurch sich die Trefferquoten im Pufferpool erhöhen. Die Datenzeilenkomprimierung arbeitet mit einem statischen, auf einem Wörterverzeichnis basierenden Komprimierungsalgorithmus, um Daten zeilenweise zu komprimieren.

Durch die Komprimierung von Daten auf Zeilenebene können sich wiederholende Muster, die mehrere Spaltenwerte innerhalb einer Zeile umfassen, durch kürzere Symbolzeichenfolgen ersetzt werden.

Anmerkung: Dieses Verfahren ist mit einem höheren Aufwand an CPU-Zyklen verbunden, die für die Komprimierung und Dekomprimierung der Daten benötigt werden. Die Einsparungen an Speicher und die Auswirkungen auf die Leistung, die sich durch die Datenzeilenkomprimierung ergeben, sind eng mit den Merkmalen der Daten in der Datenbank, mit dem Layout und der Optimierung der Datenbank sowie mit der Anwendungsauslastung verbunden. Nur die Daten auf einer Datenseite oder in Protokollsätzen werden komprimiert.

Die Komprimierung von Tabellendaten erfolgt nur, wenn ein Komprimierungswörterverzeichnis (Compression Dictionary) für die Tabelle vorhanden ist, wenn Sie das Tabellenattribut COMPRESS in der Anweisung CREATE oder ALTER TABLE auf YES gesetzt haben und wenn genügend Daten in der Tabelle enthalten sind. Wenn diese Komprimierungsbedingungen für die Tabelle erfüllt sind, werden Daten, die Sie der Tabelle mit einer Anweisung INSERT bzw. einem Befehl LOAD INSERT, IMPORT INSERT oder REDISTRIBUTE hinzufügen, komprimiert.

In Version 9.5 wird die Datenzeilenkomprimierung automatisch aktiviert, wenn für eine Tabelle das Attribut COMPRESS auf YES gesetzt ist und das Datenkomprimierungswörterverzeichnis erstellt wurde. Wenn Sie eine Tabelle erstellt oder geändert und dabei das Attribut COMPRESS auf YES gesetzt haben, ist keine manuelle Operation oder Datenbankanforderung Ihrerseits erforderlich. Das heißt, Sie brauchen keine explizite klassische (d. h. offline ausgeführte) Tabellenreorganisation auszuführen, um das Datenkomprimierungswörterverzeichnis zu erstellen.

Anmerkung: Wenn Sie das Attribut COMPRESS auf YES setzen und bereits ein Komprimierungswörterverzeichnis vorhanden ist, wird die Komprimierung auf alle Zeileneinfügeoperationen, einschließlich Einfügungen durch eine IMPORT- oder LOAD-Operation, angewendet. Die Komprimierung wird jeweils für die gesamte Tabelle aktiviert. Die Zeilen werden jedoch einzeln komprimiert. Daher kann eine Tabelle gleichzeitig sowohl komprimierte als auch nicht komprimierte Zeilen enthalten.

Zur expliziten Erstellung eines Komprimierungswörterverzeichnisses (und zur nachfolgenden Komprimierung einer Tabelle) führen Sie eine klassische Tabellenreorganisation (d. h. eine Offlinetabellenreorganisation) aus. Alle Datenzeilen, die in einer Tabelle enthalten sind, werden bei der Erstellung des Komprimierungswörterverzeichnisses berücksichtigt. Das Wörterverzeichnis wird mit den Tabellendatenzeilen in den Datenobjektabschnitten der Tabelle gespeichert.

Zur Dekomprimierung einer Tabelle setzen Sie das Attribut COMPRESS auf den Wert NO und führen anschließend eine klassische Tabellenreorganisation (d. h. eine Offlinetabellenreorganisation) aus.

Einschränkungen

- Die Datenzeilenkomprimierung kann auf Indexobjekte sowie LONG-, LOB- und XML-Datenobjekte nicht angewendet werden.
- Die Zeilenkomprimierung ist nicht mit der Replikationsunterstützung für Tabellendaten kompatibel.
- Sie können mithilfe des Befehls RUNSTATS Statistikdaten zur Zeilenkomprimierung generieren. Diese werden in der Systemkatalogtabelle SYSCAT.TABLES gespeichert. Eine Schätzoption für die Komprimierung, die die Effektivität der Zeilenkomprimierung für eine Tabelle schätzt, wird mit dem Dienstprogramm INSPECT zur Verfügung gestellt. Das Abfrageoptimierungsprogramm berücksichtigt den Dekomprimierungsaufwand in seinem Modell zur Aufwandsberechnung.
- Abhängig von den Aktualisierungsaktivitäten und der Positionierung der Aktualisierungsänderungen innerhalb einer Datenzeile kann es zu einer höheren Protokollspeicherbelegung kommen.
- Wenn eine Zeile wächst, ist es möglich, dass die neue Version der Zeile nicht auf die aktuelle Datenseite passt. In diesem Fall wird das neue Image der Zeile auf einer Überlaufseite gespeichert. Um die Erstellung solcher Überlaufzeigerdatensätze zu minimieren, können Sie mehr freien Speicherbereich innerhalb einer Datenseite hinzufügen. Wenn ohne Komprimierung zum Beispiel 5 % freier Speicherbereich verwendet wurde, ordnen Sie mit Komprimierung 10 % freien Speicherbereich zu. Diese Empfehlung ist insbesondere für Daten wichtig, die in großem Umfang aktualisiert werden.

Automatische Statistikerfassung

Das DB2-Optimierungsprogramm verwendet Katalogstatistiken, um den effizientesten Zugriffsplan für eine gegebene Abfrage zu ermitteln. Wenn die Statistiken für eine Tabelle oder einen Index nicht auf dem neuesten Stand oder unvollständig sind, könnte das Optimierungsprogramm einen Plan auswählen, der nicht optimal ist, sodass die Abfrageausführung verlangsamt würde. Allerdings ist die Entscheidung, welche Statistiken für eine gegebene Auslastung zu erfassen sind, recht komplex und die Pflege aktueller Statistiken zeitaufwendig.

Mit der automatischen Statistikerfassung, die Teil der Funktion zur automatischen Tabellenverwaltung von DB2 ist, können Sie den DB2-Datenbankmanager bestim-

men lassen, ob Datenbankstatistiken aktualisiert werden müssen. Die automatische Statistikerfassung kann durch die Verwendung der Echtzeitstatistikfunktion (RTS, Real-Time Statistics) bei der Kompilierung von Anwendungen stattfinden oder durch die Ausführung des Dienstprogramms RUNSTATS im Hintergrund erfolgen. Die im Hintergrund ausgeführte Statistikerfassung kann aktiviert werden, wenn die Echtzeitstatistikerfassung inaktiviert ist. Die Statistikerfassung im Hintergrund muss aktiviert sein, damit die Echtzeitstatistikerfassung aktiviert werden kann. Die im Hintergrund ausgeführte automatische Statistikerfassung wird standardmäßig aktiviert, wenn Sie eine neue Datenbank erstellen. Die automatische Echtzeitstatistikerfassung wird durch den dynamischen Konfigurationsparameter **auto_stmt_stats** aktiviert.

Asynchron und in Echtzeit ausgeführte Statistikerfassung

Die automatische Statistikerfassung kann synchron oder asynchron mithilfe des Dienstprogramms RUNSTATS ausgeführt werden. Die *asynchrone* Erfassung findet im Hintergrund statt. Wenn die Echtzeitstatistikfunktion aktiviert ist, können Statistikdaten auch *synchron* bei der Kompilierung von Anweisungen erfasst werden. Wenn die Echtzeitstatistikerfassung aktiviert ist, können Statistikdaten auch mithilfe von Metadaten konstruiert und durch den Index- und Datenmanager gepflegt werden. *Konstruieren* bedeutet in diesem Fall, dass Statistiken abgeleitet oder generiert und nicht im Rahmen der normalen RUNSTATS-Aktivitäten erfasst werden. Zum Beispiel lässt sich die Anzahl von Zeilen in einer Tabelle aus der Kenntnis der Anzahl von Seiten in der Tabelle, der Seitengröße und der durchschnittlichen Zeilenbreite abgeleiten. In einigen Fällen werden die Statistikdaten jedoch nicht abgeleitet, sondern durch den Index- und den Datenmanager gepflegt, sodass sie direkt im Katalog gespeichert werden können. Zum Beispiel verwaltet der Indexmanager einen Zähler für die Blattseiten und Stufen in jedem Index.

Das Abfrageoptimierungsprogramm bestimmt auf der Basis des Bedarfs der Abfrage und des Volumens an Tabellenaktualisierungsaktivitäten, wie die Statistiken erfasst werden sollen. Die Tabellenaktualisierungsaktivität wird in der Anzahl der Aktualisierungs-, Einfüge- und Löschoperationen gemessen.

Die Echtzeitstatistikerfassung wird durch die Anforderungen einer SQL-Anweisung bestimmt, bevor diese optimiert wird. Dies sorgt für eine zeitgerechtere Statistikerfassung und für präzisere Statistikdaten. Präzise Statistikdaten können bessere Abfrageausführungspläne und eine höhere Leistung zur Folge haben. Wenn die Echtzeitstatistikerfassung nicht aktiviert ist, erfolgt die asynchrone Statistikerfassung in Intervallen von jeweils zwei Stunden. Dies ist möglicherweise nicht häufig genug, um präzise Statistiken für bestimmte Anwendungen bereitzustellen.

Wenn die Echtzeitstatistikerfassung aktiviert ist, erfolgt die asynchrone Statistikerfassungsprüfung trotzdem in Intervallen von jeweils zwei Stunden. Die Echtzeitstatistikerfassung leitet in folgenden Fällen auch asynchrone Erfassungsanforderungen ein:

- Wenn das Tabellenaktivitätsvolumen nicht ausreicht, um eine synchrone Erfassung zu erfordern, jedoch ausreicht, um eine asynchrone Erfassung zu erfordern.
- Wenn die synchrone Statistikerfassung mit Stichproben gearbeitet hat, weil die Tabelle sehr groß war.
- Wenn die synchronen Statistikdaten konstruiert wurden.
- Wenn die synchrone Statistikerfassung fehlgeschlagen ist, weil die Erfassungszeit überschritten wurde.

Es können höchstens zwei asynchrone Anforderungen gleichzeitig verarbeitet werden, jedoch nur für verschiedene Tabellen. Die eine Anforderung wird durch die Echtzeitstatistikerfassung, die andere durch die Überprüfung der asynchronen Statistikerfassung eingeleitet.

Die Leistungsbeeinträchtigung durch die automatische Statistikerfassung wird auf mehrere Arten minimiert:

- Die asynchrone Statistikerfassung erfolgt durch eine gedrosselte Ausführung des Dienstprogramms RUNSTATS. Die Drosselung begrenzt je nach aktueller Datenbankaktivität die Menge der Ressourcen, die vom Dienstprogramm RUNSTATS beansprucht werden: Wenn die Datenbankaktivitäten zunehmen, läuft das Dienstprogramm RUNSTATS langsamer, sodass es weniger Ressourcen benötigt.
- Die synchrone Statistikerfassung ist auf fünf Sekunden pro Abfrage begrenzt. Dieser Wert kann durch die RTS-Optimierungsrichtlinie gesteuert werden. Wenn die synchrone Erfassung das Zeitlimit überschreitet, wird eine Anforderung zur asynchronen Erfassung übergeben.
- Die synchrone Statistikerfassung speichert die Statistiken nicht im Systemkatalog. Stattdessen werden die Statistiken in einem Statistikcache gespeichert und später durch eine asynchrone Operation im Systemkatalog gespeichert. Dadurch werden Systemaufwand und mögliche Sperrenkonflikte vermieden, die mit dem Aktualisieren des Systemkatalogs verbunden sein können. Statistiken im Statistikcache sind für nachfolgende SQL-Kompilierungsanforderungen verfügbar.
- Pro Tabelle findet nur eine synchrone Statistikerfassungsoperation statt. Andere Agenten, die eine synchrone Statistikerfassung erfordern, konstruieren Statistikdaten, sofern möglich, und setzen die Anweisungskompilierung fort. Dieses Verhalten wird auch in einer Umgebung mit partitionierten Datenbanken realisiert, in der Agenten in verschiedenen Datenbankpartitionen möglicherweise synchrone Statistiken benötigen.
- Standardmäßig sind die bei synchronen und asynchronen Operationen erfassten Statistiken grundlegende Tabellenstatistiken mit Verteilungsinformationen und durch Stichprobenentnahme erstellte detaillierte Indexstatistiken. (Der Befehl RUNSTATS wird mit den Optionen WITH DISTRIBUTION und SAMPLED DETAILED INDEXES ALL ausgeführt.) Sie können den Typ der erfassten Statistiken anpassen, indem Sie die Statistikprofilerstellung aktivieren, sodass anhand von Informationen zu früheren Datenbankaktivitäten bestimmt wird, welche Statistiken für die Auslastung der Datenbank erforderlich sind. Sie können auch den Typ der erfassten Statistiken für eine bestimmte Tabelle anpassen, indem Sie ein eigenes Statistikprofil für diese Tabelle erstellen.
- Nur Tabellen mit fehlenden Statistiken oder hohen Graden an Aktivität (gemessen an der Anzahl der Aktualisierungs-, Einfüge- und Löschoperationen) werden für die Statistikerfassung in Betracht gezogen. Auch wenn die Tabelle die Bedingungen für die Statistikerfassung erfüllt, werden die synchronen Statistiken nicht erfasst, sofern sie nicht für die Abfrageoptimierung erforderlich sind. In einigen Fällen kann das Abfrageoptimierungsprogramm einen Plan ohne Statistiken auswählen.
- Bei der asynchronen Statistikerfassungsprüfung werden für große Tabellen (mit als 4000 Seiten) Stichproben erstellt, um festzustellen, ob die Statistiken durch intensive Tabellenaktivitäten geändert wurden. Statistiken für solch große Tabellen werden nur erfasst, wenn dies gerechtfertigt ist.
- Bei der asynchronen Statistikerfassung wird das Dienstprogramm RUNSTATS automatisch zur Ausführung während des optimalen Verwaltungsfensters terminiert, das in der Definition Ihrer Verwaltungsrichtlinie angegeben ist. Diese

Richtlinie gibt außerdem die Gruppe von Tabellen an, die zum Umfang der automatischen Statistikerfassung gehören, was zusätzlich eine unnötige Ressourcennutzung minimiert.

- Die synchrone Statistikerfassung und die Konstruktion von Statistikdaten richten sich nicht nach dem optimalen Verwaltungsfenster, das in der Definition Ihrer Verwaltungsrichtlinie angegeben ist, weil synchrone Anforderungen sofort ausgeführt werden müssen und nur über eine begrenzte Erfassungszeit verfügen. Die synchrone Statistikerfassung und die Konstruktion von Statistikdaten richten sich nach der Richtlinie, die die Gruppe von Tabellen angibt, die zum Umfang der automatischen Statistikerfassung gehören.
- Während der Ausführung der automatischen Statistikerfassung bleiben die betroffenen Tabellen weiterhin für reguläre Datenbankaktivitäten (Aktualisierungs-, Einfüge- und Löschoperationen) verfügbar.
- Für die asynchrone Statistikerfassung wird die gespeicherte Prozedur SYSPROC.NNSTAT mit der katalogbasierten Erfassungsmethode ausgeführt, um Statistiken zu Kurznamen automatisch zu aktualisieren. Echtzeitstatistiken (synchrone oder konstruierte) werden für Kurznamen nicht erfasst.

Die synchrone Echtzeitstatistikerfassung wird für reguläre Tabellen, MQTs (Materialized Query Tables) und deklarierte globale temporäre Tabellen (DGTTs) ausgeführt. Asynchrone Statistiken werden für deklarierte globale temporäre Tabellen (DGTTs) nicht erfasst. Dies bedeutet, dass die Verarbeitung der Echtzeitstatistikerfassung keine asynchronen Anforderungen für deklarierte globale temporäre Tabellen einleitet.

Die automatische Statistikerfassung (synchron oder asynchron) wird für folgende Elemente nicht ausgeführt:

- Statistische Sichten
- Tabellen, die als VOLATILE markiert sind (Tabellen, deren Feld VOLATILE in der Katalogsicht SYSCAT.TABLES definiert ist)
- Tabellen, deren Statistiken durch UPDATE-Anweisungen direkt in den SYSSTAT-Sichten manuell aktualisiert wurden

  Wenn Sie die Statistikdaten für Tabellen manuell modifizieren, geht der Datenbankmanager davon aus, dass Sie nun für die Verwaltung der Statistikdaten für diese Tabellen verantwortlich sind. Wenn der Datenbankmanager die Statistikdaten für eine Tabelle wieder berücksichtigen und pflegen soll, deren Statistikdaten manuell aktualisiert wurden, führen Sie den Befehl RUNSTATS zur Erfassung der Statistikdaten aus oder geben Sie bei der Verwendung des Befehls LOAD die Statistikerfassung an. Tabellen, die vor Version 9.5 erstellt wurden und deren Statistikdaten vor der Migration manuell aktualisiert wurden, sind davon nicht betroffen. Die Statistikdaten dieser Tabellen werden vom Datenbankmanager weiterhin automatisch gepflegt, bis Sie sie manuell aktualisieren.

Für folgende Elemente erfolgt keine Konstruktion von Statistikdaten:

- Statistische Sichten
- Tabellen, deren Statistiken durch UPDATE-Anweisungen direkt in den SYSSTAT-Katalogsichten manuell aktualisiert wurden. Beachten Sie, dass, wenn die Echtzeitstatistikerfassung nicht aktiviert ist, dennoch einige Statistiken für Tabellen konstruiert werden, deren Statistiken manuell aktualisiert wurden.

In einer Umgebung mit partitionierten Datenbanken werden die Statistikdaten in nur einer Datenbankpartition erfasst und extrapoliert. Der Datenbankmanager

erfasst Statistikdaten (sowohl synchrone als auch asynchrone) stets in der ersten Datenbankpartition der Datenbankpartitionsgruppe.

Aktivitäten zur Echtzeitstatistikerfassung finden nicht vor Ablauf der ersten fünf Minuten nach der Datenbankaktivierung statt.

Wenn die Echtzeitstatistikerfassung aktiviert ist, sollten Sie ein definiertes Verwaltungsfenster terminieren. Standardmäßig ist das Verwaltungsfenster nicht definiert. Wenn kein definiertes Verwaltungsfenster vorhanden ist, wird nur die synchrone Statistikerfassung ausgeführt. In diesem Fall befinden sich die erfassten Statistiken nur im Arbeitsspeicher und werden in der Regel durch Stichprobenentnahme (außer bei kleinen Tabellen) erfasst.

Sowohl für statisches als auch für dynamisches SQL wird eine Verarbeitung der Echtzeitstatistiken ausgeführt.

Eine Tabelle, die durch die Verwendung des Befehls IMPORT abgeschnitten wurde, wird automatisch als Tabelle mit veralteten Statistiken erkannt.

Eine automatische Statistikerfassung (synchron und asynchron) macht im Cache gespeicherte dynamische Anweisungen, die auf Tabellen verweisen, für die Statistiken erfasst wurden, ungültig. Dies geschieht, damit im Cache gespeicherte dynamische Anweisungen mit den aktuellsten Statistiken reoptimiert werden können.

Echtzeitstatistiken und EXPLAIN-Verarbeitung

Es erfolgt keine Echtzeitverarbeitung für eine Abfrage, die nur durch die EXPLAIN-Einrichtung bearbeitet, jedoch nicht ausgeführt wird. In der folgenden Tabelle sind die Verhaltensweisen für die verschiedenen Werte des Sonderregisters CURRENT EXPLAIN MODE zusammengefasst.

Tabelle 7. Echtzeitstatistikerfassung als Funktion des Werts des Sonderregisters CURRENT EXPLAIN MODE

Wert in CURRENT EXPLAIN MODE	Echtzeitstatistikerfassung erfolgt?
YES	Ja
EXPLAIN	Nein
NO	Ja
REOPT	Ja
RECOMMEND INDEXES	Nein
EVALUATE INDEXES	Nein

Automatische Statistikerfassung und Statistikcache

In DB2 Version 9.5 wurde ein Statistikcache eingeführt, um synchron erfasste Statistikdaten für alle Abfragen verfügbar zu machen. Dieser Cache ist Teil des Katalogcache. In einer Umgebung mit partitionierten Datenbanken befindet sich dieser Cache nur in der Katalogdatenbankpartition. Der Katalogcache kann mehrere Einträge für dasselbe SYSTABLES-Objekt speichern, wodurch sich das Volumen des Katalogcache in allen Datenbankpartitionen erhöht. Ziehen Sie in Betracht, den Wert des Datenbankkonfigurationsparameters **catalogcache_sz** zu erhöhen, wenn die Echtzeitstatistikerfassung aktiviert ist.

Seit DB2 Version 9 können Sie den Konfigurationsadvisor dazu verwenden, die Anfangskonfiguration für neue Datenbanken zu bestimmen. Der Konfigurationsadvisor empfiehlt, den Datenbankkonfigurationsparameter **auto_stmt_stats** auf den Wert ON zu setzen.

Automatische Statistikerfassung und Statistikprofile

Synchrone und asynchrone Statistiken werden entsprechend einem Statistikprofil erfasst, das für eine Tabelle in Kraft ist. Von dieser Regel gibt es folgende Ausnahmen:

- Zur Minimierung des Aufwands für die synchrone Statistikerfassung kann der Datenbankmanager Statistikdaten durch Stichprobenentnahme erfassen. In diesem Fall können die Stichprobenrate und die Methode von denen abweichen, die im Statistikprofil angegeben sind.
- Es ist möglich, dass die synchrone Statistikerfassung Statistiken konstruiert, obwohl es vielleicht nicht möglich ist, alle im Statistikprofil angegebenen Statistiken zu konstruieren. Zum Beispiel können Spaltenstatistiken wie COLCARD, HIGH2KEY und LOW2KEY nicht konstruiert werden, wenn die Spalte nicht an führender Position in einem Index enthalten ist.

Wenn die synchrone Statistikerfassung nicht alle Statistiken erfassen kann, die im Statistikprofil angegeben sind, wird eine Anforderung zur asynchronen Statistikerfassung übergeben.

Obwohl die Echtzeitstatistikerfassung mit dem Ziel entwickelt wurde, den Aufwand für die Statistikerfassung zu minimieren, sollten Sie sie zunächst in einer Testumgebung erproben, um sicherzustellen, dass es zu keinen Leistungseinbußen kommt. Dies ist in einigen OLTP-Szenarios (OLTP - Onlinetransaktionsverarbeitung) möglich, insbesondere wenn die Ausführungsdauer einer Abfrage einer oberen Begrenzung unterliegt.

Aktivieren der automatischen Statistikerfassung

Über genaue und vollständige Datenbankstatistiken zu verfügen, ist von kritischer Bedeutung für einen effizienten Datenzugriff und eine optimale Auslastungsleistung. Verwenden Sie die Funktion zur automatischen Statistikerfassung der Funktionalität zur automatischen Tabellenverwaltung, um relevante Datenbankstatistiken zu aktualisieren und zu pflegen. In Umgebungen, in denen eine einzelne Datenbankpartition auf einem Einzelprozessor betrieben wird (serielle Umgebung), können Sie diese Funktionalität optional noch erweitern, indem Sie Abfragedaten sammeln und Statistikprofile generieren, die DB2 bei der automatischen Erfassung der exakten Gruppe der für Ihre Auslastung erforderlichen Statistiken unterstützen. Diese Option ist in MPP-Umgebungen, bestimmten Umgebungen mit föderierten Datenbanken oder Umgebungen mit aktivierter partitionsinterner Parallelität nicht verfügbar.

Gehen Sie wie folgt vor, um die automatische Erfassung von Statistiken zu aktivieren:

1. Konfigurieren Sie Ihre Datenbankinstanz, indem Sie den Assistenten 'Automatische Verwaltung konfigurieren' oder die Befehlszeile verwenden:
   - Über den Assistenten 'Automatische Verwaltung konfigurieren':
     a. Öffnen Sie den Assistenten entweder über die Steuerzentrale, indem Sie ein Datenbankobjekt mit der rechten Maustaste anklicken, oder über die Diagnosezentrale, indem Sie eine Datenbankinstanz mit der rechten Maustaste anklicken.

b. Wählen Sie **Automatische Verwaltung konfigurieren** im Kontextmenü aus. In diesem Assistenten können Sie die automatische Statistikerfassung aktivieren, die Tabellen angeben, aus denen die Statistiken automatisch zu erfassen sind, und ein Verwaltungsfenster für die Ausführung des Dienstprogramms RUNSTATS definieren.

- Bei Verwendung der Befehlszeile setzen Sie jeden der folgenden Konfigurationsparameter auf den Wert ON:
  - **AUTO_MAINT**
  - **AUTO_TBL_MAINT**
  - **AUTO_RUNSTATS**

2. Optional: Setzen Sie zur Aktivierung der automatischen Statistikprofilerstellung die folgenden beiden Konfigurationsparameter auf ON:
   - **AUTO_STATS_PROF**
   - **AUTO_PROF_UPD**
3. Optional: Setzen Sie zur Aktivierung der Echtzeitstatistikerfassung den Konfigurationsparameter **AUTO_STMT_STATS** auf den Wert ON. Wenn dieser Konfigurationsparameter auf den Wert ON gesetzt ist, werden Tabellenstatistiken automatisch bei der Kompilierung von Anweisungen erfasst, wenn sie zur Optimierung einer Abfrage erforderlich sind.

Konfigurationsadvisor

Mithilfe des Konfigurationsadvisors können Sie Empfehlungen für die Anfangswerte der Pufferpoolgröße, der Datenbankkonfigurationsparameter und der Konfigurationsparameter des Datenbankmanagers abrufen.

Zur Verwendung des Konfigurationsadvisors geben Sie den Befehl AUTOCONFIGURE für eine vorhandene Datenbank oder beim Erstellen einer Datenbank die Option AUTOCONFIGURE im Befehl CREATE DATABASE an. Zur Konfiguration Ihrer Datenbank müssen Sie über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT verfügen.

Sie können die empfohlenen Werte anzeigen oder anwenden, indem Sie die Option APPLY des Befehls CREATE DATABASE verwenden. Die Empfehlungen basieren auf Ihrer Eingabe sowie auf Systeminformationen, die von der Advisorfunktion erfasst werden.

Die vom Konfigurationsadvisor vorgeschlagenen Werte sind nur für eine Datenbank pro Instanz relevant. Wenn Sie diese Advisorfunktion für mehrere Datenbanken verwenden möchten, muss jede Datenbank zu einer separaten Instanz gehören.

Optimieren von Konfigurationsparametern mit dem Konfigurationsadvisor

Der Konfigurationsadvisor hilft Ihnen bei der Optimierung der Leistung und der ausgewogenen Verteilung des Speicherbedarfs für eine einzelne Datenbank pro Instanz, indem er Konfigurationsparameter zur Änderung vorschlägt und geeignete Werte für sie empfiehlt. Der Konfigurationsadvisor wird automatisch ausgeführt, wenn Sie eine Datenbank erstellen.

Zur Inaktivierung dieser Funktion oder zur expliziten Aktivierung dieser Funktion verwenden Sie den Befehl db2set wie folgt, bevor Sie eine Datenbank erstellen:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

Zum Definieren von Werten für verschiedene Konfigurationsparameter und zum Bestimmen des Anwendungsbereichs dieser Parameter verwenden Sie den Befehl AUTOCONFIGURE, indem Sie eine der folgenden Optionen angeben:

- NONE: Bedeutet, dass keiner der Werte angewendet wird.
- DB ONLY: Bedeutet, dass nur Werte für die Datenbankkonfiguration und Pufferpools angewendet werden.
- DB AND DBM: Bedeutet, dass alle Parameter und ihre Werte angewendet werden.

Anmerkung: Auch wenn der Konfigurationsadvisor bei der Ausführung des Befehls CREATE DATABASE automatisch aktiviert ist, können Sie dennoch die Optionen für den Befehl AUTOCONFIGURE angeben. Wenn Sie den Konfigurationsadvisor bei der Ausführung des Befehls CREATE DATABASE nicht aktiviert haben, können Sie den Konfigurationsadvisor anschließend manuell ausführen.

Generieren von Konfigurationsempfehlungen

Der Konfigurationsadvisor wird automatisch ausgeführt, wenn Sie eine Datenbank erstellen. Sie können den Konfigurationsadvisor auch ausführen, indem Sie den Befehl AUTOCONFIGURE in den Befehlszeilenprozessor (CLP) eingeben oder die API db2AutoConfig aufrufen.

Zur Anforderung von Konfigurationsempfehlungen über den Befehlszeilenprozessor geben Sie den folgenden Befehl ein:

```
AUTOCONFIGURE
     USING eingabeschlüsselwort parameterwert
     APPLY wert
```

Das folgende Beispiel zeigt einen Befehl AUTOCONFIGURE, mit dem Konfigurationsempfehlungen auf der Basis von Eingaben über die Nutzung der Datenbank angefordert werden, ohne die Empfehlungen jedoch anzuwenden:

```
DB2 AUTOCONFIGURE USING
     MEM_PERCENT 60
     WORKLOAD_TYPE MIXED
     NUM_STMTS 500
     ADMIN_PRIORITY BOTH
     IS_POPULATED YES
     NUM_LOCAL_APPS 0
     NUM_REMOTE_APPS 20
     ISOLATION RR
     BP_RESIZEABLE YES
APPLY NONE
```

Beispiel: Anfordern von Konfigurationsempfehlungen mit dem Konfigurationsadvisor

Dieses Szenario veranschaulicht, wie der Konfigurationsadvisor über die Befehlszeile ausgeführt wird, um Empfehlungen zu generieren, und zeigt eine vom Konfigurationsadvisor generierte Beispielausgabe.

Gehen Sie wie folgt vor, um den Konfigurationsadvisor auszuführen:

1. Stellen Sie eine Verbindung zur Datenbank PERSONL her, indem Sie den folgenden Befehl in die Befehlszeile eingeben:

   ```
   DB2 CONNECT TO PERSONL
   ```

2. Führen Sie den Befehl AUTOCONFIGURE über die Befehlszeile (CLP) mit Angaben zur Art und Weise der Verwendung der Datenbank aus. Wie im folgenden Beispiel gezeigt, können Sie die Option **APPLY** auf den Wert NONE setzen, um anzugeben, dass Sie die Konfigurationsempfehlungen nur prüfen, jedoch nicht anwenden möchten:

```
DB2 AUTOCONFIGURE USING
     MEM_PERCENT 60
     WORKLOAD_TYPE MIXED
     NUM_STMTS 500
     ADMIN_PRIORITY BOTH
     IS_POPULATED YES
     NUM_LOCAL_APPS 0
     NUM_REMOTE_APPS 20
     ISOLATION RR
     BP_RESIZEABLE YES
APPLY NONE
```

   Wenn Sie sich über den Wert eines Parameters für den Befehl nicht im Klaren sind, können Sie ihn auslassen, sodass der Standardwert verwendet wird. Sie können bis zu 10 Parameter ohne Werte übergeben: MEM_PERCENT, WORKLOAD_TYPE usw., wie im vorigen Beispiel gezeigt.

Die durch den Befehl AUTOCONFIGURE generierten Empfehlungen werden auf dem Bildschirm im Tabellenformat angezeigt:

Tabelle 8. Beispielausgabe des Konfigurationsadvisors: Teil 1

```
Frühere und angewendete Werte für die Datenbankmanagerkonfiguration
Beschreibung                                          Parameter   Aktueller Wert  Empfohlener Wert
-------------------------------------------------------------------------------------------------
Zwischenspeicher für Anw.unterstützungsebene (4 KB)      (ASLHEAPSZ) = 15              15
Anzahl Kommunikationspuffer (4 KB)                (FCM_NUM_BUFFERS) = AUTOMATIC       AUTOMATIC
Partitionsinterne Parallelität aktivieren          (INTRA_PARALLEL) = NO              NO
Max. Grad der Parallelität bei Abfragen           (MAX_QUERYDEGREE) = ANY             1
Größe des Agentenpools                             (NUM_POOLAGENTS) = 100(berechnet)  200
Ursprüngliche Anzahl Agenten im Pool               (NUM_INITAGENTS) = 0               0
Max. E/A-Blockgröße (Byte) für Requester                 (RQRIOBLK) = 32767           32767
Schwellenwert für Sortierspeicher (4 KB)               (SHEAPTHRES) = 0               0
```

Tabelle 9. Beispielausgabe des Konfigurationsadvisors (Fortsetzung)

```
Frühere und angewendete Werte für die Datenbankkonfiguration                                  Empf.
Beschreibung                                                    Parameter   Aktueller Wert  Wert
-------------------------------------------------------------------------------------------------
Standardzwischenspeicher für Anwendungen (4 KB)                (APPLHEAPSZ) = 256             256
Katalogcachegröße (4 KB)                                     (CATALOGCACHE_SZ) = (MAXAPPLS*4)    260
Schwellenwert für geänderte Seiten                            (CHNGPGS_THRESH) = 60              80
Zwischenspeicher der Datenbank (4 KB)                                (DBHEAP) = 1200            2791
Grad der Parallelität                                            (DFT_DEGREE) = 1               1
Standardwert für Speicherbereichsgröße (Seiten)                (DFT_EXTENT_SZ) = 32              32
Standardwert für Vorablesezugriffsgröße (Seiten)             (DFT_PREFETCH_SZ) = AUTOMATIC       AUTOMATIC
Standardabfrageoptimierungsklasse                               (DFT_QUERYOPT) = 5               5
Max. Speicher für Sperrenliste (4 KB)                            (LOCKLIST) = 100             AUTOMATIC
Protokollpuffergröße (4 KB)                                      (LOGBUFSZ) = 8               99
Protokolldateigröße (4 KB)                                      (LOGFILSIZ) = 1000            1024
Anzahl primärer Protokolldateien                                (LOGPRIMARY) = 3               8
Anzahl sekundärer Protokolldateien                               (LOGSECOND) = 2               3
Max. Anzahl aktiver Anwendungen                                   (MAXAPPLS) = AUTOMATIC       AUTOMATIC
Anzahl der Sperrenlisten pro Anwend. (in %)                       (MAXLOCKS) = 10              AUTOMATIC
Anzahl Gruppencommits                                            (MINCOMMIT) = 1               1
Anzahl asynchroner Seitenlöschfunktionen                      (NUM_IOCLEANERS) = 1             1
Anzahl von E/A-Servern                                         (NUM_IOSERVERS) = 3             4
Größe des Paketcache  (4 KB)                                    (PCKCACHESZ) = (MAXAPPLS*8)    1533
Vor bedingtem Prüfpunkt zu schreibende Protokollsätze (%)          (SOFTMAX) = 100             320
Zwischenspeicher für Sortierlisten (4 KB)                        (SORTHEAP) = 256             AUTOMATIC
Anweisungszwischenspeicher (4KB)                                 (STMTHEAP) = 4096            4096
Größe des Statistikzwischenspeichers (4 KB)                    (STAT_HEAP_SZ) = 4384          4384
Zwischenspeicher für Dienstprogramme (4 KB)                    (UTIL_HEAP_SZ) = 5000          113661
Speicher mit automatischer Leistungsoptimierung              (SELF_TUNING_MEM) = ON            ON
    Automatische Statistikerstellung                          (AUTO_RUNSTATS) = ON            ON
Sortierspeicherschwelle für gemeinsame Sortierungen (4 KB) (SHEAPTHRES_SHR) = 5000            AUTOMATIC
```

Tabelle 10. Beispielausgabe des Konfigurationsadvisors (Fortsetzung)

```
Frühere und angewendete Werte für Pufferpool(s)
Beschreibung                                        Parameter   Aktueller Wert  Empfohlener Wert
-------------------------------------------------------------------------------------------------
IBMDEFAULTBP                                  Pufferpoolgröße = -2              340985

DB210203I AUTOCONFIGURE wurde erfolgreich beendet. Die Werte für den Datenbankmanager
oder die Datenbankkonfiguration wurden möglicherweise geändert. Die Instanz muss neu
gestartet werden, damit die Änderungen wirksam werden. Außerdem sollten Sie für die Pakete
einen Rebind durchführen, sobald die neuen Konfigurationsparameter wirksam sind.
```

Wenn Sie allen Empfehlungen zustimmen, können Sie den Befehl AUTOCONFIGURE entweder erneut ausführen und dabei über die Option APPLY angeben, dass die empfohlenen Werte angewendet werden sollen, oder einzelne Konfigurationsparameter mit dem Befehl UPDATE DATABASE MANAGER CONFIGURATION und dem Befehl UPDATE DATABASE CONFIGURATION aktualisieren.

Drosselung von Dienstprogrammen

Die Dienstprogrammdrosselung reguliert die Beeinträchtigung der Leistung durch Wartungsdienstprogramme, sodass sie gleichzeitig innerhalb von Produktionszeiträumen ausgeführt werden können. Obwohl die Richtlinie für die Auslastungswirkung (eine Einstellung, die die Ausführung von Dienstprogrammen in einem gedrosselten Modus ermöglicht) standardmäßig definiert wird, müssen Sie die Priorität der Auslastungswirkung festlegen. Dies ist eine Einstellung jeder Bereinigungsfunktion, die die Drosselungspriorität angibt, wenn Sie ein Dienstprogramm ausführen, das gedrosselt werden soll.

Das Drosselungssystem stellt sicher, dass die gedrosselten Dienstprogramme so häufig wie möglich ausgeführt werden, ohne gegen die Richtlinie für die Auslastungswirkung zu verstoßen. Sie können Operationen zur Statistikerfassung, Backup-Operationen, Datenumverteilungen und asynchrone Indexbereinigungen drosseln.

Sie definieren die Richtlinie für die Auslastungswirkung mithilfe des Konfigurationsparameters **util_impact_lim**.

Bereinigungsfunktionen sind in die Drosselungseinrichtung für Dienstprogramme integriert. Standardmäßig ist jede Bereinigungsfunktion (für Indizes) auf den Wert 50 für die Priorität der Auslastungswirkung von Dienstprogrammen eingestellt. (Akzeptable Werte liegen zwischen 1 und 100, wobei 0 keine Drosselung angibt.) Sie können die Priorität mit dem Befehl SET UTIL_IMPACT_PRIORITY oder mit der API 'db2UtilityControl' ändern.

Asynchrone Indexbereinigung

Als asynchrone Indexbereinigung (AIC, Asynchronous Index Cleanup) wird die verzögerte Bereinigung von Indizes nach Operationen bezeichnet, die Indexeinträge ungültig machen. Abhängig vom Typ des Indexes können die Einträge Satz-IDs (RIDs) oder Block-IDs (BIDs) sein. In beiden Fällen werden diese Einträge durch die Indexbereinigungsfunktionen entfernt, die asynchron im Hintergrund ausgeführt werden.

Die AIC beschleunigt das Aufheben der Zuordnung einer Datenpartition zu einer partitionierten Tabelle. Wenn die partitionierte Tabelle einen oder mehrere nicht partitionierte Indizes enthält, wird die AIC eingeleitet. In diesem Fall entfernt die AIC alle nicht partitionierten Indexeinträge, die sich auf die Datenpartition mit aufgehobener Zuordnung beziehen, und außerdem alle pseudo-gelöschten Einträge. Wenn alle Indizes bereinigt sind, wird die Kennung (ID), die zu der Datenpartition mit aufgehobener Zuordnung gehört, aus dem Systemkatalog entfernt.

Anmerkung: Wenn für die partitionierte Tabelle abhängige MQTs (Materialized Query Tables, gespeicherte Abfragetabellen) definiert sind, wird die AIC erst nach der Ausführung einer SET INTEGRITY-Operation eingeleitet.

Während der Ausführung der AIC bleibt der normale Tabellenzugriff erhalten. Abfragen, die auf die Indizes zugreifen, ignorieren alle nicht gültigen Einträge, die noch nicht bereinigt wurden.

In den meisten Fällen wird eine Bereinigungsfunktion für jeden nicht partitionierten Index gestartet, der der partitionierten Tabelle zugeordnet ist. Ein interner Taskverteilungsdämonprozess ist für die Verteilung der AIC-Tasks an die jeweiligen Datenpartitionen sowie für die Zuweisung von Datenbankagenten zuständig.

Der Verteilungsdämon und die Bereinigungsagenten sind interne Systemanwendungen. Sie werden in der Ausgabe des Befehls LIST APPLICATION mit den Anwendungsnamen **db2taskd** bzw. **db2aic** aufgeführt. Zur Vermeidung versehentlicher Unterbrechungen lässt sich der Abbruch von Systemanwendungen nicht erzwingen. Der Verteilungsdämon bleibt online, solange die Datenbank aktiv ist. Die Bereinigungsfunktionen bleiben aktiv, bis die Bereinigung abgeschlossen ist. Falls die Datenbank während der Bereinigung inaktiviert wird, nimmt die AIC die Arbeit wieder auf, wenn Sie die Datenbank reaktivieren.

Leistung

Die AIC wirkt sich nur minimal auf die Leistung aus.

Ein sofortiger Zeilensperrentest ist erforderlich, um festzustellen, ob ein pseudogelöschter Eintrag festgeschrieben wurde. Da die Sperre jedoch nie aktiviert wird, wird der gemeinsame Zugriff nicht beeinträchtigt.

Jede Bereinigungsfunktion aktiviert eine minimale Tabellenbereichssperre (IX) und eine Tabellensperre (IS). Diese Sperren werden freigegeben, wenn die Bereinigungsfunktion feststellt, dass andere Anwendungen auf die Sperren warten. Wenn dieser Fall eintritt, setzt die Bereinigungsfunktion die Verarbeitung für fünf Minuten aus.

Bereinigungsfunktionen sind in die Drosselungseinrichtung für Dienstprogramme integriert. Standardmäßig ist für jede Bereinigungsfunktion der Wert 50 für die Priorität der Auslastungswirkung von Dienstprogrammen eingestellt. Sie können die Priorität mit dem Befehl SET UTIL_IMPACT_PRIORITY oder mit der API db2UtilityControl ändern.

Überwachung

Sie können die AIC mit dem Befehl LIST UTILITIES überwachen. Jede Indexbereinigungsfunktion wird im Monitor als separates Dienstprogramm aufgeführt.

Das folgende Beispiel veranschaulicht, wie die AIC-Aktivität in der Datenbank WSDB in der aktuellen Datenbankpartition über die Schnittstelle des Befehlszeilenprozessors (CLP) angezeigt wird:

```
$ db2 list utilities show detail

ID                                = 2
Typ                               = ASYNCHRONOUS INDEX CLEANUP
Datenbankname                     = WSDB
Partitionsnummer                  = 0
Beschreibung                      = Tabelle: USER1.SALES, Index: USER1.I2
Startzeit                         = 2005-12-15 11:15:01.967939
Status                            = Wird ausgeführt
Aufruftyp                         = Automatisch
Drosselung:
   Priorität                      = 50
Fortschrittsüberwachung:
      Gesamte Arbeit              = 5 Seiten
      Abgeschlossene Arbeit       = 0 Seiten
      Startzeit                   = 2005-12-15 11:15:01.979033

ID                                = 1
Typ                               = ASYNCHRONOUS INDEX CLEANUP
Datenbankname                     = WSDB
Partitionsnummer                  = 0
Beschreibung                      = Tabelle: USER1.SALES, Index: USER1.I1
Startzeit                         = 2005-12-15 11:15:01.978554
Status                            = Wird ausgeführt
```

```
Aufruftyp                       = Automatisch
Drosselung:
   Priorität                    = 50
Fortschrittsüberwachung:
      Gesamte Arbeit            = 5 Seiten
      Abgeschlossene Arbeit     = 0 Seiten
      Startzeit                 = 2005-12-15 11:15:01.980524
```

In gezeigten Fall sind zwei Bereinigungsfunktionen an der Tabelle USERS1.SALES aktiv. Eine Bereinigungsfunktion bearbeitet den Index I1, die andere den Index I2. Dem Abschnitt unter 'Fortschrittsüberwachung' ist die geschätzte Gesamtzahl von zu bereinigenden Indexseiten sowie die aktuelle Anzahl der bereits bereinigten Indexseiten zu entnehmen.

Das Feld `Status` gibt den aktuellen Status der Bereinigungsfunktion an. Normalerweise lautet der Status 'Wird ausgeführt'. Die Bereinigungsfunktion kann 'Im Wartestatus' sein, wenn sie darauf wartet, einem verfügbaren Datenbankagenten zugeordnet zu werden oder wenn sie wegen eines Sperrenkonflikts vorübergehend ausgesetzt ist.

Anmerkung: Verschiedene Tasks in verschiedenen Datenbankpartitionen können die gleiche Dienstprogramm-ID haben, da den Tasks in jeder Datenbankpartition IDs nur für diese Datenbankpartition zuordnet werden.

Asynchrone Indexbereinigung für MDC-Tabellen

Sie können die Leistung einer Rolloutlöschung durch die Nutzung einer asynchronen Indexbereinigung verbessern. Eine Rolloutlöschung ist eine effiziente Methode zum Löschen von Datenblöcken, die bestimmten Kriterien entsprechen, aus MDC-Tabellen (MDC - mehrdimensionales Clustering). Die asynchrone Indexbereinigung ist die verzögerte Bereinigung von Indizes, die im Anschluss an Operationen erfolgt, durch die Indexeinträge ungültig werden.

Während einer Standardrolloutlöschung werden Indizes mit der Löschoperation synchron bereinigt. Für Tabellen, die viele Satz-ID-Indizes (RID-Indizes) besitzen, wird ein beträchtlicher Teil des Löschzeitaufwands zum Entfernen von Indexschlüsseln verwendet, die auf die Tabellenzeilen verweisen, die gelöscht werden. Sie können den Rollout beschleunigen, indem Sie angeben, dass diese Indizes nach dem Festschreiben der Löschoperation zu bereinigen sind.

Zur Nutzung der asynchronen Indexbereinigung für MDC-Tabellen müssen Sie den Mechanismus für den *Rollout mit verzögerter Indexbereinigung* explizit aktivieren. Ein Rollout mit verzögerter Indexbereinigung kann durch zwei Methoden angegeben werden: Durch Setzen der Registrierdatenbankvariablen **DB2_MDC_ROLLOUT** auf den Wert DEFER und durch Ausführen der Anweisung SET CURRENT MDC ROLLOUT MODE. Während des Rollouts mit verzögerter Indexbereinigung werden Blöcke als durch Rollout gelöscht markiert, wobei die Aktualisierung an den Satz-ID-Indizes erst erfolgt, wenn die Transaktion festgeschrieben wurde. Block-ID-Indizes (BID-Indizes) werden weiterhin während der Löschoperation bereinigt, weil sie keine Verarbeitung auf Zeilenebene erfordern.

Die asynchrone Indexbereinigung für ein Rollout wird aufgerufen, wenn eine Rolloutlöschung festgeschrieben wurde oder, falls die Datenbank beendet wurde, wenn auf die Tabelle nach dem Neustart der Datenbank zum ersten Mal zugegriffen wird. Während der Ausführung der asynchronen Indexbereinigung bleiben Abfragen, die auf die Indizes, einschließlich des Index, der gerade bereinigt wird, zugreifen, funktionsfähig.

Pro MDC-Tabelle ist eine koordinierende Bereinigungsfunktion aktiv. Die Indexbereinigung für mehrere Rollouts wird in der Bereinigungsfunktion konsolidiert. Die Bereinigungsfunktion startet einen Bereinigungsagenten für jeden Satz-ID-Index, und die Bereinigungsagenten aktualisieren die Satz-ID-Indizes parallel. Bereinigungsfunktionen sind darüber hinaus in die Drosselungseinrichtung für Dienstprogramme integriert. Standardmäßig ist jede Bereinigungsfunktion auf den Wert 50 für die Priorität der Auslastungswirkung von Dienstprogrammen eingestellt. (Akzeptable Werte liegen zwischen 1 und 100, wobei 0 keine Drosselung angibt.) Sie können die Priorität mit dem Befehl SET UTIL_IMPACT_PRIORITY oder mit der API db2UtilityControl ändern.

Überwachung

Da die durch Rollout gelöschten Blöcke in einer MDC-Tabelle erst wieder verwendet werden können, wenn die Bereinigung abgeschlossen ist, ist es nützlich, den Fortschritt eines Rollouts mit verzögerter Indexbereinigung zu überwachen. Mit dem Überwachungsbefehl LIST UTILITIES können Sie einen Dienstprogrammüberwachungseintrag für jeden Index anzeigen, der momentan bereinigt wird. Sie können mit der Tabellenfunktion SYSPROC.ADMIN_GET_TAB_INFO_V95 auch die Anzahl der Blöcke in der Tabelle abfragen, die gerade durch einen Rollout mit verzögerter Indexbereinigung bereinigt werden (BLOCKS_PENDING_CLEANUP). Verwenden Sie zum Abfragen der Anzahl von MDC-Tabellenblöcke mit anstehender Bereinigung auf Datenbankebene den Befehl GET SNAPSHOT.

In der folgenden Beispielausgabe für den Befehl LIST UTILITIES wird der Fortschritt durch die Anzahl von Seiten in jedem Index angezeigt, die bereinigt wurden. Jede in der Ausgabe aufgeführte Phase stellt einen der Satz-ID-Indizes dar, die für die Tabelle bereinigt werden.

```
Ausgabe des Befehls 'db2 LIST UTILITIES SHOW DETAILS'
ID                             = 2
Typ                            = MDC ROLLOUT INDEX CLEANUP
Datenbankname                  = WSDB
Partitionsnummer               = 0
Beschreibung                   = TABLE.<schemaname>.<tabellenname>
Startzeit                      = 06-12-2006 08:56:33.390158
Status                         = Wird ausgeführt
Aufruftyp                      = Automatisch
Drosselung:
   Priorität                   = 50
Fortschrittsüberwachung:
  Geschätzte Fertigstellung (%) = 83
   Phasennummer                = 1
      Beschreibung             = <schemaname>.<indexname>
      Gesamte Arbeit           = 13 Seiten
      Abgeschlossene Arbeit    = 13 Seiten
      Startzeit                = 06-12-2006 08:56:33.391566
   Phasennummer                = 2
      Beschreibung             = <schemaname>.<indexname>
      Gesamte Arbeit           = 13 Seiten
      Abgeschlossene Arbeit    = 13 Seiten
      Startzeit                = 06-12-2006 08:56:33.391577
   Phasennummer                = 3
      Beschreibung             = <schemaname>.<indexname>
      Gesamte Arbeit           = 9 Seiten
      Abgeschlossene Arbeit    = 3 Seiten
      Startzeit                = 06-12-2006 08:56:33.391587
```

Kapitel 4. Instanzen

Eine *Instanz* ist eine logische Datenbankmanagerumgebung, in der Sie Datenbanken katalogisieren und Konfigurationsparameter definieren. Bei Bedarf können Sie mehrere Instanzen auf demselben physischen Server erstellen und für jede Instanz eine spezifische Datenbankserverumgebung bereitstellen.

Anmerkung: Für Nichtrootinstallationen unter Linux und UNIX-Betriebssystemen wird nur eine Instanz bei der Installation Ihres DB2-Produkts erstellt. Es können keine weiteren Instanzen erstellt werden.

Mehrere Instanzen können beispielsweise zu folgenden Zwecken wünschenswert sein:

- Betreiben einer Instanz für die Entwicklungsumgebung und einer weiteren für die Geschäftsumgebung.
- Anpassen einer Instanz für eine bestimmte Umgebung.
- Begrenzen des Zugriffs auf sensible Daten.
- Steuern der Erteilung der Berechtigung SYSADM, SYSCTRL und SYSMAINT für jede Instanz.
- Optimieren der Datenbankmanagerkonfiguration für jede Instanz.
- Begrenzen der durch den Ausfall einer Instanz entstehenden Auswirkungen. Im Fall eines Ausfalls ist jeweils nur eine Instanz betroffen. Die übrigen Instanzen funktionieren weiterhin ordnungsgemäß.

Mehrere Instanzen stellen zusätzliche Anforderungen:

- Für jede Instanz sind zusätzliche Systemressourcen (virtueller Speicher und Plattenspeicherplatz) erforderlich.
- Der Verwaltungsaufwand ist größer, da mehrere Instanzen zu verwalten sind.

Im Instanzverzeichnis werden alle Informationen für eine Datenbankinstanz gespeichert. Sobald das Instanzverzeichnis erstellt ist, können Sie dessen Position nicht mehr ändern. Das Verzeichnis enthält folgende Objekte:

- Die Konfigurationsdatei des Datenbankmanagers
- Das Systemdatenbankverzeichnis
- Das Knotenverzeichnis
- Die Knotenkonfigurationsdatei (db2nodes.cfg)
- Alle weiteren Dateien mit Debuginformationen, wie z. B. den Speicherauszug der Ausnahme- und Registrierdaten oder den Aufrufstapel für die DB2-Datenbankprozesse.

Terminologie:

Bitlänge
Die Anzahl an Bits für den virtuellen Speicher: 32-Bit und 64-Bit sind am meisten verbreitet. Dieser Begriff wird möglicherweise verwendet, um auf die Bitlänge einer Instanz, eines Anwendungscodes oder eines Codes einer externen Routine zu verweisen. 32-Bit-Anwendung bedeutet dasselbe wie Anwendung mit einer Länge von 32-Bit.

32-Bit-DB2-Instanz
Eine DB2-Instanz, die alle 32-Bit-Binärdaten enthält, einschließlich gemeinsam genutzter 32-Bit-Bibliotheken und ausführbarer 32-Bit-Dateien.

64-Bit-DB2-Instanz
Eine DB2-Instanz, die gemeinsam genutzte 64-Bit-Bibliotheken und gemeinsam genutzte 64-Bit-Dateien sowie alle 32-Bit-Clientanwendungsbibliotheken (für Client und Server) und Unterstützung für externe 32-Bit-Routinen (nur für eine Serverinstanz) enthält.

Entwerfen von Instanzen

DB2-Datenbanken werden innerhalb von DB2-Instanzen auf dem Datenbankserver erstellt. Durch die Erstellung mehrerer Instanzen auf demselben physischen Server wird für jede Instanz eine exklusive Datenbankserverumgebung bereitgestellt.

Sie können zum Beispiel eine Testumgebung und eine Produktionsumgebung auf demselben System verwalten. Sie können auch eine Instanz für jede Anwendung erstellen und sie ausschließlich für die Anwendung, der sie dienen soll, optimieren. Oder Sie können Ihre Lohndatenbank zwecks Datenschutz in einer eigenen Instanz speichern, sodass Eigner anderer Instanzen (auf demselben Server) die Lohndaten nicht anzeigen können.

Der Installationsprozess erstellt eine DB2-Standardinstanz, die durch die Umgebungsvariable DB2INSTANCE definiert wird. Diese Instanz wird für die meisten Operationen verwendet. Jedoch können auch nach der Installation Instanzen erstellt (bzw. gelöscht) werden.

Beim Festlegen und Entwerfen der Instanzen für Ihre Umgebung müssen Sie beachten, dass jede Instanz den Zugriff auf eine oder mehrere Datenbanken steuert. Jede Datenbank in einer Instanz hat einen eindeutigen Namen, besitzt eine eigene Gruppe von Systemkatalogtabellen (die zur Verwaltung der in der Datenbank erstellten Objekte dienen) und verfügt über eine eigene Konfigurationsdatei. Darüber hinaus hat jede Datenbank eine eigene Gruppe von erteilbaren Berechtigungen und Zugriffsrechten, die steuern, wie Benutzer mit den in der Datenbank gespeicherten Daten und Datenbankobjekten interagieren. Abb. 2 auf Seite 73 zeigt die hierarchische Beziehung zwischen Systemen, Instanzen und Datenbanken.

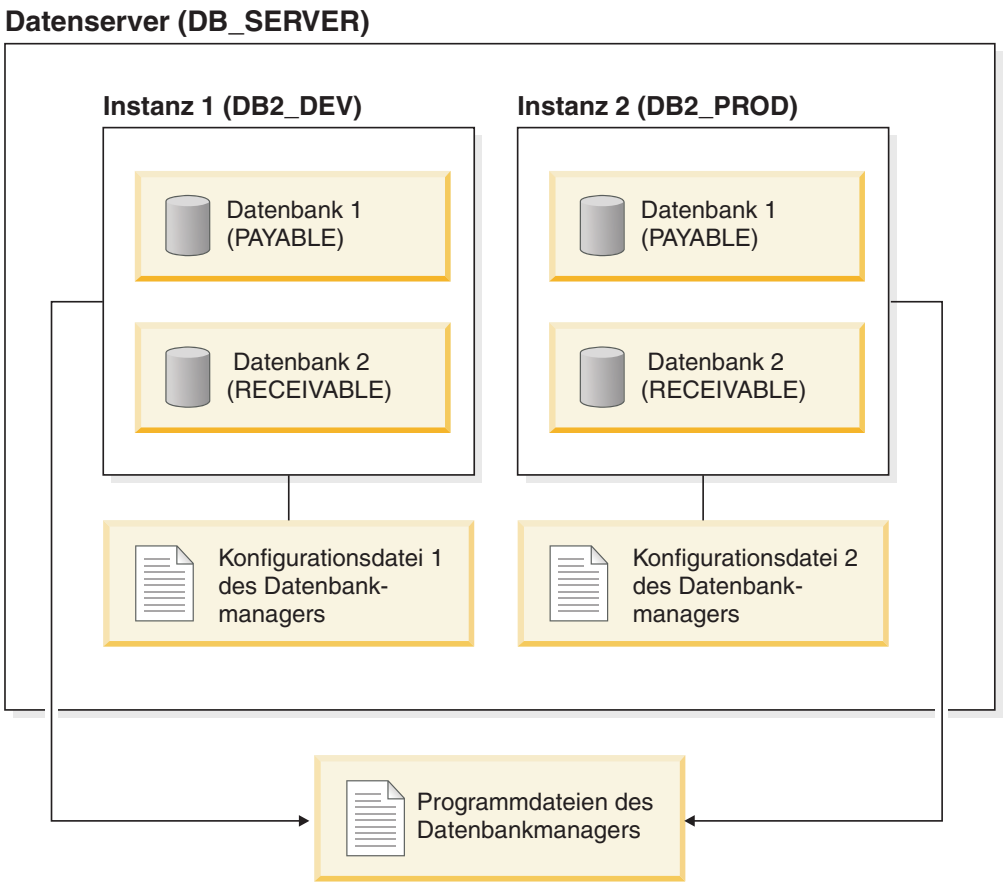


Abbildung 2. Hierarchische Beziehung zwischen DB2-Systemen, -Instanzen und -Datenbanken

Darüber hinaus müssen Sie auch einen weiteren besonderen Typ von Instanz in Ihre Überlegungen mit einbeziehen, der als *DB2-Verwaltungsserver* (DAS - DB2 Administration Server) bezeichnet wird. Der DAS ist ein spezieller DB2-Verwaltungssteuerpunkt, der nur zur Unterstützung der Verwaltungstasks auf anderen DB2-Servern verwendet wird. Ein DAS muss ausgeführt werden, wenn 'Clientkonfiguration - Unterstützung' zur Erkennung ferner Datenbanken oder grafischer Tools verwendet werden soll, die mit dem DB2-Produkt geliefert werden. Zu solchen Tools zählen zum Beispiel die Steuerzentrale oder die Taskzentrale. In einem DB2-Datenbankserver ist auch bei mehreren Instanzen immer nur ein DAS vorhanden.

Wenn Ihre Instanzen erstellt sind, können Sie zu jeder anderen verfügbaren Instanz (einschließlich der Instanzen auf anderen Systemen) eine Verbindung mit dem Befehl ATTACH herstellen. Wenn diese Verbindung besteht, können Sie Wartungs- und Dienstprogrammoperationen ausführen, die nur auf der Instanzebene ausgeführt werden können. Zum Beispiel können Sie eine Datenbank erstellen, Anwendung zwangsweise von einer Datenbank trennen, Datenbankaktivitäten überwachen oder den Inhalt der Konfigurationsdatei des Datenbankmanagers ändern, die einer bestimmten Instanz zugeordnet ist.

Standardinstanz

Im Rahmen Ihrer DB2-Installationsprozedur erstellen Sie eine Erstinstanz des Datenbankmanagers, die den Namen `DB2` besitzt, sofern nicht bereits eine andere

Instanz mit dem Namen „DB2" vorhanden ist. Wenn DB2 Version 8 installiert ist und Sie ein Upgrade auf Version 9.1 oder Version 9.5 durchführen, hat die Standardinstanz den Namen „DB2_01".

Unter Linux und UNIX kann die Erstinstanz einen beliebigen Namen, den geltenden Namensregeln entsprechenden Namen erhalten. Der Instanzname wird zum Definieren der Verzeichnisstruktur verwendet.

Damit diese Instanz sofort verwendet werden kann, werden bei der Installation folgende Konfigurationswerte festgelegt:

- Die Umgebungsvariable DB2INSTANCE wird auf den Wert „DB2" gesetzt.
- Die Registrierdatenbankvariable DB2INSTDEF wird auf den Wert „DB2" gesetzt.

Diese Einstellungen legen „DB2" als Standardinstanz fest. Sie können die Standardinstanz erst ändern, nachdem Sie eine weitere Instanz erstellt haben.

Vor der Verwendung des Datenbankmanagers muss die Datenbankumgebung für jeden Benutzer aktualisiert werden, sodass sie auf eine Instanz zugreifen und die DB2-Datenbankprogramme ausführen kann. Dies gilt für alle Benutzer (einschließlich der Benutzer mit Verwaltungsaufgaben).

Unter Linux- und UNIX-Betriebssystemen werden Beispielscriptdateien bereitgestellt, die Sie beim Einrichten der Datenbankumgebung unterstützten. Folgende Dateien werden bereitgestellt: db2profile für die Bourne- oder Korn-Shell und db2cshrc für die C-Shell. Diese Scripts befinden sich im Unterverzeichnis sqllib im Ausgangsverzeichnis des Instanzeigners. Der Instanzeigner oder jeder beliebige Benutzer, der zur SYSADM-Gruppe der Instanz gehört, kann das Script für alle Benutzer einer Instanz anpassen. Verwenden Sie sqllib/userprofile und sqllib/usercshrc, um ein Script für die einzelnen Benutzer anzupassen.

In den leeren Dateien sqllib/userprofile und sqllib/usercshrc können Sie bei der Instanzerstellung eigene Einstellungen für die Instanzumgebung hinzufügen. Die Dateien db2profile und db2cshrc werden bei der Instanzaktualisierung einer DB2 FixPak-Installation überschrieben. Wenn Sie die neuen Umgebungseinstellungen in den Scripts db2profile oder db2cshrc nicht verwenden möchten, können Sie sie mit dem entsprechenden Script *user* überschreiben, das am Ende des Scripts db2profile bzw. db2cshrc aufgerufen wird. Während der Migration einer Instanz (mit dem Befehl db2imigr) werden die Scripts 'user' mit kopiert, sodass Ihre Umgebungsänderungen weiterhin verwendet werden.

Das Beispielscript enthält Anweisungen für folgende Aufgaben:

- Aktualisieren eines Benutzerpfads (PATH) durch Hinzufügen der folgenden Verzeichnisse zum vorhandenen Suchpfad: die Unterverzeichnisse bin, adm und misc im Unterverzeichnis sqllib im Ausgangsverzeichnis des Instanzeigners
- Setzen der Umgebungsvariablen DB2INSTANCE auf den Instanznamen

Instanzverzeichnis

Im Instanzverzeichnis werden alle Informationen für eine Datenbankinstanz gespeichert. Die Position des Instanzverzeichnisses lässt sich nach der Erstellung nicht mehr ändern.

Das Instanzverzeichnis hat folgenden Inhalt:

- Die Konfigurationsdatei des Datenbankmanagers
- Das Systemdatenbankverzeichnis

- Das Knotenverzeichnis
- Die Knotenkonfigurationsdatei (db2nodes.cfg)
- Weitere Dateien mit Debuginformationen, wie z. B. den Speicherauszug der Ausnahme- und Registrierdaten oder den Aufrufstapel für die DB2-Prozesse.

Unter Linux- und UNIX-Betriebssystemen befindet sich das Instanzverzeichnis im Verzeichnis INSTHOME/sqllib, wobei INSTHOME das Ausgangsverzeichnis des Instanzeigners ist. Der Standardinstanz kann ein beliebiger Name nach den Richtlinien der Namensregeln gegeben werden.

Unter Windows-Betriebssystemen befindet sich das Instanzverzeichnis unter dem Verzeichnis /sqllib, in dem das DB2-Datenbankprodukt installiert wurde. Der Instanzname ist mit dem Namen des Service identisch. Daher sollte dieser nicht mit anderen Namen in Konflikt stehen. Kein Instanzname darf mit einem anderen Servicenamen identisch sein. Sie müssen über die richtige Berechtigung verfügen, um einen Service zu erstellen.

In einer Umgebung mit partitionierten Datenbanken wird das Instanzverzeichnis von allen Datenbankpartitionsservern, die zu der Instanz gehören, gemeinsam genutzt. Darum muss das Instanzverzeichnis auf einem gemeinsam genutzten Netzwerklaufwerk erstellt werden, auf das alle Computer in der Instanz zugreifen können.

db2nodes.cfg

Die Datei db2nodes.cfg dient zur Definition der Datenbankpartitionsserver, die an einer DB2-Instanz beteiligt sind. Darüber hinaus dient die Datei db2nodes.cfg zur Angabe der IP-Adresse oder des Hostnamens einer Hochgeschwindigkeitsverbindung, wenn Sie eine Hochgeschwindigkeitsverbindung zur Kommunikation für die Datenbankpartitionsserver verwenden möchten.

Mehrere Instanzen (Linux, UNIX)

Es ist möglich, unter einem Linux oder UNIX-Betriebssystem mehrere Instanzen zu haben, wenn das DB2-Produkt mit Rootberechtigungen installiert wurde. Obwohl alle Instanzen gleichzeitig ausgeführt werden, arbeiten diese unabhängig voneinander. Sie können jedoch immer nur jeweils innerhalb einer einzigen Instanz des Datenbankmanagers arbeiten.

Anmerkung: Zur Vermeidung von Umgebungskonflikten zwischen zwei oder mehr Instanzen sollten Sie sicherstellen, dass jede Instanz über ein eigenes Ausgangsverzeichnis verfügt. Eine gemeinsame Nutzung des Ausgangsverzeichnisses führt zur Rückgabe von Fehlern. Jedes Ausgangsverzeichnis kann sich im selben oder in einem anderen Dateisystem befinden.

Jeder Instanz wird der Instanzeigner und die Systemverwaltungsgruppe (SYSADM) zugeordnet. Dies geschieht bei der Erstellung der Instanz. Eine Benutzer-ID bzw. ein Benutzername kann nur für jeweils eine Instanz verwendet werden. Diese Benutzer-ID bzw. dieser Benutzername wird auch als der *Instanzeigner* bezeichnet.

Jeder Instanzeigner muss über ein eindeutiges Ausgangsverzeichnis verfügen. Alle zum Ausführen der Instanz erforderlichen Konfigurationsdateien werden im Ausgangsverzeichnis der Benutzer-ID bzw. des Benutzernamens des Instanzeigners erstellt. Falls es erforderlich wird, die Benutzer-ID bzw. den Benutzernamen des Instanzeigners aus dem System zu entfernen, könnten Sie der Instanz zugeordnete Dateien sowie den Zugriff auf in dieser Instanz gespeicherte Daten verlieren.

Daher sollten Sie die Benutzer-ID bzw. den Benutzernamen eines Instanzeigners ausschließlich für die Ausführung des Datenbankmanagers dedizieren.

Die Primärgruppe des Instanzeigners ist ebenfalls von Bedeutung. Diese Primärgruppe wird automatisch zur Systemverwaltungsgruppe der Instanz und erhält die Berechtigung SYSADM für die Instanz. Andere Benutzer-IDs oder Benutzernamen, die Mitglieder der Primärgruppe der Instanz sind, erhalten ebenfalls diese Berechtigungsstufe. Aus diesem Grund kann es wünschenswert sein, die Benutzer-ID bzw. den Benutzernamen des Instanzeigners einer Primärgruppe zuzuordnen, die für die Verwaltung von Instanzen reserviert ist. (Stellen Sie außerdem sicher, dass der Benutzer-ID bzw. dem Benutzernamen des Instanzeigners eine Primärgruppe zugeordnet ist. Andernfalls wird die Standardprimärgruppe des Systems verwendet.)

Wenn Sie bereits über eine Gruppe verfügen, die Sie als Systemverwaltungsgruppe für die Instanz verwenden wollen, können Sie diese Gruppe einfach beim Erstellen der Benutzer-ID bzw. des Benutzernamens des Instanzeigners als Primärgruppe zuordnen. Fügen Sie andere Benutzer, denen Sie die Verwaltungsberechtigung für die Instanz erteilen wollen, derjenigen Gruppe hinzu, die als Systemverwaltungsgruppe zugeordnet ist.

Wenn Sie die Berechtigung SYSADM für die einzelnen Instanzen voneinander trennen wollen, stellen Sie sicher, dass jede Benutzer-ID bzw. jeder Benutzername eines Instanzeigners eine andere Primärgruppe verwendet. Bei Verwendung einer allgemeinen Berechtigung SYSADM für mehrere Instanzen können Sie jedoch dieselbe Primärgruppe für mehrere Instanzen verwenden.

Mehrere Instanzen (Windows)

Es ist möglich, mehrere Instanzen des Datenbankmanagers auf demselben Computer auszuführen. Jede Instanz des Datenbankmanagers pflegt eigene Datenbanken und verfügt über eigene Konfigurationsparameter für den Datenbankmanager.

Anmerkung: Die Instanzen können auch unterschiedlichen DB2-Kopien auf einem Computer angehören, der unterschiedliche Versionen des Datenbankmanagers aufweisen kann.

Eine Instanz des Datenbankmanagers besteht aus folgenden Komponenten:

- Ein Windows-Dienst (Service), der die Instanz darstellt. Der Name des Service ist mit dem Namen der Instanz identisch. Der Anzeigename des Service (im Fenster 'Dienste') ist der Instanzname, dem die Zeichenfolge „DB2 - " vorangestellt ist. Zum Beispiel gibt es für eine Instanz mit dem Namen „DB2" einen Windows-Dienst mit dem Namen „DB2", der als „DB2 - <name_der_db2-kopie> - DB2" angezeigt wird.

  Anmerkung: Ein Windows-Dienst (Service) wird für Clientinstanzen nicht erstellt.
- Ein Instanzverzeichnis. Dieses Verzeichnis enthält die Konfigurationsdateien des Datenbankmanagers, das Systemdatenbankverzeichnis, das Knotenverzeichnis, das DCS-Verzeichnis, alle Diagnoseprotokolldateien sowie alle Speicherauszugsdateien, die der Instanz zugeordnet sind. Das Instanzverzeichnis ist standardmäßig ein Unterverzeichnis im Verzeichnis SQLLIB und hat den gleichen Namen wie die Instanz. Für die Instanz „DB2" ist dieses Verzeichnis zum Beispiel `C:\SQLLIB\DB2`, wobei `C:\SQLLIB` das Verzeichnis ist, in dem der Datenbankmanager installiert ist. Sie können mithilfe der Registrierdatenbankvariablen DB2INSTPROF die Standardposition des Instanzverzeichnisses ändern. Wenn die

Registrierdatenbankvariable DB2INSTPROF eine andere Position angibt, wird das Instanzverzeichnis unter dem Verzeichnis erstellt, auf das die Variable DB2INSTPROF zeigt. Wenn zum Beispiel `DB2INSTPROF=D:\DB2PROFS` definiert würde, wäre das Instanzverzeichnis `D:\DB2PROFS\DB2`.

  – Setzen Sie unter Verwendung des Befehls db2set.exe -g die Variable DB2INSTPROF auf `c:\DB2PROFS`.
  – Führen Sie den Befehl DB2ICRT.exe aus, um die Instanz zu erstellen.

- Wenn Sie unter Windows-Betriebssystemen eine Instanz erstellen, sind die Standardpositionen für Datendateien des Benutzers wie zum Beispiel Instanzverzeichnisse und die Datei 'db2cli.ini' die folgenden Verzeichnisse:
  – Dokumente und Einstellungen\All Users\Application Data\IBM\DB2\name_der_kopie unter den Betriebssystemen Windows XP und Windows 2003
  – ProgramData\IBM\DB2\name_der_kopie unter dem Betriebssystem Windows Vista

Erstellen von Instanzen

Im Rahmen der Installation des Datenbankmanagers wird zwar eine Instanz erstellt, jedoch können Ihre Geschäftsanforderungen möglicherweise eine Erstellung zusätzlicher Instanzen erforderlich machen.

Voraussetzungen

Wenn Sie der Administratorengruppe unter Windows angehören oder über die Rootberechtigung auf Linux- oder UNIX-Plattformen verfügen, können Sie zusätzliche Instanzen hinzufügen. Der Computer, auf dem die Instanz hinzugefügt wird, wird als Instanzeignercomputer (Knoten 0) definiert. Fügen Sie Instanzen unbedingt auf einem Computer hinzu, auf dem sich ein DB2-Verwaltungsserver (DAS) befindet. Die Instanz-IDs dürfen nicht 'root' sein und kein abgelaufenes Kennwort haben.

Einschränkungen

- Unter Linux- und UNIX-Betriebssystemen können zusätzliche Instanzen nicht für Nichtrootinstallationen erstellt werden.
- Wenn vorhandene Benutzer-IDs zur Erstellung von DB2-Instanzen verwendet werden, stellen Sie sicher, dass für die Benutzer-IDs folgende Bedingungen zutreffen:
  – Sie sind nicht gesperrt.
  – Sie haben keine abgelaufenen Kennwörter.

Geben Sie den folgenden Befehl in die Befehlszeile ein, um eine Instanz hinzuzufügen:

```
db2icrt <instanzname>
```

Bei der Erstellung einer Instanz auf einem AIX-Server müssen Sie die abgeschirmte Benutzer-ID (FencedID) angeben. Beispiel:

```
DB2DIR/instance/db2icrt -u db2fenc1 db2inst1
```

Bei der Verwendung des Befehls db2icrt zum Hinzufügen einer weiteren DB2-Instanz sollten Sie den Anmeldenamen des Instanzeigners und optional den Authentifizierungstyp der Instanz angeben. Der Authentifizierungstyp gilt für alle Datenbanken, die unter dieser Instanz erstellt werden. Der Authentifizierungstyp ist eine Anweisung, wo die Authentifizierung von Benutzern stattfinden soll.

Mit der Umgebungsvariablen DB2INSTPROF können Sie die Position des in DB2PATH angegebenen Instanzverzeichnisses ändern. Sie benötigen dafür Schreibzugriff auf das Instanzverzeichnis. Wenn die Verzeichnisse in einem anderen Pfad als DB2PATH erstellt werden sollen, müssen Sie die Variable DB2INSTPROF definieren, bevor Sie den Befehl db2icrt eingeben.

Für DB2 Enterprise Server Edition (ESE) müssen Sie außerdem deklarieren, dass Sie eine neue Instanz hinzufügen, bei der es sich um ein partitioniertes Datenbanksystem handelt. Wenn Sie darüber hinaus mit einer ESE-Instanz arbeiten, die mehr als eine Datenbankpartition hat, und Fast Communication Manager (FCM) verwenden, können Sie mehrere Verbindungen zwischen Datenbankpartitionen haben, indem Sie bei der Erstellung der Instanz mehrere TCP/IP-Ports definieren.

Für Windows-Betriebssysteme verwenden Sie zum Beispiel den Befehl db2icrt mit dem Parameter -r <portbereich>. Der Portbereich wird wie folgt dargestellt, wobei `basisport` der erste Port und `endport` der letzte Port in einem Bereich von Portnummern ist, die von FCM verwendet werden können:

```
-r:<basisport,endport>
```

Ändern von Instanzen

Die Instanzen wurden so gestaltet, dass sie durch ein nachfolgendes Installieren und Deinstallieren anderer Produkte möglichst wenig beeinflusst werden. Unter Linux und UNIX können Sie Instanzen nach der Installation oder dem Entfernen von ausführbaren Dateien oder Komponenten aktualisieren. Unter Windows führen Sie den Befehl db2iupdt aus.

In den meisten Fällen erhalten bzw. verlieren vorhandene Instanzen automatisch den Zugriff auf die Funktionen des Produkts, das installiert bzw. deinstalliert wird. Beim Installieren bestimmter Programmdateien oder Komponenten übernehmen die vorhandenen Instanzen jedoch nicht automatisch die neuen Systemkonfigurationsparameter oder erhalten nicht automatisch Zugriff auf die neu hinzugekommenen Funktionen. In solchen Fällen muss die betreffende Instanz aktualisiert werden.

Wenn der Datenbankmanager durch Installieren einer vorläufigen Programmkorrektur (PTF - Program Temporary Fix) oder eines Patch-Codes aktualisiert wird, sollten alle vorhandenen Datenbankinstanzen mithilfe des Befehls db2iupdt (Rootinstallationen) oder des Befehls db2nrupdt (Nichtrootinstallationen) aktualisiert werden.

Machen Sie sich zunächst mit den vorhandenen Instanzen und Datenbankpartitionsservern vertraut, bevor Sie versuchen, eine Instanz zu ändern oder zu löschen.

Aktualisieren der Instanzkonfiguration (Linux, UNIX)

Dieser Abschnitt bezieht sich nur auf Rootinstanzen. Zur Aktualisierung von Nichtrootinstanzen führen Sie den Befehl db2nrupdt aus.

Der Befehl db2iupdt aktualisiert die angegebene Instanz durch folgende Operationen:

- Die Dateien im Unterverzeichnis `sqllib` im Ausgangsverzeichnis des Instanzeigners werden ersetzt.
- Wenn der Knotentyp geändert wurde, wird eine neue Konfigurationsdatei für den Datenbankmanager erstellt. Dabei werden Werte aus der vorhandenen

Konfigurationsdatei des Datenbankmanagers mit der Standardkonfigurationsdatei des Datenbankmanagers für den neuen Knotentyp zusammengefügt. Beim Erstellen einer neuen Konfigurationsdatei für den Datenbankmanager wird die alte Datei im Unterverzeichnis `backup` des Unterverzeichnisses `sqllib` im Ausgangsverzeichnis des Instanzeigners gesichert.

Unter AIX befindet sich der Befehl db2iupdt im Verzeichnis `/usr/opt/db2_09_05/instance/`. Unter HP-UX, Solaris oder Linux befindet sich der Befehl db2iupdt im Verzeichnis `/opt/IBM/db2/V9.5/instance/`.

Geben Sie den folgenden Befehl in die Befehlszeile ein, um eine Instanz zu aktualisieren:

```
db2iupdt InstName
```

Die Angabe `InstName` ist der Anmeldename des Instanzeigners.

Für diesen Befehl sind weitere optionale Parameter verfügbar:

-h oder -?
Zeigt ein Hilfemenü für diesen Befehl an.

-d Dieser Parameter konfiguriert den Debugmodus für die Fehlerbestimmung.

-a AuthTyp
Dieser Parameter gibt den Authentifizierungstyp für die Instanz an. Gültige Authentifizierungstypen sind SERVER, SERVER_ENCRYPT und CLIENT. Wenn dieser Parameter nicht angegeben wird, wird standardmäßig der Authentifizierungstyp SERVER verwendet, wenn ein DB2-Server installiert ist. Ansonsten wird die Option CLIENT verwendet. Der für die Instanz angegebene Authentifizierungstyp gilt für alle Datenbanken, die zu dieser Instanz gehören.

-e Ermöglicht Ihnen die Aktualisierung aller vorhandenen Instanzen. Mit dem Befehl db2ilist können Sie die vorhandenen Instanzen auflisten.

-u Abgeschirmte_ID
Dieser Parameter gibt den Benutzer an, unter dem die abgeschirmten benutzerdefinierten Funktionen (UDFs) und die gespeicherten Prozeduren ausgeführt werden. Dieser Parameter ist nicht erforderlich, wenn Sie den Data Server Client oder das DB2 Software Developer's Kit installieren. Für andere DB2-Produkte ist dies ein erforderlicher Parameter. Anmerkung: Die 'Abgeschirmte-ID' darf nicht "root" oder "bin" sein.

-k Dieser Parameter behält den aktuellen Instanztyp bei. Wenn Sie diesen Parameter nicht angeben, wird die aktuelle Instanz in der nachfolgend angegebenen Reihenfolge auf den höchsten verfügbaren Instanztyp hochgestuft:

- Partitionierter Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen und fernen Clients
- Client

Beispiele:

- Wenn Sie nach dem Erstellen der Instanz DB2 Workgroup Server Edition oder DB2 Enterprise Server Edition installiert haben, geben Sie den folgenden Befehl ein, um die Instanz zu aktualisieren:

  ```
  db2iupdt -u db2fenc1 db2inst1
  ```

- Wenn Sie nach dem Erstellen der Instanz DB2 Connect Enterprise Edition installiert haben, können Sie den Instanznamen wie folgt auch als abgeschirmte ID verwenden:

  ```
  db2iupdt -u db2inst1 db2inst1
  ```

- Zur Aktualisierung von Clientinstanzen rufen Sie den folgenden Befehl auf:

  ```
  db2iupdt db2inst1
  ```

Aktualisieren der Instanzkonfiguration (Windows)

Zur Aktualisierung der Instanzkonfiguration unter Windows verwenden Sie den Befehl db2iupdt.

Der Befehl db2iupdt aktualisiert die angegebene Instanz durch folgende Operationen:

- Die Dateien im Unterverzeichnis `sqllib` im Ausgangsverzeichnis des Instanzeigners werden ersetzt.
- Wenn der Knotentyp geändert wird, wird eine neue Konfigurationsdatei für den Datenbankmanager erstellt. Dabei werden Werte aus der vorhandenen Konfigurationsdatei des Datenbankmanagers mit der Standardkonfigurationsdatei des Datenbankmanagers für den neuen Knotentyp zusammengefügt. Beim Erstellen einer neuen Konfigurationsdatei für den Datenbankmanager wird die alte Datei im Unterverzeichnis `backup` des Unterverzeichnisses `sqllib` im Ausgangsverzeichnis des Instanzeigners gesichert.

Der Befehl db2iupdt befindet sich im Verzeichnis `\sqllib\bin`.

Der Befehl wird wie folgt verwendet:

```
db2iupdt InstName
```

Die Angabe 'InstName' ist der Anmeldename des Instanzeigners.

Für diesen Befehl sind weitere optionale Parameter verfügbar:

/h: hostname
: Überschreibt den Standard-TCP/IP-Hostnamen, wenn es einen oder mehrere TCP/IP-Hostnamen für den aktuellen Computer gibt.

/p: instanzprofilpfad
: Gibt den neuen Instanzprofilpfad für die aktualisierte Instanz an.

/r: basisport,endport
: Gibt den Bereich der TCP/IP-Ports an, die von der partitionierten Datenbankinstanz beim Betrieb mit mehreren Datenbankpartitionen verwendet werden.

/u: benutzername,kennwort
: Gibt den Kontonamen und das Kennwort für den DB2-Service (Dienst) an.

Arbeiten mit Instanzen

Bei der Arbeit mit Instanzen können Sie Instanzen starten oder stoppen sowie Verbindungen zu Instanzen herstellen (ATTACH) oder trennen (DETACH).

Jede Instanz wird von Benutzern verwaltet, die zur Gruppe SYSADM_GROUP gehören, die in der *Instanzkonfigurationsdatei* definiert ist. Diese Datei wird auch als *Konfigurationsdatei des Datenbankmanagers* bezeichnet. Die Erstellung von Benutzer-IDs und Benutzergruppen ist je nach Betriebsumgebung unterschiedlich.

Automatisches Starten von Instanzen

Unter Windows-Betriebssystemen wird die bei der Installation erstellte Datenbankinstanz standardmäßig auf automatischen Start eingestellt. Eine Instanz, die mithilfe des Befehls 'db2icrt' erstellt wird, wird auf manuelles Starten eingestellt. Zum Ändern der Startart müssen Sie dieses Merkmal des DB22-Service im Fenster 'Dienste' ändern.

Geben Sie unter UNIX-Betriebssystemen den folgenden Befehl ein, um eine Instanz auf automatischen Start nach jedem Neustart des Systems einzustellen:

```
db2iauto -on <instanzname>
```

Dabei ist `<instanzname>` der Anmeldename der Instanz. Geben Sie unter UNIX-Betriebssystemen den folgenden Befehl ein, um zu verhindern, dass eine Instanz nach jedem Systemneustart automatisch gestartet wird:

```
db2iauto -off <instanzname>
```

Dabei ist `<instanzname>` der Anmeldename der Instanz.

Starten von Instanzen (Linux, UNIX)

Das Starten und Stoppen einer DB2-Instanz kann während des normalen Geschäftsbetriebs erforderlich werden. Sie müssen beispielsweise eine Instanz starten, um einige der folgenden Tasks ausführen zu können: Verbindung zu einer Datenbank in der Instanz herstellen, eine Anwendung vorkompilieren, ein Paket an eine Datenbank binden oder auf Hostdatenbanken zugreifen.

Führen Sie folgende Schritte aus, bevor Sie eine Instanz auf Ihrem Linux- oder UNIX-System starten:

1. Melden Sie sich mit einer Benutzer-ID oder einem Benutzernamen an, die bzw. der über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT für die Instanz verfügt. Oder melden Sie sich als Instanzeigner an.
2. Führen Sie das Startscript wie folgt aus, wobei `INSTHOME` das Ausgangsverzeichnis der Instanz ist, die Sie verwenden wollen:

```
. INSTHOME/sqllib/db2profile      (für Bourne- oder Korn-Shell)
source INSTHOME/sqllib/db2cshrc   (für C-Shell)
```

Geben Sie in der Befehlszeile Folgendes ein, um die Instanz zu starten:

```
db2start
```

Anmerkung: Wenn Sie Befehle zum Starten oder Stoppen des Datenbankmanagers einer Instanz ausführen, wendet der DB2-Datenbankmanager den Befehl auf die aktuelle Instanz an.

Starten von Instanzen (Windows)

Das Starten und Stoppen einer DB2-Instanz kann während des normalen Geschäftsbetriebs erforderlich werden. Sie müssen beispielsweise eine Instanz starten, um einige der folgenden Tasks ausführen zu können: Verbindung zu einer Datenbank in der Instanz herstellen, eine Anwendung vorkompilieren, ein Paket an eine Datenbank binden oder auf eine Hostdatenbank zugreifen.

Damit die DB2-Datenbankinstanz erfolgreich als Service (bzw. Dienst) mit 'db2start' gestartet werden kann, muss das Benutzerkonto über das richtige Zugriffsrecht verfügen, wie es vom Betriebssystem Windows zum Starten eines Windows-Dienstes definiert wird. Das Benutzerkonto muss ein Mitglied der

Gruppe der Administratoren, der Serveroperatoren oder der Hauptbenutzer sein. Wenn die erweiterte Sicherheit aktiviert ist, können standardmäßig nur Mitglieder der Gruppen DB2ADMNS und Administratoren die Datenbank starten.

Geben Sie den folgenden Befehl in die Befehlszeile ein, um eine Instanz zu starten:

```
db2start
```

Anmerkung: Wenn Sie Befehle zum Starten oder Stoppen des Datenbankmanagers einer Instanz ausführen, wendet der DB2-Datenbankmanager den Befehl auf die aktuelle Instanz an.

Durch den Befehl db2start wird die DB2-Datenbankinstanz als Windows-Dienst (Service) gestartet. Die DB2-Datenbankinstanz unter Windows kann immer noch als Prozess ausgeführt werden, indem der Schalter '/D' beim Aufrufen des Befehls 'db2start' angegeben wird. Die DB2-Datenbankinstanz kann als Service auch über die Systemsteuerung oder mithilfe des Befehls NET START gestartet werden.

Bei der Ausführung in einer Umgebung mit partitionierten Datenbanken wird jeder Datenbankpartitionsserver als Windows-Dienst gestartet. In einer Umgebung mit partitionierten Datenbanken können Sie den Schalter '/D' nicht verwenden, um eine DB2-Instanz als Prozess zu starten.

Herstellen und Trennen von Verbindungen zu Instanzen (mit ATTACH und DETACH)

Zum Herstellen der Verbindung zu einer anderen Instanz der Datenbank, die auch eine ferne Instanz sein kann, verwenden Sie auf allen Plattformen den Befehl ATTACH. Zum Trennen der Verbindung zu einer Instanz verwenden Sie den Befehl DETACH.

Es muss bereits mehr als eine Instanz vorhanden sein.

Geben Sie in die Befehlszeile Folgendes ein, um eine Verbindung zu einer Instanz herzustellen:

```
db2 attach to <instanzname>
```

Um zum Beispiel eine Verbindung zu einer Instanz mit dem Namen 'testdb2' herzustellen, die zuvor im Knotenverzeichnis katalogisiert wurde, müssen Sie folgenden Befehl eingeben:

```
db2 attach to testdb2
```

Nach Beendigung der Wartungsaktivitäten für die Instanz 'testdb2' geben Sie zum Trennen der Verbindung zu der Instanz den folgenden Befehl in die Befehlszeile ein:

```
db2 detach
```

Herstellen und Trennen der Verbindung aus Clientanwendungen heraus:

- Zur Herstellung der Verbindung zu einer Instanz aus einer Clientanwendung heraus rufen Sie die API 'sqleatin' auf.
- Zum Trennen der Verbindung zu einer Instanz aus einer Clientanwendung heraus rufen Sie die API 'sqledtin' auf.

Arbeiten mit Instanzen in derselben oder in anderen DB2-Kopien

Sie können mehrere Instanzen gleichzeitig in derselben DB2-Kopie oder in verschiedenen DB2-Kopien ausführen.

Wenn Sie mit Instanzen in derselben DB2-Kopie arbeiten möchten, müssen Sie wie folgt vorgehen:

1. Erstellen Sie alle Instanzen in ein und derselben DB2-Kopie, oder migrieren Sie alle Instanzen auf ein und dieselbe DB2-Kopie.
2. Setzen Sie die Umgebungsvariable DB2INSTANCE auf den Namen der Instanz, mit der Sie arbeiten möchten, bevor Sie Befehle für diese Instanz absetzen.

Damit eine Instanz nicht auf die Datenbanken einer anderen Instanz zugreift, werden die Datenbankdateien für eine Instanz unter einem Verzeichnis erstellt, das den gleichen Namen wie die Instanz besitzt. Wenn zum Beispiel eine Datenbank auf dem Laufwerk `C:` für die Instanz „DB2" erstellt wird, werden die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\DB2` erstellt. Analog werden bei der Erstellung einer Datenbank auf Laufwerk `C:` für die Instanz TEST die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\TEST` erstellt. Standardmäßig entspricht der Wert dem Buchstaben des Laufwerks, in dem das DB2-Produkt installiert ist. Weitere Informationen finden Sie im Abschnitt zum Datenbankmanagerkonfigurationsparameter *dftdbpath*.

Verwenden Sie eine der folgenden Methoden, wenn Sie mit einer Instanz in einem System mit mehreren DB2-Kopien arbeiten möchten:

- Mithilfe des Befehlsfensters über Start → Programme → IBM DB2 → *<name_der_DB2-kopie>* → Befehlszeilentools → Befehlsfenster: Das Befehlsfenster ist bereits mit den richtigen Umgebungsvariablen für die jeweils ausgewählte DB2-Kopie eingerichtet.
- Mithilfe der Datei db2envar.bat über ein Befehlsfenster:
  1. Öffnen Sie ein Befehlsfenster.
  2. Führen Sie die Datei db2envar.bat aus, und verwenden Sie dabei den vollständig qualifizierten Pfad für die DB2-Kopie, die die Anwendung verwenden soll:

     ```
     <installationsverzeichnis_der_db2-kopie>\bin\db2envar.bat
     ```

Stoppen von Instanzen (Linux, UNIX)

Möglicherweise müssen Sie die aktuelle Instanz des Datenbankmanagers stoppen.

Vorbereitung

Zum Stoppen einer Instanz auf dem Linux- oder UNIX-System sind folgende Schritte erforderlich:

1. Melden Sie sich mit einer Benutzer-ID oder einem Benutzernamen, die bzw. der über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT verfügt, bei der Instanz an bzw. stellen Sie die Verbindung (ATTACH) zu der Instanz her. Oder melden Sie sich als Instanzeigner an.
2. Zeigen Sie alle Anwendungen und Benutzer an, die mit der Datenbank, die Sie stoppen wollen, verbunden sind. Lassen Sie sich eine Liste der Anwendungen anzeigen, um sicherzugehen, dass keine wichtigen oder kritischen Anwendungen aktiv sind. Dazu benötigen Sie die Berechtigung SYSADM, SYSCTRL oder SYSMAINT.

3. Erzwingen Sie die Trennung aller Anwendungen und Benutzer von der Datenbank. Für das Erzwingen der Trennung von Benutzern benötigen Sie die Berechtigung SYSADM oder SYSCTRL.

Informationen zu dieser Task

Der Befehl db2stop kann nur auf dem Server ausgeführt werden. Bei der Ausführung dieses Befehls sind keine Datenbankverbindungen zulässig. Wenn es Instanzverbindungen (mit ATTACH) gibt, werden diese zwangsweise getrennt, bevor die Instanz gestoppt wird.

Anmerkung: Wenn Sitzungen des Befehlszeilenprozessors mit einer Instanz verbunden sind, müssen Sie jede dieser Sitzungen einzeln durch Ausführen des Befehls terminate beenden, bevor Sie den Befehl db2stop ausführen. Der Befehl db2stop stoppt die durch die Umgebungsvariable **DB2INSTANCE** definierte Instanz.

Anmerkung: Wenn Sie Befehle zum Starten oder Stoppen des Datenbankmanagers einer Instanz ausführen, wendet der DB2-Datenbankmanager den Befehl auf die aktuelle Instanz an. Weitere Informationen finden Sie in Festlegen der Umgebungsvariablen der aktuellen Instanz.

Vorgehensweise

Geben Sie in die Befehlszeile den folgenden Befehl ein, um die Instanz zu stoppen:
`db2stop`
Sie können den Befehl db2stop zum Stoppen oder Löschen einzelner Datenbankpartitionen innerhalb einer Umgebung mit partitionierten Datenbanken verwenden. Wenn Sie in einer Umgebung mit partitionierten Datenbanken versuchen, eine logische Partition mit dem folgenden Befehl zu löschen

```
db2stop drop nodenum <0>
```

müssen Sie sicherstellen, dass momentan keine Benutzer versuchen, auf die Datenbank zuzugreifen. Wenn dies dennoch der Fall ist, empfangen Sie eine Fehlernachricht SQL6030N.

Stoppen von Instanzen (Windows)

Möglicherweise müssen Sie die aktuelle Instanz des Datenbankmanagers stoppen.

Vorbereitung

Zum Stoppen einer Instanz auf Ihrem System sind folgende Schritte erforderlich:

1. Das Benutzerkonto, das den DB2-Datenbankservice stoppt, muss über das richtige im Windows-Betriebssystem definierte Zugriffsrecht verfügen. Das Benutzerkonto muss ein Mitglied der Gruppe der Administratoren, der Serveroperatoren oder der Hauptbenutzer sein.
2. Zeigen Sie alle Anwendungen und Benutzer an, die mit der Datenbank, die Sie stoppen wollen, verbunden sind. Lassen Sie sich eine Liste der Anwendungen anzeigen, um sicherzugehen, dass keine wichtigen oder kritischen Anwendungen aktiv sind. Dazu benötigen Sie die Berechtigung SYSADM, SYSCTRL oder SYSMAINT.
3. Erzwingen Sie die Trennung aller Anwendungen und Benutzer von der Datenbank. Für das Erzwingen der Trennung von Benutzern benötigen Sie die Berechtigung SYSADM oder SYSCTRL.

Informationen zu dieser Task

Der Befehl db2stop kann nur auf dem Server ausgeführt werden. Bei der Ausführung dieses Befehls sind keine Datenbankverbindungen zulässig. Wenn jedoch Instanzverbindungen vorhanden sind, werden diese zwangsweise getrennt, bevor der DB2-Datenbankservice gestoppt wird.

Anmerkung: Wenn Sitzungen des Befehlszeilenprozessors mit einer Instanz verbunden sind, müssen Sie jede dieser Sitzungen einzeln durch Ausführen des Befehls terminate beenden, bevor Sie den Befehl db2stop ausführen. Der Befehl db2stop stoppt die durch die Umgebungsvariable **DB2INSTANCE** definierte Instanz.

Berücksichtigen Sie, dass bei der Verwendung des Datenbankmanagers in einer Umgebung mit partitionierten Datenbanken jeder Datenbankpartitionsserver als Service (bzw. Dienst) gestartet wird. Jeder dieser Services muss gestoppt werden.

Anmerkung: Wenn Sie Befehle zum Starten oder Stoppen des Datenbankmanagers einer Instanz ausführen, wendet der Datenbankmanager den Befehl auf die aktuelle Instanz an. Weitere Informationen finden Sie im Abschnitt Festlegen der Umgebungsvariablen der aktuellen Instanz.

Vorgehensweise

Stoppen Sie eine Instanz auf Ihrem System mit einer der folgenden Methoden:

- Stoppen mit dem Befehl db2stop
- Stoppen mit dem Befehl NET STOP
- Stoppen der Instanz aus einer Anwendung heraus

Löschen von Instanzen

Dieser Abschnitt bezieht sich nur auf Rootinstanzen auf allen Plattformen. Zum Löschen von Nichtrootinstanzen müssen Sie Ihr DB2-Produkt deinstallieren.

Geben Sie den folgenden Befehl in die Befehlszeile ein, um eine Instanz zu entfernen:

```
db2idrop <instanzname>
```

Zum Entfernen einer Instanz über die Befehlszeile müssen Sie die folgenden vorbereitenden Arbeitsschritte ausführen:

1. Stoppen Sie alle Anwendungen, die momentan mit der Instanz arbeiten.
2. Stoppen Sie den Befehlszeilenprozessor durch Ausführen des Befehls 'terminate' in jedem Befehlsfenster.
3. Stoppen Sie die Instanz durch Ausführen des Befehls db2stop.
4. Führen Sie ein Backup für das in der Registrierdatenbankvariablen DB2INSTPROF angegebene Instanzverzeichnis durch.

   Unter Linux- und UNIX-Betriebssystemen könnte es sinnvoll sein, die Dateien im Verzeichnis INSTHOME/sqllib durch ein Backup zu sichern (dabei ist INSTHOME das Ausgangsverzeichnis des Instanzeigners). Beispielsweise könnten Sie beabsichtigen, die Konfigurationsdatei des Datenbankmanagers (db2systm), die Datei db2nodes.cfg, die benutzerdefinierten Funktionen oder die abgeschirmten gespeicherten Prozeduren zu sichern.

5. (Nur unter Linux- und UNIX-Betriebssystemen:) Melden Sie sich als Instanzeigner ab.
6. (Nur unter Linux- und UNIX-Betriebssystemen:) Melden Sie sich als Benutzer mit Rootberechtigung an.
7. Setzen Sie den folgenden Befehl db2idrop ab:

   ```
   db2idrop InstName
   ```

   Dabei ist `InstName` der Name der zu löschenden Instanz.

   Dieser Befehl entfernt den Eintrag für diese Instanz aus der Instanzliste und löscht das Instanzverzeichnis.
8. (Nur unter Linux- und UNIX-Betriebssystemen:) Entfernen Sie (optional) als Benutzer mit Rootberechtigung die Gruppe und die Benutzer-ID des Instanzeigners (sofern sie nur für diese Instanz verwendet wurde). Löschen Sie die Gruppe und die Benutzer-ID nicht, wenn Sie vorhaben, die Instanz erneut zu erstellen.

   Dieser Schritt ist optional, da die Benutzer-ID und die Gruppe des Instanzeigners möglicherweise auch für andere Zwecke verwendet werden.

Der Befehl db2idrop entfernt den Instanzeintrag aus der Instanzliste und löscht das Unterverzeichnis sqllib im Ausgangsverzeichnis des Instanzeigners.

Anmerkung: Wenn Sie unter Linux- und UNIX-Betriebssystemen versuchen, eine Instanz mithilfe des Befehls db2idrop zu löschen, wird eine Nachricht generiert, die besagt, dass das Unterverzeichnis 'sqllib' nicht gelöscht werden kann, und im Unterverzeichnis 'adm' werden verschiedene Dateien mit der Erweiterung .nfs generiert. Das Unterverzeichnis adm ist ein über NFS angehängtes System, und die Dateien werden auf dem Server gesteuert. Sie müssen die Dateien *.nfs von der Position auf dem Dateiserver löschen, an der das Verzeichnis angehängt ist. Anschließend können Sie das Unterverzeichnis sqllib entfernen.

Kapitel 5. Lightweight Directory Access Protocol (LDAP)

LDAP (Lightweight Directory Access Protocol) ist eine Standardmethode zum Zugriff auf Verzeichnisservices. Bei einem Verzeichnisservice handelt es sich um ein Repository mit Ressourceninformationen zu mehreren Systemen und Services innerhalb einer verteilten Umgebung. Er stellt den Client- und Serverzugriff auf diese Ressourcen bereit.

Jede Datenbankserverinstanz veröffentlicht Informationen über ihre Existenz auf einem LDAP-Server und stellt dem LDAP-Verzeichnis Datenbankinformationen zur Verfügung, wenn die Datenbanken erstellt werden. Wenn ein Client eine Verbindung zur Datenbank herstellt, können die Kataloginformationen für den Server aus dem LDAP-Verzeichnis abgerufen werden. Die einzelnen Clients müssen die Kataloginformationen nun nicht mehr lokal auf den verschiedenen Maschinen speichern. Clientanwendungen durchsuchen das LDAP-Verzeichnis nach den erforderlichen Informationen für die Herstellung der Verbindung zur Datenbank.

Es gibt einen Cachingmechanismus, der es ermöglicht, dass der Client den LDAP-Verzeichnisserver nur einmal durchsuchen muss. Wenn die Informationen aus dem LDAP-Verzeichnisserver abgerufen sind, werden sie auf der lokalen Maschine entsprechend den Werten des Konfigurationsparameters *dir_cache* des Datenbankmanagers und der Registrierdatenbankvariablen DB2LDAPCACHE gespeichert bzw. im Cache abgelegt. Der Konfigurationsparameter *dir_cache* des Datenbankmanagers dient zum Speichern von Datenbank-, Knoten- und DCS-Verzeichnisdateien in einem Speichercache. Der Verzeichniscache wird von einer Anwendung genutzt, bis die Anwendung geschlossen wird. Die Registrierdatenbankvariable DB2LDAPCACHE dient zum Speichern von Datenbank-, Knoten- und DCS-Verzeichnisdateien in einem lokalen Plattencache.

- Sind DB2LDAPCACHE=NO und *dir_cache*=NO definiert, werden die Informationen immer aus LDAP gelesen.
- Bei DB2LDAPCACHE=NO und *dir_cache*=YES werden die Informationen einmal aus dem LDAP gelesen und in den DB2-Cache eingefügt.
- Ist DB2LDAPCACHE=YES oder diese Registrierdatenbankvariable überhaupt nicht definiert, werden die Informationen einmal aus dem LDAP gelesen und in den Cache für lokale Datenbank-, Knoten- und DCS-Verzeichnisse gestellt.

Anmerkung: Die Registrierdatenbankvariable DB2LDAPCACHE gilt nur für die Datenbank- und Knotenverzeichnisse.

Sicherheitsaspekte in einer LDAP-Umgebung

Vor dem Zugriff auf Informationen im LDAP-Verzeichnis wird eine Anwendung oder ein Benutzer vom LDAP-Server authentifiziert. Die Authentifizierung wird als *Binden* an den LDAP-Server bezeichnet. Für die im LDAP-Verzeichnis gespeicherten Informationen muss eine Zugriffssteuerung angewendet werden, um zu verhindern, dass anonyme Benutzer Informationen löschen, ändern oder hinzufügen.

Die Zugriffssteuerung wird standardmäßig übernommen und kann auf Containerebene angewendet werden. Wenn ein neues Objekt erstellt wird, übernimmt es dasselbe Sicherheitsattribut wie das übergeordnete Objekt. Ein für den LDAP-Server verfügbares Verwaltungstool kann zum Definieren der Zugriffssteuerung für das Containerobjekt verwendet werden.

Standardmäßig wird die Zugriffssteuerung wie folgt definiert:

- Auf Datenbank- und Knoteneinträge in LDAP hat jeder (auch jeder anonyme) Benutzer Lesezugriff. Nur der Verzeichnisadministrator und der Eigner oder Ersteller des Objekts haben Schreib-/Lesezugriff.
- Auf Benutzerprofile haben der Profileigner und der Verzeichnisadministrator Schreib-/Lesezugriff. Ein Benutzer kann nicht auf das Profil eines anderen Benutzers zugreifen, wenn er keine Verzeichnisadministratorberechtigung hat.

Anmerkung: Die Berechtigungsprüfung wird immer vom LDAP-Server und nicht von DB2 durchgeführt. Die LDAP-Berechtigungsprüfung ist unabhängig von der DB2-Berechtigung. Ein Konto oder eine Berechtigungs-ID mit der Berechtigung SYSADM hat möglicherweise keinen Zugriff auf das LDAP-Verzeichnis.

Wird bei der Ausführung der LDAP-Befehle oder -APIs kein registrierter Bindenamen (bindDN) und kein Kennwort angegeben, führt DB2 eine Bindeoperation mit dem LDAP-Server durch und verwendet dabei die Standardidentitätsnachweise, denen möglicherweise nicht ausreichende Berechtigungen zur Ausführung der angeforderten Befehle zugeordnet wurden. In diesem Fall wird ein Fehler zurückgegeben.

Sie können den registrierten Bindenamen (bindDN) und das Kennwort eines Benutzers mit den Klauseln USER und PASSWORD der DB2-Befehle und -APIs explizit angeben.

Von DB2 verwendete LDAP-Objektklassen und -Attribute

In den folgenden Tabellen werden die vom DB2-Datenbankmanager verwendeten Objektklassen beschrieben:

Tabelle 11. cimManagedElement

Objektklasse	**cimManagedElement**
Active Directory-LDAP-Anzeigename	Nicht verfügbar
Allgemeiner Active Directory-Name (cn)	Nicht verfügbar
Beschreibung	Stellt eine Basisklasse für viele der Systemverwaltungsobjektklassen im IBM Schema bereit.
Unterklasse von (SubClassOf)	top
Erforderliche(s) Attribut(e)	
Optionale(s) Attribut(e)	description
Typ	abstract
OID (Objekt-ID)	1.3.18.0.2.6.132
GUID (globale eindeutige Kennung)	b3afd63f-5c5b-11d3-b818-002035559151

Tabelle 12. cimSetting

Objektklasse	**cimSetting**
Active Directory-LDAP-Anzeigename	Nicht verfügbar
Allgemeiner Active Directory-Name (cn)	Nicht verfügbar
Beschreibung	Stellt eine Basisklasse für die Konfiguration und Einstellungen im IBM Schema zur Verfügung.
Unterklasse von (SubClassOf)	cimManagedElement
Erforderliche(s) Attribut(e)	

Tabelle 12. cimSetting (Forts.)

Objektklasse	**cimSetting**
Optionale(s) Attribut(e)	settingID
Typ	abstract
OID (Objekt-ID)	1.3.18.0.2.6.131
GUID (globale eindeutige Kennung)	b3afd64d-5c5b-11d3-b818-002035559151

Tabelle 13. eProperty

Objektklasse	**eProperty**
Active Directory-LDAP-Anzeigename	ibm-eProperty
Allgemeiner Active Directory-Name (cn)	ibm-eProperty
Beschreibung	Wird zur Angabe von anwendungsspezifischen Einstellungen für Benutzervorgabemerkmale verwendet.
Unterklasse von (SubClassOf)	cimSetting
Erforderliche(s) Attribut(e)	
Optionale(s) Attribut(e)	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
Typ	structural
OID (Objekt-ID)	1.3.18.0.2.6.90
GUID (globale eindeutige Kennung)	b3afd69c-5c5b-11d3-b818-002035559151

Tabelle 14. DB2Node

Objektklasse	**DB2Node**
Active Directory-LDAP-Anzeigename	ibm-db2Node
Allgemeiner Active Directory-Name (cn)	ibm-db2Node
Beschreibung	Stellt einen DB2-Server dar.
Unterklasse von (SubClassOf)	eSap / ServiceConnectionPoint
Erforderliche(s) Attribut(e)	db2nodeName
Optionale(s) Attribut(e)	db2nodeAlias db2instanceName db2Type host / dNSHostName (siehe Anm. 2) protocolInformation/ServiceBindingInformation
Typ	structural
OID (Objekt-ID)	1.3.18.0.2.6.116

Tabelle 14. DB2Node (Forts.)

Objektklasse	**DB2Node**
GUID (globale eindeutige Kennung)	b3afd65a-5c5b-11d3-b818-002035559151
Besondere Anmerkungen	1. Die Klasse *DB2Node* wird von der Objektklasse *eSap* unter IBM Tivoli Directory Server und von der Objektklasse *ServiceConnectionPoint* unter Microsoft Active Directory abgeleitet. 2. Das Attribut *host* wird in der IBM Tivoli Directory Server-Umgebung verwendet. Das Attribut *dNSHostName* wird unter Microsoft Active Directory verwendet. 3. Das Attribut *protocolInformation* wird nur in der IBM Tivoli Directory Server-Umgebung verwendet. Im Microsoft Active Directory wird das Attribut *ServiceBindingInformation*, das von der Klasse 'ServiceConnectionPoint' übernommen wird, zur Speicherung der Protokollinformationen verwendet.

Das Attribut *protocolInformation* (in IBM Tivoli Directory Server) oder *ServiceBindingInformation* (in Microsoft Active Directory) im Objekt *DB2Node* enthält die Informationen zum Kommunikationsprotokoll für das Binden des DB2-Datenbankservers. Es besteht aus Token, die das unterstützte Netzwerkprotokoll beschreiben. Die Token werden jeweils durch ein Semikolon getrennt. Zwischen den Token steht kein Leerzeichen. Zur Angabe eines wahlfreien Parameters wird ein Stern (*) benutzt.

Es gibt folgende Token für TCP/IP:

- „TCPIP"
- Server-Hostname oder IP-Adresse
- Servicename (svcename) oder Portnummer (z. B. 50000)
- (Optional) Sicherheit („NONE" oder „SOCKS")

Es gibt folgende Token für benannte Pipes:

- „NPIPE"
- Computername des Servers
- Instanzname des Servers

Tabelle 15. DB2Database

Objektklasse	**DB2Database**
Active Directory-LDAP-Anzeigename	ibm-db2Database
Allgemeiner Active Directory-Name (cn)	ibm-db2Database
Beschreibung	Stellt eine DB2-Datenbank dar.
Unterklasse von (SubClassOf)	top
Erforderliche(s) Attribut(e)	db2databaseName db2nodePtr

Tabelle 15. DB2Database (Forts.)

Objektklasse	**DB2Database**
Optionale(s) Attribut(e)	db2databaseAlias db2additionalParameters db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName db2altgwPtr db2altnodePtr
Typ	structural
OID (Objekt-ID)	1.3.18.0.2.6.117
GUID (globale eindeutige Kennung)	b3afd659-5c5b-11d3-b818-002035559151

Tabelle 16. db2additionalParameters

Attribut	**db2additionalParameters**
Active Directory-LDAP-Anzeigename	ibm-db2AdditionalParameters
Allgemeiner Active Directory-Name (cn)	ibm-db2AdditionalParameters
Beschreibung	Enthält alle zusätzlichen Parameter, die zum Herstellen einer Verbindung zum Hostdatenbankserver verwendet werden.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.426
GUID (globale eindeutige Kennung)	b3afd315-5c5b-11d3-b818-002035559151

Tabelle 17. db2authenticationLocation

Attribut	**db2authenticationLocation**
Active Directory-LDAP-Anzeigename	ibm-db2AuthenticationLocation
Allgemeiner Active Directory-Name (cn)	ibm-db2AuthenticationLocation
Beschreibung	Gibt an, wo die Authentifizierung stattfindet.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.425
GUID (globale eindeutige Kennung)	b3afd317-5c5b-11d3-b818-002035559151
Anmerkungen	Zulässige Werte sind: CLIENT, SERVER, DCS, DCE, KERBEROS, SVRENCRYPT oder DCSENCRYPT

Tabelle 18. db2ARLibrary

Attribut	**db2ARLibrary**
Active Directory-LDAP-Anzeigename	ibm-db2ARLibrary
Allgemeiner Active Directory-Name (cn)	ibm-db2ARLibrary
Beschreibung	Name der Anwendungsrequesterbibliothek
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	256
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.427
GUID (globale eindeutige Kennung)	b3afd316-5c5b-11d3-b818-002035559151

Tabelle 19. db2databaseAlias

Attribut	**db2databaseAlias**
Active Directory-LDAP-Anzeigename	ibm-db2DatabaseAlias
Allgemeiner Active Directory-Name (cn)	ibm-db2DatabaseAlias
Beschreibung	Aliasname(n) der Datenbank
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.422
GUID (globale eindeutige Kennung)	b3afd318-5c5b-11d3-b818-002035559151

Tabelle 20. db2databaseName

Attribut	**db2databaseName**
Active Directory-LDAP-Anzeigename	ibm-db2DatabaseName
Allgemeiner Active Directory-Name (cn)	ibm-db2DatabaseName
Beschreibung	Datenbankname
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.421
GUID (globale eindeutige Kennung)	b3afd319-5c5b-11d3-b818-002035559151

Tabelle 21. db2databaseRelease

Attribut	**db2databaseRelease**
Active Directory-LDAP-Anzeigename	ibm-db2DatabaseRelease
Allgemeiner Active Directory-Name (cn)	ibm-db2DatabaseRelease
Beschreibung	Datenbank-Release-Nummer
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.429

Tabelle 21. db2databaseRelease (Forts.)

Attribut	**db2databaseRelease**
GUID (globale eindeutige Kennung)	b3afd31a-5c5b-11d3-b818-002035559151

Tabelle 22. db2nodeAlias

Attribut	**db2nodeAlias**
Active Directory-LDAP-Anzeigename	ibm-db2NodeAlias
Allgemeiner Active Directory-Name (cn)	ibm-db2NodeAlias
Beschreibung	Aliasname(n) der Knoten
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.420
GUID (globale eindeutige Kennung)	b3afd31d-5c5b-11d3-b818-002035559151

Tabelle 23. db2nodeName

Attribut	**db2nodeName**
Active Directory-LDAP-Anzeigename	ibm-db2NodeName
Allgemeiner Active Directory-Name (cn)	ibm-db2NodeName
Beschreibung	Knotenname
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.419
GUID (globale eindeutige Kennung)	b3afd31e-5c5b-11d3-b818-002035559151

Tabelle 24. db2nodePtr

Attribut	**db2nodePtr**
Active Directory-LDAP-Anzeigename	ibm-db2NodePtr
Allgemeiner Active Directory-Name (cn)	ibm-db2NodePtr
Beschreibung	Zeiger auf das Knotenobjekt (DB2Node) für den Datenbankserver, der Eigner der Datenbank ist
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.423
GUID (globale eindeutige Kennung)	b3afd31f-5c5b-11d3-b818-002035559151
Besondere Anmerkungen	Diese Abhängigkeit erlaubt es dem Client, die erforderlichen Informationen zum Übertragungsprotokoll abzurufen, um eine Verbindung zur Datenbank herzustellen.

Tabelle 25. db2altnodePtr

Attribut	**db2altnodePtr**
Active Directory-LDAP-Anzeigename	ibm-db2AltNodePtr
Allgemeiner Active Directory-Name (cn)	ibm-db2AltNodePtr
Beschreibung	Zeiger auf das Knotenobjekt (DB2Node), das den alternativen Datenbankserver darstellt
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.3093
GUID (globale eindeutige Kennung)	5694e266-2059-4e32-971e-0778909e0e72

Tabelle 26. db2gwPtr

Attribut	**db2gwPtr**
Active Directory-LDAP-Anzeigename	ibm-db2GwPtr
Allgemeiner Active Directory-Name (cn)	ibm-db2GwPtr
Beschreibung	Zeiger auf das Knotenobjekt für den Gateway-Server, über den auf die Datenbank zugegriffen werden kann
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.424
GUID (globale eindeutige Kennung)	b3afd31b-5c5b-11d3-b818-002035559151

Tabelle 27. db2altgwPtr

Attribut	**db2altgwPtr**
Active Directory-LDAP-Anzeigename	ibm-db2AltGwPtr
Allgemeiner Active Directory-Name (cn)	ibm-db2AltGwPtr
Beschreibung	Zeiger auf das Knotenobjekt, das den alternativen Gateway-Server darstellt
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.3092
GUID (globale eindeutige Kennung)	70ab425d-65cc-4d7f-91d8-084888b3a6db

Tabelle 28. db2instanceName

Attribut	**db2instanceName**
Active Directory-LDAP-Anzeigename	ibm-db2InstanceName
Allgemeiner Active Directory-Name (cn)	ibm-db2InstanceName
Beschreibung	Name der Datenbankserverinstanz
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	256

Tabelle 28. db2instanceName (Forts.)

Attribut	**db2instanceName**
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.428
GUID (globale eindeutige Kennung)	b3afd31c-5c5b-11d3-b818-002035559151

Tabelle 29. db2Type

Attribut	**db2Type**
Active Directory-LDAP-Anzeigename	ibm-db2Type
Allgemeiner Active Directory-Name (cn)	ibm-db2Type
Beschreibung	Datenbankservertyp
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.418
GUID (globale eindeutige Kennung)	b3afd320-5c5b-11d3-b818-002035559151
Anmerkungen	Zulässige Datenbankservertypen: SERVER, MPP und DCS

Tabelle 30. DCEPrincipalName

Attribut	**DCEPrincipalName**
Active Directory-LDAP-Anzeigename	ibm-DCEPrincipalName
Allgemeiner Active Directory-Name (cn)	ibm-DCEPrincipalName
Beschreibung	DCE-Principal-Name
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	2048
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.443
GUID (globale eindeutige Kennung)	b3afd32d-5c5b-11d3-b818-002035559151

Tabelle 31. cesProperty

Attribut	**cesProperty**
Active Directory-LDAP-Anzeigename	ibm-cesProperty
Allgemeiner Active Directory-Name (cn)	ibm-cesProperty
Beschreibung	Werte dieses Attributs können zur Bereitstellung anwendungsspezifischer Vorgabekonfigurationsparameter genutzt werden. Ein Wert kann zum Beispiel Daten im XML-Format enthalten. Alle Werte dieses Attributs müssen einen einheitlichen Attributwert für cesPropertyType aufweisen.
Syntax	Zeichenfolge mit genauer Beachtung der Groß-/Kleinschreibung
Maximale Länge	32700
Werte	Mit mehreren Werten

Tabelle 31. cesProperty (Forts.)

Attribut	**cesProperty**
OID (Objekt-ID)	1.3.18.0.2.4.307
GUID (globale eindeutige Kennung)	b3afd2d5-5c5b-11d3-b818-002035559151

Tabelle 32. cesPropertyType

Attribut	**cesPropertyType**
Active Directory-LDAP-Anzeigename	ibm-cesPropertyType
Allgemeiner Active Directory-Name (cn)	ibm-cesPropertyType
Beschreibung	Werte dieses Attributs können zum Beschreiben der Syntax, Semantik oder anderer Merkmale aller Werte des Attributs cesProperty verwendet werden. Der Wert „XML" kann zum Beispiel verwendet werden, um anzugeben, dass alle Werte des Attributs cesProperty in der XML-Syntax codiert sind.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.308
GUID (globale eindeutige Kennung)	b3afd2d6-5c5b-11d3-b818-002035559151

Tabelle 33. cisProperty

Attribut	**cisProperty**
Active Directory-LDAP-Anzeigename	ibm-cisProperty
Allgemeiner Active Directory-Name (cn)	ibm-cisProperty
Beschreibung	Werte dieses Attributs können zur Bereitstellung anwendungsspezifischer Vorgabekonfigurationsparameter genutzt werden. Ein Wert kann zum Beispiel eine INI-Datei enthalten. Alle Werte dieses Attributs müssen einen einheitlichen Attributwert für cisPropertyType aufweisen.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	32700
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.309
GUID (globale eindeutige Kennung)	b3afd2e0-5c5b-11d3-b818-002035559151

Tabelle 34. cisPropertyType

Attribut	**cisPropertyType**
Active Directory-LDAP-Anzeigename	ibm-cisPropertyType
Allgemeiner Active Directory-Name (cn)	ibm-cisPropertyType
Beschreibung	Werte dieses Attributs können zum Beschreiben der Syntax, Semantik oder anderer Merkmale aller Werte des Attributs cisProperty verwendet werden. Der Wert „INI File" kann zum Beispiel verwendet werden, um anzugeben, dass alle Werte des Attributs cisProperty INI-Dateien sind.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge

Tabelle 34. cisPropertyType (Forts.)

Attribut	**cisPropertyType**
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.310
GUID (globale eindeutige Kennung)	b3afd2e1-5c5b-11d3-b818-002035559151

Tabelle 35. binProperty

Attribut	**binProperty**
Active Directory-LDAP-Anzeigename	ibm-binProperty
Allgemeiner Active Directory-Name (cn)	ibm-binProperty
Beschreibung	Werte dieses Attributs können zur Bereitstellung anwendungsspezifischer Vorgabekonfigurationsparameter genutzt werden. Ein Wert kann z. B. eine Gruppe binär codierter Lotus 1-2-3-Merkmale enthalten. Alle Werte dieses Attributs müssen einen einheitlichen Attributwert für binPropertyType aufweisen.
Syntax	binär
Maximale Länge	250000
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.305
GUID (globale eindeutige Kennung)	b3afd2ba-5c5b-11d3-b818-002035559151

Tabelle 36. binPropertyType

Attribut	**binPropertyType**
Active Directory-LDAP-Anzeigename	ibm-binPropertyType
Allgemeiner Active Directory-Name (cn)	ibm-binPropertyType
Beschreibung	Werte dieses Attributs können zum Beschreiben der Syntax, Semantik oder anderer Merkmale aller Werte des Attributs binProperty verwendet werden. Der Wert „Lotus 123" kann zum Beispiel verwendet werden, um anzugeben, dass alle Werte des Attributs 'binProperty' binär codierte Lotus 123-Merkmale sind.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.306
GUID (globale eindeutige Kennung)	b3afd2bb-5c5b-11d3-b818-002035559151

Tabelle 37. PropertyType

Attribut	**PropertyType**
Active Directory-LDAP-Anzeigename	ibm-propertyType
Allgemeiner Active Directory-Name (cn)	ibm-propertyType
Beschreibung	Werte dieses Attributs beschreiben die semantischen Merkmale des eProperty-Objekts.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge

Tabelle 37. PropertyType (Forts.)

Attribut	PropertyType
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.320
GUID (globale eindeutige Kennung)	b3afd4ed-5c5b-11d3-b818-002035559151

Tabelle 38. settingID

Attribut	settingID
Active Directory-LDAP-Anzeigename	Nicht verfügbar
Allgemeiner Active Directory-Name (cn)	Nicht verfügbar
Beschreibung	Ein Benennungsattribut, das zum Identifizieren der von cimSetting abgeleiteten Objekteinträge, zum Beispiel eProperty, dient.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	256
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.325
GUID (globale eindeutige Kennung)	b3afd596-5c5b-11d3-b818-002035559151

Erweitern des LDAP-Verzeichnisschemas mit DB2-Objektklassen und -Attributen

Das LDAP-Verzeichnisschema definiert Objektklassen und Attribute für die in den LDAP-Verzeichniseinträgen gespeicherten Informationen. Eine Objektklasse besteht aus einer Reihe von verbindlichen und optionalen Attributen. Jedem Eintrag im LDAP-Verzeichnis ist eine Objektklasse zugeordnet.

Bevor der DB2-Datenbankmanager Informationen in LDAP speichern kann, muss das Verzeichnisschema für den LDAP-Server die vom DB2-Datenbanksystem verwendeten Objektklassen und Attribute enthalten. Das Hinzufügen neuer Objektklassen und Attribute zum Basisschema wird als *Schemaerweiterung* bezeichnet.

Unterstützte LDAP-Client- und -Serverkonfigurationen

Die folgende Tabelle bietet eine Übersicht über die unterstützten LDAP-Client- und -Serverkonfigurationen.

Tabelle 39. Unterstützte LDAP-Client- und -Serverkonfigurationen

	LDAP-Server		
LDAP-Clients	**IBM Tivoli Directory Server [1]**	**Microsoft Active Directory-Server [2]**	**Sun One LDAP-Server**
IBM LDAP Client [3]	Unterstützt	Unterstützt	Unterstützt
Microsoft-LDAP/ ADSI- Client[4]	Unterstützt	Unterstützt	Unterstützt

- [1] IBM Tivoli Directory Server ist ein LDAP-Server der Version 3 und für Windows, AIX, Solaris, Linux und HP-UX verfügbar. Er wird als Teil des Basisbetriebssystems für AIX und System i sowie mit OS/390 Security Server geliefert.
- [2] Microsoft Active Directory-Server ist ein LDAP-Server der Version 3 und als Teil der Betriebssystemfamilien Windows 2000 Server und Windows Server 2003 verfügbar.
- [3] Das DB2-Datenbanksystem unterstützt IBM LDAP Client unter AIX, Solaris, HP-UX 11.11, Windows und Linux.
- [4] Der LDAP-Client von Microsoft ist in das Windows-Betriebssystem integriert.

Anmerkung: Bei der Ausführung unter Windows-Betriebssystemen unterstützt der DB2-Datenbankmanager die Verwendung des IBM LDAP-Clients oder des Microsoft-LDAP-Clients. Zur expliziten Auswahl des IBM LDAP-Clients setzen Sie mit dem Befehl db2set die Registrierdatenbankvariable **DB2LDAP_CLIENT_PROVIDER** auf den Wert IBM. Der LDAP-Client von Microsoft ist in das Windows-Betriebssystem integriert.

LDAP-Unterstützung und DB2 Connect

Wenn die LDAP-Unterstützung auf dem DB2 Connect-Gateway zur Verfügung steht und die Datenbank im Datenbankverzeichnis auf dem Gateway nicht gefunden werden kann, sucht der DB2-Datenbankmanager die Datenbankposition in LDAP und versucht, die gefundenen Informationen zu speichern.

Registrieren von Hostdatenbanken in LDAP

Wenn Sie Hostdatenbanken in LDAP registrieren, sind zwei Konfigurationen möglich: eine Direktverbindung zu den Hostdatenbanken oder eine Verbindung zur Hostdatenbank über ein Gateway.

Für eine Direktverbindung zu den Hostdatenbanken registrieren Sie den Host-Server in LDAP und katalogisieren anschließend die Hostdatenbank in LDAP, indem Sie den Knotennamen des Host-Servers angeben. Für eine Verbindung zur Hostdatenbank über ein Gateway registrieren Sie den Gateway-Server in LDAP und katalogisieren anschließend die Hostdatenbank in LDAP, indem Sie den Knotennamen des Gateway-Servers angeben.

Wenn die LDAP-Unterstützung auf dem DB2 Connect-Gateway zur Verfügung steht und die Datenbank im Datenbankverzeichnis auf dem Gateway nicht gefunden wird, sucht das DB2-Datenbanksystem in LDAP und versucht, die gefundenen Informationen zu speichern.

Das folgende Beispiel veranschaulicht beide Fälle: Nehmen Sie an, es ist eine Hostdatenbank mit dem Namen NIAGARA_FALLS vorhanden. Diese Datenbank kann eingehende Verbindungen über TCP/IP akzeptieren. Wenn der Client keine direkte Verbindung zum Host herstellen kann, weil er nicht über DB2 Connect verfügt, stellt er die Verbindung über ein Gateway mit dem Namen goto@niagara her.

Die folgenden Arbeitsschritte müssen ausgeführt werden:

1. Der Hostdatenbankserver muss in LDAP für TCP/IP-Konnektivität registriert werden. Der TCP/IP-Hostname des Servers ist ″myhost″ und die Portnummer ist ″446″. Die Klausel **NODETYPE** wird auf den Wert DCS gesetzt, um anzugeben, dass es sich um einen Hostdatenbankserver handelt.

```
db2 register ldap as nftcpip tcpip hostname myhost svcename 446
   remote mvssys instance mvsinst nodetype dcs
```

2. Ein DB2 Connect-Gateway-Server muss in LDAP für TCP/IP-Konnektivität registriert werden. Der TCP/IP-Hostname für den Gateway-Server ist ″niagara″ und die Portnummer ist ″50000″.

   ```
   db2 register ldap as whasf tcpip hostname niagara svcename 50000
      remote niagara instance goto nodetype server
   ```

3. Die Hostdatenbank muss in LDAP mit TCP/IP-Konnektivität katalogisiert werden. Der Name der Hostdatenbank ist ″NIAGARA_FALLS″, der Aliasname der Datenbank ″nftcpip″. Die Klausel **GWNODE** wird verwendet, um den Knotennamen des DB2 Connect-Gateway-Servers anzugeben.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nftcpip at node nftcpip
      gwnode whasf authentication server
   ```

Nach Abschluss der oben gezeigten Registrierung und Katalogisierung können Sie eine Verbindung zu 'nftcpip' herstellen, wenn Sie über TCP/IP auf den Host zugreifen wollen. Wenn auf Ihrer Client-Workstation DB2 Connect nicht installiert ist, wird die Verbindung mit TCP/IP über das Gateway hergestellt. Vom Gateway aus wird die Verbindung zum Host über TCP/IP hergestellt.

Sie können Angaben zur Hostdatenbank in LDAP normalerweise manuell konfigurieren, sodass die Clients die Datenbank und den Knoten nicht lokal auf jeder Maschine katalogisieren müssen. Gehen Sie dazu wie folgt vor:

1. Registrieren Sie den Hostdatenbankserver in LDAP. Dazu müssen Sie den Namen des fernen Computers, den Instanznamen und den Knotentyp für den Hostdatenbankserver im Befehl REGISTER angeben, indem Sie die Klauseln **REMOTE**, **INSTANCE** und **NODETYPE** verwenden. Die Klausel **REMOTE** kann entweder auf den Hostnamen oder den LU-Namen der Hostservermaschine gesetzt werden. Die Klausel **INSTANCE** kann auf eine beliebige Zeichenfolge von höchsten acht Zeichen gesetzt werden. (Der Instanzname kann z. B. auf ″DB2″ gesetzt werden.) Die Klausel **NODETYPE** muss auf den Wert DCS gesetzt werden, um anzugeben, dass es sich um einen Hostdatenbankserver handelt.
2. Registrieren Sie die Hostdatenbank in LDAP mit dem Befehl CATALOG LDAP DATABASE. Weitere DRDA-Parameter können mit der Klausel **PARMS** angegeben werden. Der Datenbankauthentifizierungstyp sollte auf SERVER gesetzt werden.

Erweitern des Verzeichnisschemas für IBM Tivoli Directory Server

Wenn Sie IBM Tivoli Directory Server verwenden, sind alle für die DB2-Datenbank vor Version 8.2 erforderlichen Objektklassen und Attribute im Basisschema enthalten.

Führen Sie den folgenden Befehl aus, um das Basisschema mit den neuen, seit Version 8.2 eingeführten DB2-Datenbankattributen zu erweitern:

```
ldapmodify -c -h maschinenname:389 -D dn -w kennwort -f altgwnode.ldif
```

Die Datei altgwnode.ldif besitzt folgenden Inhalt:

```
dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3092
 NAME 'db2altgwPtr'
 DESC 'DN pointer to DB2 alternate gateway (node) object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3092
 DBNAME ('db2altgwPtr' 'db2altgwPtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3093
 NAME 'db2altnodePtr'
 DESC 'DN pointer to DB2 alternate node object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3093
 DBNAME ('db2altnodePtr' 'db2altnodePtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:            cn=schema
changetype:    modify
replace:       objectclasses
objectclasses: (
 1.3.18.0.2.6.117
 NAME 'DB2Database'
 DESC 'DB2 database'
 SUP cimSetting
 MUST ( db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr
    $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias
    $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName ) )
```

Die Dateien altgwnode.ldif und altgwnode.readme stehen unter folgender URL-Adresse zur Verfügung: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

Nach dem Hinzufügen der DB2-Schemadefinition muss der Directory Server erneut gestartet werden, um alle Änderungen in Kraft zu setzen.

Netscape-Unterstützung und Attributdefinitionen für das LDAP-Verzeichnis

Der unterstützte Stand für Netscape LDAP Server ist Version 4.12 oder eine spätere Version.

In Netscape LDAP Server Version 4.12 (oder später) ermöglicht der Netscape Directory Server Anwendungen, das Schema durch Hinzufügen von Attribut- und Objektklassendefinitionen in den beiden Dateien slapd.user_oc.conf und

slapd.user_at.conf zu erweitern. Diese beiden Dateien befinden sich im Verzeichnis <Netscape_installationspfad>\slapd-<maschinenname>\config.

Anmerkung: Wenn Sie Sun One Directory Server 5.0 verwenden, finden Sie im Abschnitt über das Erweitern des Verzeichnisschemas für Sun One Directory Server entsprechende Informationen.

Die DB2-Attribute müssen der Datei slapd.user_at.conf wie folgt hinzugefügt werden:

```
########################################################################
#
# IBM DB2 Database
# Attributdefinitionen
#
# bin -> binär
# ces -> Zeichenfolge in exakter Groß-/Kleinschreibung
# cis -> Zeichenfolge ohne Unterscheidung der Groß-/Kleinschreibung
# dn -> registrierter Name (Distinguished Name)
#
########################################################################

attribute binProperty                    1.3.18.0.2.4.305     bin
attribute binPropertyType                1.3.18.0.2.4.306     cis
attribute cesProperty                    1.3.18.0.2.4.307     ces
attribute cesPropertyType                1.3.18.0.2.4.308     cis
attribute cisProperty                    1.3.18.0.2.4.309     cis
attribute cisPropertyType                1.3.18.0.2.4.310     cis
attribute propertyType                   1.3.18.0.2.4.320     cis
attribute systemName                     1.3.18.0.2.4.329     cis
attribute db2nodeName                    1.3.18.0.2.4.419     cis
attribute db2nodeAlias                   1.3.18.0.2.4.420     cis
attribute db2instanceName                1.3.18.0.2.4.428     cis
attribute db2Type                        1.3.18.0.2.4.418     cis
attribute db2databaseName                1.3.18.0.2.4.421     cis
attribute db2databaseAlias               1.3.18.0.2.4.422     cis
attribute db2nodePtr                     1.3.18.0.2.4.423     dn
attribute db2gwPtr                       1.3.18.0.2.4.424     dn
attribute db2additionalParameters        1.3.18.0.2.4.426     cis
attribute db2ARLibrary                   1.3.18.0.2.4.427     cis
attribute db2authenticationLocation      1.3.18.0.2.4.425     cis
attribute db2databaseRelease             1.3.18.0.2.4.429     cis
attribute DCEPrincipalName               1.3.18.0.2.4.443     cis
```

Die DB2-Objektklassen müssen der Datei slapd.user_oc.conf wie folgt hinzugefügt werden:

```
########################################################################
#
# IBM DB2 Database
# Objektklassendefinitionen
#
########################################################################

objectclass eProperty
        oid 1.3.18.0.2.6.90
        requires
                objectClass
        allows
                cn,
                propertyType,
                binProperty,
                binPropertyType,
                cesProperty,
                cesPropertyType,
                cisProperty,
```

```
                cisPropertyType

objectclass eApplicationSystem
        oid 1.3.18.0.2.6.84
        requires
                objectClass,
                systemName


objectclass DB2Node
        oid 1.3.18.0.2.6.116
        requires
                objectClass,
                db2nodeName
        allows
                db2nodeAlias,
                host,
                db2instanceName,
                db2Type,
                description,
                protocolInformation

objectclass DB2Database
        oid 1.3.18.0.2.6.117
        requires
                objectClass,
                db2databaseName,
                db2nodePtr
        allows
                db2databaseAlias,
                description,
                db2gwPtr,
                db2additionalParameters,
                db2authenticationLocation,
                DCEPrincipalName,
                db2databaseRelease,
                db2ARLibrary
```

Nach dem Hinzufügen der DB2-Schemadefinition muss der Directory Server erneut gestartet werden, um alle Änderungen in Kraft zu setzen.

Erweitern des Verzeichnisschemas für Sun One Directory Server

Sun One Directory Server wird auch als Netscape- oder iPlanet-Verzeichnisserver bezeichnet.

Zur funktionsfähigen Einrichtung von Sun One Directory Server in Ihrer Umgebung fügen Sie die Datei 60ibmdb2.ldif dem folgenden Verzeichnis hinzu:

Unter Windows, wenn iPlanet im Verzeichnis C:\iPlanet\Servers installiert ist, fügen Sie die oben genannte Datei dem Verzeichnis .\sldap-<maschinenname>\config\schema hinzu.

Unter UNIX, wenn iPlanet im Verzeichnis /usr/iplanet/servers installiert ist, fügen Sie die oben genannte Datei dem Verzeichnis ./slapd-<maschinenname>/config/schema hinzu.

Die Datei besitzt folgenden Inhalt:

```
########################################################################
# IBM DB2 Database
########################################################################
dn: cn=schema
```

```
#########################################################################
# Attributdefinitionen (vor Version 8.2)
#########################################################################
attributetypes: ( 1.3.18.0.2.4.305  NAME 'binProperty'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.5              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.306  NAME 'binPropertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.307  NAME 'cesProperty'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.308  NAME 'cesPropertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.309  NAME 'cisProperty'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.310  NAME 'cisPropertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.320  NAME 'propertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.329  NAME 'systemName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.419  NAME 'db2nodeName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.420  NAME 'db2nodeAlias'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.428  NAME 'db2instanceName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.418  NAME 'db2Type'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.421  NAME 'db2databaseName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.422  NAME 'db2databaseAlias'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.426  NAME 'db2additionalParameters'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.427  NAME 'db2ARLibrary'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.425  NAME 'db2authenticationLocation'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.429  NAME 'db2databaseRelease'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.443  NAME 'DCEPrincipalName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.423  NAME 'db2nodePtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.424  NAME 'db2gwPtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
#########################################################################
# Attributdefinitionen (Version 8.2 und spätere Versionen)
#########################################################################
attributetypes: ( 1.3.18.0.2.4.3092 NAME 'db2altgwPtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.3093 NAME 'db2altnodePtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12             X-ORIGIN 'IBM DB2' )
#########################################################################
# Objektklassendefinitionen
# DB2 Database Version 8.2 hat die obigen beiden neuen optionalen Attribute.
#########################################################################
objectClasses: ( 1.3.18.0.2.6.90  NAME 'eProperty'
   SUP top STRUCTURAL MAY ( cn $ propertyType $ binProperty
   $ binPropertyType $ cesProperty $ cesPropertyType $ cisProperty
   $ cisPropertyType ) X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.84  NAME 'eApplicationSystem'
   SUP top STRUCTURAL MUST systemName
   X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.116 NAME 'DB2Node'
   SUP top STRUCTURAL MUST db2nodeName MAY ( db2instanceName $ db2nodeAlias
   $ db2Type $ description $ host $ protocolInformation )
   X-ORIGIN 'IBM DB2' )
```

```
objectClasses: ( 1.3.18.0.2.6.117 NAME 'DB2Database'
   SUP top STRUCTURAL MUST (db2databaseName $ db2nodePtr ) MAY
   ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary
   $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease
   $ db2gwPtr $ DCEPrincipalName $ description )
   X-ORIGIN 'IBM DB2' )
```

Die Dateien 60ibmdb2.ldif und 60ibmdb2.readme stehen unter folgender URL-Adresse zur Verfügung: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

Nach dem Hinzufügen der DB2-Schemadefinition muss der Directory Server erneut gestartet werden, um alle Änderungen in Kraft zu setzen.

Windows Active Directory

Die DB2-Datenbankserver werden in Active Directory als ibm_db2Node-Objekte bereitgestellt. Die Objektklasse ibm_db2Node ist eine Unterklasse der Objektklasse ServiceConnectionPoint (SCP).

Jedes ibm_db2Node-Objekt enthält Protokollkonfigurationsdaten, die es Clientanwendungen ermöglichen, eine Verbindung zum DB2-Datenbankserver herzustellen. Beim Erstellen einer neuen Datenbank wird diese in Active Directory als ibm_db2Database-Objekt unter dem ibm_db2Node-Objekt bereitgestellt.

Beim Herstellen einer Verbindung zu einer fernen Datenbank fragt der DB2-Client Active Directory über die LDAP-Schnittstelle nach dem ibm_db2Database-Objekt ab. Die Daten zur Protokollkommunikation zur Herstellung der Verbindung zum Datenbankserver (Bindeinformationen) werden aus dem ibm_db2Node-Objekt abgerufen, unter dem das ibm_db2Database-Objekt erstellt ist.

Eigenschaftenseiten für die ibm_db2Node- und ibm_db2Database-Objekte können über das Snap-in *Active Directory-Benutzer und -Computer* der Managementkonsole (MMC) auf einem Domänencontroller angezeigt werden. Zur Einrichtung der Eigenschaftenseite führen Sie den Befehl regsrv32 wie folgt aus, um die Eigenschaftenseiten für die DB2-Objekte zu registrieren:

```
regsvr32 %DB2PATH%\bin\db2ads.dll
```

Sie können die Objekte über das Snap-in *Active Directory-Benutzer und -Computer* der Managementkonsole (MMC) auf einem Domänencontroller anzeigen. Auf dieses Verwaltungstool können Sie über **Start—> Programme—> Verwaltung—> Active Directory-Benutzer und -Computer** zugreifen.

Anmerkung: Sie müssen *Benutzer, Gruppen und Computer als Container* im Menü **Ansicht** auswählen, um die DB2-Datenbankobjekte unterhalb der Computerobjekte anzuzeigen.

Anmerkung: Wenn das DB2-Datenbanksystem nicht auf dem Domänencontroller installiert ist, können Sie die Eigenschaftenseiten von DB2-Datenbankobjekten dennoch anzeigen, indem Sie die Datei db2ads.dll aus dem Verzeichnis %DB2PATH%\bin und die Ressourcen-DLL-Datei db2adsr.dll aus dem Verzeichnis %DB2PATH%\msg\locale-name in ein lokales Verzeichnis auf dem Domänencontroller kopieren. (Das Verzeichnis, in dem Sie diese beiden kopierten Dateien ablegen, muss eines der Verzeichnisse sein, die in der Umgebungsvariablen PATH definiert sind.) Anschließend führen Sie den Befehl regsvr32 in dem lokalen Verzeichnis aus, um die DLL-Datei zu registrieren.

Konfigurieren des DB2-Datenbankmanagers zur Verwendung von Active Directory

Um auf Microsoft Active Directory zugreifen zu können, müssen die folgenden Bedingungen erfüllt sein:

1. Die Maschine, auf der das DB2-Datenbanksystem ausgeführt wird, muss zu einer Windows 2000- oder Windows Server 2003-Domäne gehören.
2. Der Microsoft-LDAP-Client wurde installiert. Der Microsoft-LDAP-Client ist Bestandteil der Betriebssysteme Windows 2000, Windows XP und Windows Server 2003.
3. Aktivieren Sie die LDAP-Unterstützung. Weitere Informationen dazu finden Sie im Abschnitt „Erweitern des Active Directory-Schemas für LDAP-Verzeichnisservices (Windows)" in *DB2-Server - Einstieg*.
4. Melden Sie sich bei der Ausführung des DB2-Datenbanksystems über ein Domänenbenutzerkonto an, um Informationen aus Active Directory zu lesen.

Sicherheitsaspekte für Active Directory

Die DB2-Datenbank- und -Knotenobjekte werden im Active Directory unter dem Computerobjekt der Maschine erstellt, auf der der DB2-Server installiert ist. Zum Registrieren eines Datenbankservers oder zum Katalogisieren einer Datenbank im Active Directory müssen Sie über ausreichende Zugriffsberechtigungen zum Erstellen oder Aktualisieren der Objekte unter dem Computerobjekt verfügen.

Standardmäßig können die unter einem Computerobjekt angelegten Objekte von allen authentifizierten Benutzern gelesen und von Administratoren (d. h. Benutzern, die zur Administrator-, Domänenadministrator- und Unternehmensadministratorgruppe gehören) aktualisiert werden. Um einem bestimmten Benutzer bzw. einer bestimmten Gruppe Zugriffsberechtigungen zu erteilen, verwenden Sie das Snap-in *Active Directory-Benutzer und -Computer* der Microsoft Management Console (MMC) wie folgt:

1. Starten Sie das Verwaltungstool **Active Directory-Benutzer und -Computer**.
   (Start—> Programme—> Verwaltung—> Active Directory-Benutzer und -Computer)
2. Wählen Sie unter **Ansicht** die Option **Erweiterte Funktionen** aus.
3. Wählen Sie den Container **Computer** aus.
4. Klicken Sie mit der rechten Maustaste das Computerobjekt an, das die Servermaschine darstellt, auf der DB2 installiert ist, und wählen Sie **Eigenschaften** aus.
5. Wählen Sie die Registerkarte **Sicherheitseinstellungen** aus und fügen Sie für die angegebene Gruppe bzw. den angegebenen Benutzer die erforderliche Zugriffsberechtigungen hinzu.

Die DB2-Registrierdatenbankvariablen und CLI-Einstellungen auf Benutzerebene werden im Objekt für DB2-Eigenschaften unter dem Benutzerobjekt verwaltet. Um die DB2-Registrierdatenbankvariablen oder CLI-Einstellungen auf Benutzerebene zu definieren, muss ein Benutzer über ausreichende Zugriffsberechtigungen verfügen, um Objekte unter dem Benutzerobjekt zu erstellen.

Standardmäßig verfügen nur Administratoren über die Zugriffsberechtigung zum Erstellen von Objekten unter dem Benutzerobjekt. Um einem Benutzer die Zugriffsberechtigung zum Definieren von DB2-Registrierdatenbankvariablen oder CLI-Einstellungen auf Benutzerebene zu erteilen, verwenden Sie das Snap-in *Active Directory-Benutzer und -Computer* der Management Console (MMC) wie folgt:

1. Starten Sie das Verwaltungstool **Active Directory-Benutzer und -Computer**. (Start—> Programme—> Verwaltung—> Active Directory-Benutzer und -Computer)
2. Wählen Sie das Benutzerobjekt im Container **Benutzer** ('Users') aus.
3. Klicken Sie mit der rechten Maustaste das Benutzerobjekt an, und wählen Sie **Eigenschaften** aus.
4. Wählen Sie die Registerkarte **Sicherheitseinstellungen** aus.
5. Fügen Sie den Benutzername mithilfe der Schaltfläche **Hinzufügen** der Liste hinzu.
6. Erteilen Sie die Zugriffsberechtigungen „Schreiben" und „Alle untergeordneten Objekte erstellen".
7. In den erweiterten Einstellungen (Schaltfläche **Erweitert**) wählen Sie im Feld **Übernehmen für** den Eintrag „Dieses und alle untergeordneten Objekte" aus.
8. Wählen Sie das Markierungsfeld „Vererbbare übergeordnete Berechtigungen übernehmen" aus.

DB2-Objekte im Active Directory

Der DB2-Datenbankmanager erstellt Objekte im Active Directory an den beiden folgenden Positionen:

1. Die DB2-Datenbank- und -Knotenobjekte werden unter dem Computerobjekt der Maschine erstellt, auf der der DB2-Server installiert ist. Für die DB2-Servermaschine, die nicht zur Windows-Domäne gehört, werden die DB2-Datenbank- und -Knotenobjekte im Container „System" erstellt.
2. Die DB2-Registrierdatenbankvariablen und CLI-Einstellungen auf der Benutzerebene werden in den DB2-Eigenschaftsobjekten unter dem Benutzerobjekt gespeichert. Diese Objekte enthalten benutzerspezifische Informationen.

Erweitern des Verzeichnisschemas für Active Directory

Bevor der DB2-Datenbankmanager Informationen im Active Directory speichern kann, muss das Verzeichnisschema so erweitert werden, dass es die neuen Objektklassen und Attribute der DB2-Datenbank enthält. Das Hinzufügen neuer Objektklassen und Attribute zum Verzeichnisschema wird als *Schemaerweiterung* bezeichnet.

Sie müssen das Schema für Active Directory durch Ausführung des DB2-Schemainstallationsprogramms db2schex erweitern. Führen Sie diesen Befehl vor der Installation der DB2-Produkte und der Erstellung der Datenbanken aus, da Sie andernfalls die Registrierung des Knotens und die Katalogisierung der Datenbanken manuell durchführen müssen.

Das Programm db2schex befindet sich auf der Produkt-CD-ROM in folgendem Verzeichnis: `x:\db2\windows\utilities\`. Dabei steht x: für das DVD-Laufwerk.

Zum Aktualisieren des Schemas müssen Sie Mitglied der Schemaadministratorgruppe sein oder über die delegierten Berechtigungen zum Aktualisieren des Schemas verfügen. Führen Sie den folgenden Befehl auf jeder Maschine aus, die Teil der Windows-Domäne ist:

```
runas /user:MyDomain\Administrator x:\db2\Windows\utilities\db2schex.exe
```

Hierbei ist x: der DVD-Laufwerkbuchstabe.

Wenn Sie den Befehl db2schex aus einer früheren Version des DB2-Datenbankverwaltungssystems ausgeführt haben und diesen Befehl erneut für DB2 UDB Ver-

sion 8.2 oder eine spätere Version ausführen, werden die beiden folgenden optionalen Attribute der Klasse 'ibm-db2Database' hinzugefügt:

```
ibm-db2AltGwPtr
ibm-db2NodePtr
```

Wenn Sie den Befehl db2schex aus einer früheren Version des DB2-Datenbankverwaltungssystems unter Windows nicht ausgeführt haben und diesen Befehl für DB2 Version 9.5 oder eine spätere Version ausführen, werden alle Klassen und Attribute für die LDAP-Unterstützung des DB2-Datenbanksystems hinzugefügt.

Für diesen Befehl sind weitere optionale Klauseln verfügbar. Weitere Informationen finden Sie im Thema „db2schex - Active Directory-Schemaerweiterung (Befehl)".

Beispiele:

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu installieren:

  ```
  db2schex
  ```

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu installieren und einen definierten Bindenamen (bindDN) und ein Kennwort anzugeben:

  ```
  db2schex -b "cn=A Name,dc=toronto1,dc=ibm,dc=com"
           -w kennwort
  ```

  Alternativ können Sie auch Folgendes eingeben:

  ```
  db2schex -b Administrator -w kennwort
  ```

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu deinstallieren:

  ```
  db2schex -u
  ```

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu deinstallieren und Fehlermeldungen zu ignorieren:

  ```
  db2schex -u -k
  ```

Aktivieren der LDAP-Unterstützung nach Abschluss der DB2-Installation

Bevor Sie LDAP verwenden können, müssen Sie nach Abschluss der DB2-Produktinstallation die LDAP-Unterstützung aktivieren.

Gehen Sie wie folgt vor, um die LDAP-Unterstützung zu aktivieren:

1. Führen Sie auf jeder Maschine, die Teil einer Windows-Domäne ist, die folgenden Schritte aus:
   a. Zur Verwendung von Microsoft Active Directory müssen Sie das Verzeichnisschema erweitern, falls Sie dies nicht bereits vor der Installation des DB2-Produkts getan haben. Weitere Informationen hierzu finden Sie im Thema „Erweitern des Verzeichnisschemas für Active Directory".
   b. Installieren Sie die Binärdateien der LDAP-Unterstützung, indem Sie DB2 Setup ausführen und die Unterstützung für die LDAP-Verzeichnisnutzung in der angepassten Installation auswählen. Das Konfigurationsprogramm setzt die DB2-Registrierdatenbankvariable **DB2_ENABLE_LDAP** automatisch auf YES; diese Einstellung ist für die Aktivierung der LDAP-Unterstützung erforderlich.
   c. Optional: Wenn Sie den IBM LDAP-Client anstelle des Microsoft-LDAP-Clients verwenden möchten, müssen Sie die Registrierdatenbankvariable **DB2LDAP_CLIENT_PROVIDER** auf IBM setzen.
2. Führen Sie auf jedem LDAP-Client die folgenden Schritte aus:

a. Geben Sie den TCP/IP-Hostnamen und wahlweise die Portnummer des LDAP-Servers an, indem Sie den folgenden Befehl ausführen: `db2set DB2LDAPHOST=basisdomänenname[:portnummer]`. Dabei ist *basisdomänenname* der TCP/IP-Hostname und *[:portnummer]* die Portnummer. Wenn Sie keine Portnummer angeben, wird die LDAP-Standardportnummer 389 verwendet.

   DB2-Objekte befinden sich unter dem definierten LDAP-Basis-DN (baseDN). Sie können den Basis-DN auf den einzelnen Maschinen konfigurieren, indem Sie den folgenden Befehl ausführen:

   ```
   db2set DB2LDAP_BASEDN=basis-DN
   ```

   Dabei ist *basis-DN* der Name des LDAP-Suffixes, das auf dem LDAP-Server definiert ist.

b. Optional: Wenn Sie LDAP zum Speichern von benutzerspezifischen DB2-Informationen verwenden möchten, geben Sie den DN und das Kennwort des LDAP-Benutzers ein.

3. Wenn Sie das Verzeichnisschema nach der Installation des DB2-Produkts erweitert haben, führen Sie die folgenden Schritte aus:

   a. Registrieren Sie die aktuelle Instanz des DB2-Servers bei LDAP, indem Sie den folgenden Befehl ausführen:

      ```
      db2 register ldap as knotenname protocol tcpip
      ```

   b. Registrieren Sie bestimmte Datenbanken bei LDAP, indem Sie den folgenden Befehl ausführen:

      ```
      db2 catalog ldap database dbname as db_aliasname
      ```

Sie können nun die LDAP-Einträge registrieren.

Registrieren von LDAP-Einträgen

Registrierung von DB2-Servern nach der Installation

Jede DB2-Serverinstanz muss in LDAP registriert werden, um die Protokollkonfigurationsdaten zu veröffentlichen, die von den Clientanwendung zum Herstellen einer Verbindung zu dieser DB2-Serverinstanz verwendet werden.

Beim Registrieren einer Instanz des Datenbankservers müssen Sie einen *Knotennamen* angeben. Der Knotenname wird von Clientanwendungen verwendet, wenn sie eine Verbindung zum Server herstellen. Sie können mit dem Befehl CATALOG LDAP NODE einen anderen Aliasnamen für den LDAP-Knoten katalogisieren.

Anmerkung: Wenn Sie in einer Domänenumgebung unter Windows arbeiten, wird die DB2-Serverinstanz während der Installation im Active Directory automatisch mit den folgenden Informationen registriert:

```
Knotenname: TCP/IP-Hostname
Protokolltyp: TCP/IP
```

Wenn die Länge des TCP/IP-Hostnamens 8 Zeichen überschreitet, wird dieser auf 8 Zeichen abgeschnitten.

Der Befehl REGISTER sieht wie folgt aus:

```
db2 register db2 server in ldap
   as <ldap-knotenname>
   protocol tcpip
```

Die Klausel protocol gibt das Übertragungsprotokoll an, das zum Herstellen einer Verbindung zu diesem Datenbankserver verwendet wird.

Beim Erstellen einer Instanz für DB2 Enterprise Server Edition, die mehrere physische Maschinen umfasst, muss der Befehl REGISTER für jede Maschine einmal aufgerufen werden. Verwenden Sie den Befehl rah, um den Befehl REGISTER an alle Maschinen abzusetzen.

Anmerkung: In ldap_knotenname kann nicht derselbe Knotenname für alle Maschinen verwendet werden, da der Name in LDAP eindeutig sein muss. Sie müssen im Befehl REGISTER den Hostnamen jeder Maschine anstelle von ldap_knotenname verwenden. Beispiel:

```
rah ">DB2 REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

"<>" wird durch den Hostnamen jeder Maschine ersetzt, auf der der Befehl rah ausgeführt wird. Für den seltenen Fall, dass mehrere Instanzen von DB2 Enterprise Server Edition vorhanden sind, kann die Kombination aus Instanz und Hostindex als Knotenname im Befehl rah verwendet werden.

Der Befehl REGISTER kann für einen fernen DB2-Server abgesetzt werden. Dazu müssen Sie den fernen Systemnamen, den Instanznamen und die Protokollkonfigurationsparameter beim Registrieren eines fernen Servers angeben. Der Befehl kann folgendermaßen verwendet werden:

```
db2 register db2 server in ldap
   as <ldap-knotenname>
   protocol tcpip
   hostname <hostname>
   svcename <tcpip-servicename>
   remote <name_des_fernen_computers>
   instance <instanzname>
```

Für den Computernamen gilt folgende Konvention:

- Wenn TCP/IP konfiguriert ist, muss der Systemname mit dem TCP/IP-Hostnamen übereinstimmen.

Beim Betrieb in einer Umgebung mit hoher Verfügbarkeit oder Funktionsübernahme (Failover) und bei Verwendung von TCP/IP als Übertragungsprotokoll muss die Cluster-IP-Adresse verwendet werden. Mit der Cluster-IP-Adresse kann der Client eine Verbindung zum Server auf einer beliebigen Maschine herstellen, ohne für jede Maschine einen separaten TCP/IP-Knoten katalogisieren zu müssen. Die Cluster-IP-Adresse wird mit der Klausel hostname wie folgt angegeben:

```
db2 register db2 server in ldap
   as <ldap-knotenname>
   protocol tcpip
   hostname n.nn.nn.nn
```

Dabei ist n.nn.nn.nn die Cluster-IP-Adresse.

Zur Registrierung des DB2-Servers in LDAP aus einer Clientanwendung heraus rufen Sie die API db2LdapRegister auf.

Katalogisieren eines Knoten-Aliasnamens für ATTACH

Beim Registrieren eines DB2-Servers in LDAP muss ein Knotenname für den Server angegeben werden. Anwendungen verwenden den Knotennamen zum Herstellen einer Verbindung zum Datenbankserver.

Wenn Sie einen anderen Knotennamen benötigen, zum Beispiel wenn der Knotenname in einer Anwendung fest codiert ist, verwenden Sie den Befehl CATALOG LDAP NODE, um die Änderung vorzunehmen. Beispiel:

```
db2 catalog ldap node <ldap_knotenname>
   as <neuer_aliasname>
```

Zum Entfernen eines LDAP-Knotens aus dem Katalog verwenden Sie den Befehl UNCATALOG LDAP NODE. Beispiel:

```
db2 uncatalog ldap node <ldap_knotenname>
```

Registrierung von Datenbanken im LDAP-Verzeichnis

Bei der Erstellung einer Datenbank in einer Instanz wird die Datenbank automatisch in LDAP registriert. Die Registrierung erlaubt Verbindungen ferner Clients zur Datenbank, ohne dass die Datenbank und der Knoten auf der Clientmaschine katalogisiert werden müssen. Wenn ein Client versucht, eine Verbindung zu einer Datenbank herzustellen, wird das LDAP-Verzeichnis durchsucht, wenn die Datenbank nicht im Datenbankverzeichnis auf der lokalen Maschine vorhanden ist.

Wenn der Name bereits im LDAP-Verzeichnis vorhanden ist, wird die Datenbank trotzdem auf der lokalen Maschine erstellt, allerdings mit einer Warnung bezüglich der Namensunverträglichkeit im LDAP-Verzeichnis. Aus diesem Grund können Sie eine Datenbank manuell im LDAP-Verzeichnis katalogisieren. Der Benutzer kann Datenbanken auf einem fernen Server in LDAP mit dem Befehl CATALOG LDAP DATABASE registrieren. Beim Registrieren einer fernen Datenbank geben Sie den Namen des LDAP-Knotens an, der den fernen Datenbankserver darstellt. Sie müssen den fernen Datenbankserver in LDAP mit dem Befehl REGISTER DB2 SERVER IN LDAP registrieren, bevor Sie die Datenbank registrieren. Verwenden Sie den Befehl CATALOG LDAP DATABASE, um eine Datenbank manuell in LDAP zu registrieren:

```
db2 catalog ldap database <dbname>
   at node <knotenname>
   with "Meine LDAP-Datenbank"
```

Zur Registrierung einer Datenbank in LDAP aus einer Clientanwendung heraus rufen Sie die API db2LdapCatalogDatabase auf.

Zurücknehmen registrierter LDAP-Einträge

Zurücknehmen der Registrierung des DB2-Servers

Beim Zurücknehmen der Registrierung einer Instanz in LDAP werden auch alle Knoten- oder Aliasnamenobjekte sowie Datenbankobjekte entfernt, die auf die Instanz verweisen.

Zum Zurücknehmen der Registrierung des DB2-Servers auf einer lokalen oder fernen Maschine muss der LDAP-Knotenname für den Server angegeben werden:

```
db2 deregister db2 server in ldap
   node <knotenname>
```

Zur Rücknahme der Registrierung des DB2-Servers in LDAP aus einer Clientanwendung heraus rufen Sie die API db2LdapDeregister auf.

Wenn die Registrierung des DB2-Servers zurückgenommen wird, werden alle LDAP-Knoteneinträge und LDAP-Datenbankeinträge, die auf dieselbe Instanz im DB2-Server verweisen, ebenfalls aus dem Katalog entfernt.

Zurücknehmen der Registrierung der Datenbank aus dem LDAP-Verzeichnis

Die Registrierung der Datenbank in LDAP wird automatisch zurückgenommen, wenn die Datenbank gelöscht wird oder wenn die Registrierung der Eignerinstanz in LDAP zurückgenommen wird.

Sie können die Registrierung der Datenbank in LDAP mit dem folgenden Befehl manuell zurücknehmen:

```
db2 uncatalog ldap database <dbname>
```

Zur Zurücknahme der Registrierung einer Datenbank in LDAP aus einer Clientanwendung heraus rufen Sie die API db2LdapUncatalogDatabase auf.

Konfigurieren von LDAP-Benutzern

Erstellen eines LDAP-Benutzers

Das DB2-Datenbanksystem unterstützt das Definieren von DB2-Registrierdatenbankvariablen und die CLI-Konfiguration auf Benutzerebene. (Diese Funktion steht auf Linux- und UNIX-Plattformen nicht zur Verfügung.) Durch die Unterstützung auf Benutzerebene stehen in Mehrplatzsystemumgebungen benutzerspezifische Einstellungen zur Verfügung. Ein Beispiel ist der Windows Terminal Server, bei dem jeder angemeldete Benutzer seine Umgebung anpassen kann, ohne dass sich dadurch Konflikte mit der Systemumgebung oder der Umgebung eines anderen Benutzers ergeben.

Beim Einsatz des IBM Tivoli-Verzeichnisses müssen Sie einen LDAP-Benutzer definieren, bevor Informationen auf Benutzerebene unter LDAP gespeichert werden können. Sie können einen LDAP-Benutzer durch Erstellen einer LDIF-Datei erstellen, in der alle Attribute für das Benutzerobjekt enthalten sind. Führen Sie anschließend das Importdienstprogramm LDIF aus, um das Objekt in das LDAP-Verzeichnis zu importieren. Das Dienstprogramm LDIF für IBM Tivoli Directory Server hat den Namen LDIF2DB.

Eine LDIF-Datei mit den Attributen für einen Personenobjekt sieht etwa wie folgt aus:

```
File name: newuser.ldif

dn: cn=Mary Burnnet, ou=DB2 Development, ou=Toronto, o=ibm, c=ca
objectclass: ePerson
cn: Mary Burnnet
sn: Burnnet
uid: mburnnet
userPassword: password
telephonenumber: 1-416-123-4567
facsimiletelephonenumber: 1-416-123-4568
title: Software Developer
```

Das folgende Beispiel zeigt den LDIF-Befehl zum Importieren einer LDIF-Datei mit dem IBM LDIF-Importdienstprogramm:

```
LDIF2DB -i newuser.ldif
```

Anmerkung:

1. Der Befehl LDIF2DB muss auf der LDAP-Servermaschine ausgeführt werden.

2. Sie müssen die erforderlichen Zugriffsberechtigungen (ACL) für das LDAP-Benutzerobjekt erteilen, damit der LDAP-Benutzer sein eigenes Objekt hinzufügen, löschen, lesen und schreiben kann. Verwenden Sie das Tool LDAP Directory Server Web Administration, um für das Benutzerobjekt die ACL-Zugriffsberechtigungen zu erteilen.

Konfigurieren des LDAP-Benutzers für DB2-Anwendungen

Bei Verwendung des Microsoft LDAP-Clients stimmt der LDAP-Benutzer mit dem Benutzerkonto des Betriebssystems überein. Wenn Sie den IBM LDAP-Client jedoch verwenden, bevor Sie den DB2-Datenbankmanager verwenden, müssen Sie den definierten LDAP-Benutzernamen (DN) und das Kennwort für den zurzeit angemeldeten Benutzer konfigurieren.

Verwenden Sie zum Konfigurieren des definierten Namens (DN) und des Kennworts des LDAP-Benutzers das Dienstprogramm db2ldcfg:

```
db2ldcfg -u <benutzer-DN> -w <kennwort> --> DN und Kennwort des Benutzers festlegen
         -r                          --> DN und Kennwort des Benutzers löschen
```

Beispiel:

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 Development,ou=Toronto,o=ibm,c=ca"
         -w kennwort
```

Definieren von DB2-Registrierdatenbankvariablen auf Benutzerebene in der LDAP-Umgebung

In der LDAP-Umgebung können die DB2-Variablen der Profilregistrierdatenbank auf Benutzerebene definiert werden. Dadurch können Benutzer ihre eigene DB2-Umgebung anpassen.

Um die DB2-Variablen der Profilregistrierdatenbank auf Benutzerebene zu definieren, verwenden Sie die Option `-ul`:

```
db2set -ul <variable>=<wert>
```

Anmerkung: Diese Funktion wird unter den Betriebssystemen AIX und Solaris nicht unterstützt.

DB2 arbeitet mit einem Caching-Verfahren. Die DB2-Variablen der Profilregistrierdatenbank auf Benutzerebene werden auf der lokalen Maschine zwischengespeichert. Wenn der Parameter `-ul` angegeben wird, liest DB2 die DB2-Registrierdatenbankvariablen immer aus dem Cache. Der Cache wird in folgenden Fällen aktualisiert:

- Sie aktualisieren eine DB2-Registrierdatenbankvariable auf Benutzerebene oder setzen sie zurück.
- Der Befehl zum Aktualisieren der LDAP-Profilvariablen auf Benutzerebene lautet wie folgt:

  ```
  db2set -ur
  ```

Inaktivieren der LDAP-Unterstützung

Verwenden Sie die folgende Prozedur, um die LDAP-Unterstützung zu inaktivieren:

1. Nehmen Sie für jede Instanz des DB2-Servers die Registrierung des DB2-Servers in LDAP zurück:

   ```
   db2 deregister db2 server in ldap node <knotenname>
   ```

2. Setzen Sie die Variable DB2_ENABLE_LDAP der DB2-Profilregistrierdatenbank auf den Wert ″NO″.

Aktualisieren der Protokollinformationen für den DB2-Server

Die DB2-Serverinformationen in LDAP müssen immer aktuell sein. Änderungen an den Protokollkonfigurationsparametern oder der Server-Netzwerkadresse beispielsweise machen eine LDAP-Aktualisierung erforderlich.

Aktualisieren Sie den DB2-Server in LDAP auf der lokalen Maschine mit dem folgenden Befehl:

```
db2 update ldap ...
```

Beispiele für Protokollkonfigurationsparameter, die aktualisiert werden können, sind Parameter für TCP/IP-Hostname und Servicename oder Portnummer.

Zum Aktualisieren der Protokollkonfigurationsparameter eines fernen DB2-Servers verwenden Sie den Befehl UPDATE LDAP mit der Klausel node:

```
db2 update ldap
   node <knotenname>
   hostname <hostname>
   svcename <tcpip-servicename>
```

Weiterleiten von LDAP-Clients an andere Server

Die Möglichkeit zur Weiterleitung von Clients bei Systemausfällen steht auch bei Einsatz von LDAP zur Verfügung.

Die Registrierdatenbankvariable DB2_ENABLE_LDAP muss auf den Wert „Yes” sein.

In einer LDAP-Umgebung werden sämtliche Datenbank- und Knotenverzeichnisdaten auf einem LDAP-Server verwaltet. Der Client ruft Daten aus dem LDAP-Verzeichnis ab. Diese Daten werden in den lokalen Datenbank- und Knotenverzeichnissen aktualisiert, wenn die Registrierdatenbankvariable DB2LDAPCACHE auf den Wert „Yes” gesetzt ist.

Verwenden Sie den Befehl UPDATE ALTERNATE SERVER FOR LDAP DATABASE, um den alternativen Server für eine Datenbank zu definieren, die die DB2-Datenbank in LDAP darstellt. Alternativ können Sie die API ’db2LdapUpdateAlternateServerForDB’ von einer Clientanwendung aus aufrufen, um den alternativen Server für die Datenbank in LDAP zu aktualisieren.

Wenn die Daten über diesen alternativen Server eingerichtet sind, werden sie an den Client zurückgegeben, wenn eine Verbindung hergestellt wird.

Anmerkung: Es wird dringend empfohlen, die im LDAP-Server gespeicherten Daten über den alternativen Server mit den Daten über den alternativen Server, die in der Datenbankserverinstanz gespeichert sind, synchron zu halten. Das Ausführen des Befehls UPDATE ALTERNATE SERVER FOR DATABASE (beachten Sie, dass es nicht ″FOR LDAP DATABASE″ heißt) in der Datenbankserverinstanz hilft bei der Sicherstellung der Synchronisierung zwischen der Datenbankserverinstanz und dem LDAP-Server.

Wenn Sie den Befehl UPDATE ALTERNATE SERVER FOR DATABASE in der Serverinstanz eingeben oder wenn die LDAP-Unterstützung auf dem Server akti-

viert (DB2_ENABLE_LDAP=Yes) ist und die LDAP-Benutzer-ID und das Kennwort im Cache gespeichert werden (der Befehl db2ldcfg wurde zuvor ausgeführt), dann wird der alternative Server für die Datenbank automatisch bzw. implizit auf dem LDAP-Server aktualisiert. Dies funktioniert genau so, wie bei einer expliziten Eingabe des Befehls UPDATE ALTERNATE SERVER FOR LDAP DATABASE.

Wenn der Befehl UPDATE ALTERNATE SERVER FOR LDAP DATABASE von einer anderen Instanz als der Datenbankserverinstanz aus abgesetzt wird, stellen Sie mithilfe des Befehls UPDATE ALTERNATE SERVER FOR DATABASE sicher, dass die Daten über den alternativen Server in der Datenbankserverinstanz identisch konfiguriert werden. Nachdem der Client zu Anfang eine Verbindung zur Datenbankserverinstanz hergestellt hat, erhalten die Daten über den alternativen Server, die aus der Datenbankserverinstanz zurückgegeben werden, Vorrang vor den Daten, die im LDAP-Server konfiguriert sind. Wenn die Datenbankserverinstanz keine konfigurierten Daten über einen alternativen Server hat, wird die Clientweiterleitung nach der einleitenden Verbindung als inaktiviert betrachtet.

Herstellen einer ATTACH-Verbindung zu einem fernen Server in der LDAP-Umgebung

In der LDAP-Umgebung können Sie eine Verbindung zu einem fernen Datenbankserver mit dem LDAP-Knotennamen im Befehl ATTACH herstellen: `db2 attach to <ldap-knotenname>`.

Wenn eine Clientanwendung zum ersten Mal eine Verbindung zu einem Knoten oder einer Datenbank herstellt, durchsucht der Datenbankmanager das LDAP-Verzeichnis nach dem Zielknoteneintrag, weil der Knoten im lokalen Knotenverzeichnis nicht definiert ist. Wenn der Eintrag im LDAP-Verzeichnis gefunden wird, werden die Protokollinformationen zum fernen Server abgerufen. Wenn Sie eine Verbindung zur Datenbank herstellen und der Eintrag im LDAP-Verzeichnis gefunden wird, werden die Datenbankinformationen ebenfalls abgerufen. Mithilfe dieser Informationen katalogisiert der Datenbankmanager automatisch einen Datenbankeintrag und einen Knoteneintrag auf der lokalen Maschine. Wenn die Clientanwendung das nächste Mal eine Verbindung zum selben Knoten oder zur selben Datenbank herstellt, werden die Informationen im lokalen Datenbankverzeichnis verwendet, sodass das LDAP-Verzeichnis nicht durchsucht werden muss.

Es gibt einen Cachingmechanismus, der dafür sorgt, dass der Client den LDAP-Server nur einmal durchsucht. Wenn die Informationen abgerufen sind, werden sie auf der lokalen Maschine nach Maßgabe der Werte des Konfigurationsparameters *dir_cache* des Datenbankmanagers und der Registrierdatenbankvariablen DB2LDAPCACHE gespeichert bzw. im Cache abgelegt.

- Sind DB2LDAPCACHE=NO und dir_cache=NO definiert, werden die Informationen immer aus LDAP gelesen.
- Bei DB2LDAPCACHE=NO und dir_cache=YES werden die Informationen einmal aus LDAP gelesen und in den DB2(R)-Cache eingefügt.
- Ist DB2LDAPCACHE=YES oder diese Registrierdatenbankvariable überhaupt nicht definiert, werden die Informationen einmal aus dem LDAP-Server gelesen und in den Cache für lokale Datenbank-, Knoten- und DCS-Verzeichnisse gestellt.

Anmerkung: Das Caching von LDAP-Informationen ist für CLI-Variablen oder Variablen der DB2-Profilregistrierdatenbank auf Benutzerebene nicht gültig.

Aktualisieren der LDAP-Einträge in lokalen Datenbank- und Knotenverzeichnissen

Für das DB2-Datenbanksystem steht ein Caching-Mechanismus zur Verfügung, durch den die Häufigkeit reduziert werden kann, mit der ein Client den LDAP-Server durchsucht.

Wenn die Informationen abgerufen sind, werden sie auf der lokalen Maschine nach Maßgabe der Werte des Konfigurationsparameters 'dir_cache' des Datenbankmanagers und der Registrierdatenbankvariablen DB2LDAPCACHE gespeichert bzw. im Cache abgelegt.

- Sind DB2LDAPCACHE=NO und dir_cache=NO definiert, werden die Informationen immer aus LDAP gelesen.
- Bei DB2LDAPCACHE=NO und dir_cache=YES werden die Informationen einmal aus LDAP gelesen und in den DB2-Cache eingefügt.
- Ist DB2LDAPCACHE=YES oder diese Registrierdatenbankvariable überhaupt nicht definiert, werden die Informationen einmal aus dem LDAP-Server gelesen und in den Cache für lokale Datenbank-, Knoten- und DCS-Verzeichnisse gestellt.

Anmerkung: Das Caching von LDAP-Informationen ist für CLI-Variablen oder Variablen der DB2-Profilregistrierdatenbank auf Benutzerebene nicht gültig. Da LDAP-Informationen Änderungen unterliegen, kann es erforderlich sein, die LDAP-Einträge, die in den lokalen Datenbank- und Knotenverzeichnissen zwischengespeichert sind, zu aktualisieren. Hierfür gibt es eine Reihe möglicher Vorgehensweisen.

Verwenden Sie den folgenden Befehl, um alle lokalen Datenbank- und Knoteneinträge zu aktualisieren, die von LDAP abgerufen wurden:

```
db2 refresh ldap immediate
```

Entsprechend kann der folgende Befehl verwendet werden, um vorhandene lokale Datenbank- und Knoteneinträge zu aktualisieren und um neue Einträge aus LDAP hinzuzufügen:

```
db2 refresh ldap immediate all
```

Durch die Angabe der Option IMMEDIATE ALL werden alle im LDAP-Server enthaltenen Knoten- und Datenbankeinträge zu den lokalen Verzeichnissen hinzugefügt.

Alternativ dazu können Sie den folgenden Befehl verwenden, um das Aktualisieren der Datenbankeinträge, die auf LDAP-Ressourcen verweisen, durch DB2 beim nächsten Herstellen der Datenbankverbindung bzw. bei der nächsten Instanzverbindung zu erzwingen:

```
db2 refresh ldap database directory
```

Entsprechend können Sie den folgenden Befehl verwenden, um das Aktualisieren der Knoteneinträge, die auf LDAP-Ressourcen verweisen, durch den DB2-Datenbankmanager beim nächsten Herstellen der Datenbankverbindung bzw. bei der nächsten Instanzverbindung zu erzwingen:

```
db2 refresh ldap node directory
```

Im Rahmen der Aktualisierung werden alle LDAP-Einträge entfernt, die in den lokalen Datenbank- und Knotenverzeichnissen gespeichert sind. Beim nächsten Zugriff der Anwendung auf die Datenbank oder den Knoten liest sie die Informationen direkt aus LDAP und generiert einen neuen Eintrag im lokalen Datenbank- oder Knotenverzeichnis.

Sie können wie folgt vorgehen, um eine rechtzeitige Aktualisierung sicherzustellen:

- Planen Sie eine in regelmäßigen Abständen durchgeführte Aktualisierung.
- Führen Sie den Befehl REFRESH beim Systemstart aus.
- Verwenden Sie ein verfügbares Verwaltungspaket, um den Befehl REFRESH auf allen Clientmaschinen aufzurufen.
- Definieren Sie DB2LDAPCACHE="NO", um das Caching von LDAP-Informationen in den Datenbank-, Knoten- und DCS-Verzeichnissen zu verhindern.

Durchsuchen von LDAP-Servern

Das DB2-Datenbanksystem durchsucht den aktuellen LDAP-Server. In einer Umgebung, in der mehrere LDAP-Server vorhanden sind, können Sie den Suchbereich jedoch definieren.

Wenn beispielsweise die Informationen im aktuellen LDAP-Server nicht gefunden werden, können Sie eine automatische Durchsuchung aller anderen LDAP-Server angeben oder, alternativ, den Suchbereich allein auf den aktuellen LDAP-Server oder auf den lokalen DB2-Datenbankkatalog einschränken.

Wenn Sie den Suchbereich festlegen, wird dadurch der Standardsuchbereich für das gesamte Unternehmen festgelegt. Der Suchbereich wird durch die Variable DB2LDAP_SEARCH_SCOPE der DB2-Profildatenbank gesteuert. Zum Festlegen des Suchbereichswerts verwenden Sie die Option -gl (d. h. global in LDAP) im Befehl db2set:

```
db2set -gl db2ldap_search_scope = <wert>
```

Mögliche Werte sind: local, domain oder global. Wenn diese Registrierdatenbankvariable nicht definiert ist, gilt der Standardwert 'domain', der den Suchbereich auf das Verzeichnis auf dem aktuellen LDAP-Server einschränkt.

Sie können zum Beispiel den Suchbereich zunächst auf den Wert „global" setzen, nachdem eine neue Datenbank erstellt wurde. Dadurch kann jeder zur Verwendung von LDAP konfigurierte DB2-Client alle LDAP-Server durchsuchen, um die Datenbank zu finden. Sobald der Eintrag nach der ersten Verbindung (CONNECT oder ATTACH) für jeden Client erfasst wurde (sofern Sie das Caching aktiviert haben), kann der Suchbereich in „local" geändert werden. Nachdem der Suchbereich in „local" geändert wurde, durchsuchen die Clients keine LDAP-Server mehr.

Anmerkung: Die Variablen DB2LDAP_KEEP_CONNECTION und DB2LDAP_SEARCH_SCOPE der DB2-Profildatenbank sind die einzigen Registrierdatenbankvariablen, die auf der globalen Ebene in LDAP definiert werden können.

Teil 2. Datenbanken

Kapitel 6. Datenbanken

Bei einer DB2-Datenbank handelt es sich um eine *relationale Datenbank*. In einer solchen *Datenbank* werden alle Daten in Tabellen gespeichert, die in einer bestimmten Relation zueinander stehen. Diese Relationen zwischen den Tabellen sind so konzipiert, dass die Daten gemeinsam benutzt werden können und dass die doppelte Datenhaltung auf ein Minimum beschränkt werden kann.

Eine *relationale Datenbank* wird als eine Gruppe von Tabellen betrachtet. Die in dieser Datenbank gespeicherten Daten werden auf der Basis des relationalen Datenmodells bearbeitet. Sie enthält eine Gruppe von Objekten, die zum Speichern, Verwalten und für den Zugriff auf die gespeicherten Daten verwendet werden. Zu diesen Objekten gehören z. B. Tabellen, Sichten, Indizes, Funktionen, Trigger sowie Pakete. Objekte werden entweder vom System (systemdefinierte Objekte) oder von einem Benutzer (benutzerdefinierte Objekte) definiert.

Eine *verteilte relationale Datenbank* besteht aus einer Gruppe von Tabellen und anderen Objekten, die über mehrere, jedoch miteinander verbundene Computersysteme verteilt sind. Jedes Computersystem verfügt hierbei über einen Manager für relationale Datenbanken, mit dem die Tabellen in der jeweiligen Umgebung verwaltet werden. Diese Datenbankmanager kommunizieren und kooperieren miteinander auf eine Weise, die es einem bestimmten Datenbankmanager erlaubt, SQL-Anweisungen auf einem der anderen Computersysteme auszuführen.

Eine *partitionierte relationale Datenbank* ist eine relationale Datenbank, deren Daten über mehrere Datenbankpartitionen verteilt sind und partitionsübergreifend verwaltet werden. Diese Verteilung der Daten über mehrere Datenbankpartitionen ist für den Benutzer bei den meisten SQL-Anweisungen transparent. Bei bestimmten DDL-Anweisungen (DDL = Data Definition Language, Datendefinitionssprache) werden die Datenbankpartitionsinformationen jedoch berücksichtigt (z. B. CREATE DATABASE PARTITION GROUP). Bei DDL handelt es sich um eine Untergruppe von SQL-Anweisungen, die zur Beschreibung der Datenrelationen in einer Datenbank verwendet werden.

Bei einer *föderierten Datenbank* handelt es sich um eine relationale Datenbank, deren Datenbestand in mehreren Datenquellen (z. B. in separaten relationalen Datenbanken) gespeichert ist. Die Daten erscheinen für den Benutzer hierbei so, als wären sie alle in einer einzigen, großen Datenbank enthalten. Sie können mithilfe traditioneller SQL-Abfragen abgerufen werden. Änderungen der Daten können explizit für die gewünschte Datenquelle ausgeführt werden.

Entwerfen von Datenbanken

Beim Entwerfen einer Datenbank modellieren Sie ein reales Business-System, das eine Reihe von Entitäten und deren Merkmale (oder *Attribute*) sowie die Regeln bzw. Beziehungen umfasst, die zwischen diesen Entitäten gelten.

Der erste Schritt besteht in der Beschreibung des Systems, das Sie darstellen möchten. Wenn Sie z. B. eine Datenbank für ein Publikationssystem erstellen möchten, dann sollte das System verschiedene Entitätstypen wie beispielsweise Bücher, Autoren, Herausgeber und Verleger umfassen. Für jede dieser Entitäten stehen bestimmte Informationen (oder Attribute) zur Verfügung, die Sie aufzeichnen müssen:

- *Bücher:* Titel, ISBN-Nummer, Erscheinungsdatum und -ort, Verleger, ...
- *Autoren:* Name, Anschrift, Telefon- und Faxnummer, E-Mail-Adresse, ...
- *Herausgeber* Name, Anschrift, Telefon- und Faxnummer, E-Mail-Adresse, ...
- *Verleger:* Name, Anschrift, Telefon- und Faxnummer, E-Mail-Adresse, ...

Ihre Datenbank muss nicht nur diese Entitätstypen sowie die zugehörigen Attribute darstellen, sondern Sie müssen auch in der Lage sein, diese Entitäten in Beziehung zueinander zu setzen. Sie müssen z. B. die Relation darstellen können, die zwischen Büchern und ihren Autoren, zwischen Büchern/Autoren und Verlegern sowie zwischen Büchern/Autoren und Herausgebern besteht.

Zwischen den Entitäten einer Datenbank können drei verschiedene Typen von Beziehungen bestehen:

Eins-zu-eins-Beziehung
Bei diesem Beziehungstyp ist jeder Instanz einer Entität genau eine Instanz der anderen Entität zugeordnet. Momentan besteht in dem hier beschriebenen Szenario keine Eins-zu-eins-Beziehung.

Eins-zu-viele-Beziehung
Bei diesem Beziehungstyp ist jeder Instanz einer Entität mindestens eine Instanz einer anderen Entität zugeordnet. Ein Autor kann z. B. mehrere Bücher geschrieben haben, für die einzelnen Bücher gibt es jedoch nur einen Autor. Dies ist der Beziehungstyp, der in relationalen Datenbanken am häufigsten modelliert wird.

Viele-zu-viele-Beziehungen
Bei diesem Beziehungstyp besteht eine Beziehung zwischen mehreren Instanzen einer bestimmten Entität zu mindestens einer Instanz einer anderen Entität. So kann es z. B. der Fall sein, dass mehrere Koautoren mehrere Bücher geschrieben haben.

Da Datenbanken aus Tabellen aufgebaut sind, müssen Sie eine Gruppe von Tabellen erstellen, in denen die Daten optimal gespeichert werden können. Hierbei enthält jede Zelle der Tabelle eine einzige Sicht. Diese Aufgabe kann auf unterschiedliche Arten ausgeführt werden. Als Datenbankentwickler sind Sie dafür verantwortlich, die optimale Tabellengruppe für Ihre Daten zu entwerfen.

Sie können zum Beispiel eine einzelne Tabelle erstellen, die mehrere Zeilen und Spalten umfasst, in denen alle verfügbaren Informationen gespeichert werden können. Bei diesem Verfahren kommt es allerdings dazu, dass bestimmte Informationen wiederholt aufgeführt sind. Darüber hinaus ist die Dateneingabe und die Datenpflege bei diesem Verfahren zeitaufwendig und fehleranfällig. Im Gegensatz zum Einzeltabellenentwurf ermöglicht Ihnen eine *relationale Datenbank* die Verwendung mehrerer einfacher Tabellen. Hierdurch kann die Datenredundanz reduziert und es können Probleme vermieden werden, die sich durch sehr große und schwierig zu verwaltende Tabellen ergeben. In einer relationalen Datenbank müssen Tabellen die Informationen zu einem einzigen Entitätstyp enthalten.

Darüber hinaus muss in einer relationalen Datenbank die Datenintegrität gewährleistet werden, wenn mehrere Benutzer auf die Daten zugreifen oder sie ändern. Immer wenn Daten gemeinsam verwendet werden, muss die Richtigkeit der Werte innerhalb von Datenbanktabellen sichergestellt werden.

Mögliche Operationen:

- Mithilfe von Isolationsstufen können Sie festlegen, wie Daten gesperrt oder von anderen Prozessen isoliert werden, während auf die Daten zugegriffen wird.
- Sie können Daten schützen und Beziehungen zwischen Daten herstellen, indem Sie Integritätsbedingungen zur Umsetzung von Geschäftsregeln definieren.
- Sie können Trigger erstellen, die eine komplexe, tabellenübergreifende Datenprüfung durchführen können.
- Sie können eine Recoverystrategie implementieren, um Daten so zu schützen, dass sie in einem konsistenten Zustand wiederhergestellt werden können.

Der Datenbankentwurf ist weitaus komplexer als dies hier beschrieben werden kann, und es gibt viele Faktoren, die berücksichtigt werden müssen. Hierzu gehören beispielsweise die Speicherplatzanforderungen, die zu verwendenden Schlüssel, Indizes und Integritätsbedingungen sowie Sicherheitsaspekte und Fragen zur Zugriffsberechtigung usw. Einige dieser Informationen finden Sie in der DB2-Informationszentrale und in den zahlreichen DB2-Dokumentationen, die zu diesem Thema zur Verfügung stehen.

Verzeichnisse und Dateien einer Datenbank

Beim Erstellen einer Datenbank werden zugehörige Informationen einschließlich Standardinformationen in einer Verzeichnishierarchie gespeichert.

Die hierarchische Verzeichnisstruktur wird an der Speicherposition erstellt, die durch die von Ihnen im Befehl CREATE DATABASE angegebenen Informationen festgelegt wird. Wenn Sie beim Erstellen der Datenbank die Position des Verzeichnispfads oder Laufwerks nicht angeben, wird die Standardposition verwendet. Es empfiehlt sich, explizit anzugeben, wo die Datenbank erstellt werden soll.

In dem Verzeichnis, das Sie im Befehl CREATE DATABASE als Datenbankpfad angeben, wird ein Unterverzeichnis erstellt, das den Namen der Instanz verwendet. Dieses Unterverzeichnis stellt sicher, dass Datenbanken, die in verschiedenen Instanzen in demselben Verzeichnis erstellt werden, nicht denselben Pfad verwenden. Unterhalb des Unterverzeichnisses mit dem Instanznamen wird ein Unterverzeichnis mit dem Namen NODE0000 erstellt. Dieses Unterverzeichnis unterscheidet Datenbankpartitionen in einer logisch partitionierten Datenbankumgebung. Unterhalb des Verzeichnisses mit dem NODE-Namen wird ein Unterverzeichnis mit dem Namen SQL00001 erstellt. Der Name dieses Unterverzeichnisses verwendet das Datenbanktoken und stellt die Datenbank dar, die erstellt wird. Das Verzeichnis SQL00001 enthält Objekte, die der ersten erstellten Datenbank zugeordnet sind. Nachfolgend erstellte Datenbanken erhalten höhere Nummern: SQL00002 usw. Diese Unterverzeichnisse unterscheiden Datenbanken, die in dieser Instanz in dem Verzeichnis erstellt werden, das Sie im Befehl CREATE DATABASE angegeben haben.

Die Verzeichnisstruktur wird wie folgt dargestellt: *ihr_datenbankpfad*/*ihre_instanz*/NODE0000/SQL00001/.

Das Datenbankverzeichnis enthält die folgenden Dateien, die durch den Befehl CREATE DATABASE erstellt werden.

- Die Dateien SQLBP.1 und SQLBP.2 enthalten Pufferpoolinformationen. Diese Dateien sind miteinander identisch und werden als Backup erstellt.

- Die Dateien SQLSPCS.1 und SQLSPCS.2 enthalten Tabellenbereichsinformationen. Diese Dateien sind miteinander identisch und werden als Backup erstellt.
- Die Dateien SQLSGF.1 und SQLSGF.2 enthalten Speicherpfadinformationen, die dem dynamischen Speicher der Datenbank zugeordnet sind. Diese Dateien sind miteinander identisch und werden als Backup erstellt.
- Die Datei SQLDBCONF enthält Datenbankkonfigurationsdaten. Editieren Sie diese Datei nicht.

  Anmerkung: Die Datei SQLDBCON wurde in Vorgängerreleases verwendet und enthält ähnliche Informationen, die verwendet werden können, wenn SQLDBCONF beschädigt wurde.
  Verwenden Sie zum Ändern von Konfigurationsparametern die Befehle UPDATE DATABASE CONFIGURATION und RESET DATABASE CONFIGURATION.
- Die Protokolldatei DB2RHIST.ASC und ihre Backup-Kopie DB2RHIST.BAK enthalten Protokollinformationen über Backups, Restores, Ladeoperationen für Tabellen, Reorganisationen von Tabellen, Änderungen an Tabellenbereichen und andere Änderungen an einer Datenbank.

  Die Datei DB2TSCHG.HIS enthält ein Protokoll über Tabellenbereichsänderungen auf Protokolldateiebene. Für jede Protokolldatei enthält die Datei DB2TSCHG.HIS Informationen, die bei der Ermittlung der von der Protokolldatei betroffenen Tabellenbereiche helfen. Die Funktion zur Recovery von Tabellenbereichen verwendet Informationen aus dieser Datei, um festzustellen, welche Protokolldateien bei einer Tabellenbereichsrecovery zu verarbeiten sind. Die können den Inhalt beider Protokolldateien in einem Texteditor untersuchen.
- Die Protokollsteuerdateien, d. h. SQLOGCTL.LFH.1, die zugehörige Spiegelkopie SQLOGCTL.LFH.2 und SQLOGMIR.LFH, enthalten Informationen zu den aktiven Protokollen.

  Bei der Recoveryverarbeitung wird anhand von Informationen aus diesen Dateien festgestellt, an welcher Stelle in den Protokollen die Recovery beginnen soll. Das Unterverzeichnis SQLOGDIR enthält die eigentlichen Protokolldateien.

  Anmerkung: Sie sollten sicherstellen, dass das Unterverzeichnis für die Protokolle anderen Platten zugeordnet ist als denen, die für Ihre Daten verwendet werden. Ein Plattenproblem kann in diesem Fall auf Ihre Daten bzw. Ihre Protokolle beschränkt werden, sodass nicht beide gleichzeitig davon betroffen sind. Dies kann zudem einen deutlichen Vorteil für die Leistung mit sich bringen, da die Protokolldateien und die Datenbankcontainer nicht um die Bewegung derselben Plattenschreib-/leseköpfe konkurrieren. Ändern Sie die Position des Protokollunterverzeichnisses mithilfe des Konfigurationsparameters **newlogpath** der Datenbank.
- Die Datei SQLINSLK hilft bei der Sicherstellung, dass eine Datenbank nur von einer Instanz des Datenbankmanagers verwendet wird.

Bei der Erstellung einer Datenbank wird gleichzeitig auch ein detaillierter Ereignismonitor für Deadlocks erstellt. Die Dateien des detaillierten Ereignismonitors für Deadlocks werden im Datenbankverzeichnis auf dem Katalogknoten gespeichert. Wenn der Ereignismonitor die für ihn festgelegte maximale Anzahl von Dateien zur Ausgabe erreicht, wird er inaktiviert und eine Nachricht wird in das Benachrichtigungsprotokoll geschrieben. Dadurch wird verhindert, dass der Ereignismonitor zu viel Plattenspeicherplatz belegt. Durch das Entfernen der Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

Weitere Informationen für SMS-Datenbankverzeichnisse in Datenbanken mit nicht dynamischem Speicher

Bei Datenbanken mit nicht dynamischem Speicher enthalten die SQLT*-Unterverzeichnisse die SMS-Standardtabellenbereiche (SMS = System Managed Space, vom System verwalteter Speicher), die für eine betriebsfähige Datenbank erforderlich sind. Die folgenden drei Standardtabellenbereiche werden erstellt:

- Das Unterverzeichnis SQLT0000.0 enthält den Katalogtabellenbereich mit den Systemkatalogtabellen.
- Das Unterverzeichnis SQLT0001.0 enthält den Standardtabellenbereich für temporäre Tabellen.
- Das Unterverzeichnis SQLT0002.0 enthält den Standardtabellenbereich für Benutzerdaten.

In jedem Unterverzeichnis bzw. jedem Container wird eine Datei mit dem Namen SQLTAG.NAM erstellt. Diese Datei markiert das betreffende Unterverzeichnis als in Gebrauch, sodass bei späteren Erstellungsoperationen für Tabellenbereiche nicht versucht wird, diese Unterverzeichnisse zu verwenden.

Darüber hinaus werden in einer Datei mit dem Namen SQL*.DAT Informationen zu jeder Tabelle gespeichert, die das Unterverzeichnis bzw. der Container enthält. Der Stern (*) wird durch eine eindeutige Folge von Ziffern ersetzt, die jede Tabelle identifiziert. Für jede Datei SQL*.DAT können je nach Tabellentyp, Reorganisationsstatus der Tabelle bzw. nach Vorhandensein von Indizes, LOB- oder LONG-Felder für die Tabelle eine oder mehrere der folgenden Dateien vorhanden sein:

- SQL*.BKM (enthält bei einer MDC-Tabelle Blockzuordnungsinformationen)
- SQL*.LF (enthält Daten vom Typ LONG VARCHAR oder LONG VARGRAPHIC)
- SQL*.LB (enthält Daten der Typen BLOB, CLOB oder DBCLOB)
- SQL*.XDA (enthält XML-Daten)
- SQL*.LBA (enthält Informationen zu zugeordnetem und freiem Speicherbereich für SQL*.LB-Dateien)
- SQL*.INX (enthält Indextabellendaten)
- SQL*.IN1 (enthält Indextabellendaten)
- SQL*.DTR (enthält temporäre Daten für eine Reorganisation einer SQL*.DAT-Datei)
- SQL*.LFR (enthält temporäre Daten für eine Reorganisation einer SQL*.LF-Datei)
- SQL*.RLB (enthält temporäre Daten für eine Reorganisation einer SQL*.LB-Datei)
- SQL*.RBA (enthält temporäre Daten für eine Reorganisation einer SQL*.LBA-Datei)

Datenbankkonfigurationsdatei

Für jede Datenbank wird eine *Datenbankkonfigurationsdatei* erstellt. Diese Datei hat vor Version 8.2 den Namen SQLDBCON und ab Version 8.2 den Namen SQLDBCONF. Diese Datei wird automatisch für Sie erstellt.

Diese Datei enthält Werte für verschiedene *Konfigurationsparameter*, die sich auf die Verwendung der Datenbank auswirken, wie die folgenden:

- Parameter, die bei der Erstellung der Datenbank angegeben oder verwendet werden (z. B. Codepage der Datenbank, Sortierfolge, DB2-Datenbank-Release-Level)
- Parameter, die den aktuellen Status der Datenbank anzeigen (z. B. Anzeiger für Backup anstehend, Datenbankkonsistenz, aktualisierende Recovery anstehend)

- Parameter, die die Menge an Systemressourcen definieren, die für den Betrieb der Datenbank verwendet werden sollen (z. B. Pufferpoolgröße, Datenbankprotokollierung, Sortierspeichergröße)

Anmerkung: Wenn Sie db2system oder SQLDBCONF (vor Version 8.2) bzw. SQLDBCONF (ab Version 8.2) mit anderen als den vom DB2-Datenbankmanager bereitgestellten Methoden editieren, können Sie die Datenbank unbrauchbar machen. Aus diesem Grund sollten Sie diese Dateien nicht mit anderen als den dokumentierten und vom Datenbankmanager unterstützten Methoden ändern.

Hinweis zur Leistung: Viele Konfigurationsparameter verfügen zwar über Standardwerte, müssen aber möglicherweise aktualisiert werden, um eine optimale Leistung für Ihre Datenbank zu erzielen. Standardmäßig wird der Konfigurationsadvisor während der Ausführung des Befehls CREATE DATABASE aufgerufen, sodass die Anfangswerte für einige der Parameter für Ihre Umgebung bereits konfiguriert sind.

Bei Mehrpartitionsdatenbanken: Wenn Sie eine Datenbank haben, die auf mehr als eine Datenbankpartition verteilt ist, sollte die Konfigurationsdatei in allen Datenbankpartitionen die gleiche sein. Diese Konsistenz ist erforderlich, da der Abfragecompiler verteilte SQL-Anweisungen anhand der Informationen in der Konfigurationsdatei des lokalen Knotens kompiliert und einen Zugriffsplan erstellt, der die Anforderungen der jeweiligen SQL-Anweisung erfüllt. Wenn sich verschiedene Konfigurationsdateien in den Datenbankpartitionen befinden, kann dies je nachdem, in welcher Datenbankpartition die Anweisung vorbereitet wird, zu verschiedenen Zugriffsplänen führen.

Knotenverzeichnis

Der Datenbankmanager erstellt das *Knotenverzeichnis*, wenn die erste Datenbankpartition katalogisiert wird.

Zum Katalogisieren einer Datenbankpartition wird der Befehl CATALOG NODE verwendet. Zum Auflisten des Inhalts des lokalen Knotenverzeichnisses wird der Befehl LIST NODE DIRECTORY verwendet.

Das Knotenverzeichnis wird auf jedem Datenbankclient erstellt und verwaltet. Das Verzeichnis enthält einen Eintrag für jede ferne Workstation, die eine oder mehrere Datenbanken enthält, auf die der Client zugreifen kann. Der DB2-Client verwendet die Informationen über den Kommunikationsendpunkt im Knotenverzeichnis jedes Mal, wenn eine Datenbankverbindung oder eine Instanzverbindung (mit ATTACH) angefordert wird.

Die Einträge im Verzeichnis enthalten außerdem Informationen zum Typ des Übertragungsprotokolls, das für die Kommunikation zwischen dem Client und der fernen Datenbankpartition verwendet werden soll. Durch das Katalogisieren einer lokalen Datenbankpartition wird ein Aliasname für eine Instanz erstellt, die sich auf demselben Computer befindet.

Lokales Datenbankverzeichnis

In jedem Pfad (bzw. „Laufwerk" unter Windows-Betriebssystemen), in dem eine Datenbank definiert wurde, ist eine Datei für das *lokale Datenbankverzeichnis* vorhanden. Dieses Verzeichnis enthält einen Eintrag für jede Datenbank, auf die von dieser Position aus zugegriffen werden kann.

Ein Eintrag enthält folgende Daten:

- Den Datenbanknamen, der mit dem Befehl CREATE DATABASE angegeben wurde
- Den Aliasnamen der Datenbank (dieser Name ist mit dem Datenbanknamen identisch, wenn kein Aliasname angegeben wird)
- Einen Kommentar zur Datenbank, der mit dem Befehl CREATE DATABASE angegeben wurde
- Den Namen des Stammverzeichnisses für die Datenbank
- Andere Systeminformationen

Systemdatenbankverzeichnis

Für jede Instanz des Datenbankmanagers ist eine Datei mit einem *Systemdatenbankverzeichnis* vorhanden, die für jede Datenbank, die für dieser Instanz katalogisiert wurde, einen separaten Eintrag enthält.

Datenbanken werden implizit katalogisiert, wenn der Befehl CREATE DATABASE ausgegeben wird. Sie können auch explizit mit dem Befehl CATALOG DATABASE katalogisiert werden.

Für jede erstellte Datenbank wird dem Verzeichnis ein Eintrag hinzugefügt, der folgende Informationen enthält:

- Den Datenbanknamen, der mit dem Befehl CREATE DATABASE angegeben wurde
- Den Aliasnamen der Datenbank (dieser Name ist mit dem Datenbanknamen identisch, wenn kein Aliasname angegeben wird)
- Den Kommentar zur Datenbank, der mit dem Befehl CREATE DATABASE angegeben wurde
- Die Speicherposition des lokalen Datenbankverzeichnisses
- Einen Anzeiger, dass die Datenbank *indirekt* ist, d. h., dass sich die Datenbank in der aktuellen Datenbankmanagerinstanz befindet
- Andere Systeminformationen

Auf UNIX-Plattformen und in einer Umgebung mit partitionierten Datenbanken müssen Sie sicherstellen, dass alle Datenbankpartitionen jederzeit auf dieselbe Datei mit dem Systemdatenbankverzeichnis (sqldbdir) im Unterverzeichnis sqldbdir des Ausgangsverzeichnisses für die Instanz zugreifen. Unvorhersehbare Fehler können auftreten, wenn entweder das Systemdatenbankverzeichnis oder die Systemintentionsdatei sqldbins im selben Unterverzeichnis sqldbdir symbolische Verbindungen zu einer anderen Datei darstellen, die sich in einem gemeinsam benutzten Dateisystem befindet.

Erstellen von Knotenkonfigurationsdateien

Wenn Ihre Datenbank in einer partitionierten Datenbankumgebung arbeiten soll, müssen Sie eine Knotenkonfigurationsdatei mit dem Namen db2nodes.cfg erstellen.

Diese Datei muss sich im Unterverzeichnis sqllib des Ausgangsverzeichnisses (Home) für die Instanz befinden, bevor Sie den Datenbankmanager mit der Fähigkeit zur Parallelverarbeitung in mehreren Datenbankpartitionen starten können. Die Datei enthält Konfigurationsdaten für alle Datenbankpartitionen einer Instanz und wird von allen Datenbankpartitionen für diese Instanz gemeinsam genutzt.

Hinweise für Windows

Wenn Sie DB2 Enterprise Server Edition unter Windows verwenden, wird die Knotenkonfigurationsdatei beim Erstellen der Instanz erstellt. Sie sollten nicht ver-

suchen, eine Knotenkonfigurationsdatei manuell zu erstellen oder manuell zu ändern. Mithilfe des Befehls db2ncrt können Sie einer Instanz einen Datenbankpartitionsserver hinzufügen. Mithilfe des Befehls db2ndrop können Sie einen Datenbankpartitionsserver aus einer Instanz löschen. Mithilfe des Befehls db2nchg können Sie die Konfiguration der Datenbankpartitionsserver ändern. Dies umfasst das Versetzen des Datenbankpartitionsservers von einem Computer auf einen anderen, das Ändern des TCP/IP-Hostnamens oder das Auswählen eines anderen logischen Portnamens oder Netzwerknamens.

Anmerkung: Erstellen Sie keine anderen als die vom Datenbankmanager erstellten Dateien oder Verzeichnisse unter dem Unterverzeichnis sqllib, um Datenverluste zu vermeiden, wenn eine Instanz gelöscht wird. Es gibt jedoch zwei Ausnahmen. Wenn Ihr System gespeicherte Prozeduren unterstützt, stellen Sie die gespeicherten Prozeduranwendungen in das Unterverzeichnis 'function' im Unterverzeichnis sqllib. Die andere Ausnahme betrifft eventuell erstellte benutzerdefinierte Funktionen (UDFs). Benutzerdefinierte Funktionen können im selben Verzeichnis gespeichert werden.

Die Datei enthält eine Zeile für jede Datenbankpartition, die zu einer Instanz gehört. Jede Zeile hat folgendes Format:

dbpartitionnum hostname [*logischer-port* [*netzname*]]

Die Token einer Zeile sind durch Leerzeichen voneinander getrennt. Die Variablen sind:

dbpartitionsnummer
: Die Datenbankpartitionsnummer (mögliche Werte: 0 - 999) definiert eine Datenbankpartition eindeutig. Datenbankpartitionsnummern müssen in aufsteigender Reihenfolge angegeben sein. Es dürfen Sprünge in der Folge der Nummern auftreten.

  Wenn eine Datenbankpartitionsnummer einmal zugeordnet ist, kann sie nicht mehr geändert werden. (Ansonsten könnten die Informationen in der Verteilungszuordnung, die bestimmt, wie Daten verteilt werden, inkonsistent werden.)

  Wenn Sie eine Datenbankpartition löschen, kann ihre Datenbankpartitionsnummer für jede neue Datenbankpartition, die Sie hinzufügen, wieder verwendet werden.

  Die Datenbankpartitionsnummer wird zur Generierung eines Datenbankpartitionsnamens im Datenbankverzeichnis verwendet. Er hat folgendes Format:

  ```
  NODE nnnn
  ```

  Die Ziffernfolge *nnnn* ist die Datenbankpartitionsnummer, die links mit Nullen aufgefüllt wird. Diese Datenbankpartitionsnummer wird außerdem durch die Befehle CREATE DATABASE und DROP DATABASE verwendet.

hostname
: Der Hostname der IP-Adresse für die partitionsübergreifende Kommunikation. Verwenden Sie den vollständig qualifizierten Namen für den Hostnamen. In der Datei /etc/hosts sollte ebenfalls der vollständig qualifizierte Name verwendet werden. Wenn der vollständig qualifizierte Name in der Datei db2nodes.cfg und in der Datei /etc/hosts nicht verwendet wird, empfangen Sie eventuell die Fehlernachricht SQL30082N RC=3.

(Es gibt eine Ausnahme, wenn der Netzname angegeben wird. In diesem Fall wird der Netzname für den Großteil der Kommunikation verwendet, während Hostname nur für die Befehle db2start, db2stop und db2_all verwendet wird.)

logischer-port
Dieser Parameter ist optional und gibt die logische Portnummer für die Datenbankpartition an. Diese Nummer wird mit dem Instanznamen des Datenbankmanagers verwendet, um einen TCP/IP-Servicenamenseintrag in der Datei etc/services anzugeben.

Die Kombination aus IP-Adresse und logischem Port wird als allgemein bekannte Adresse verwendet und muss für alle Anwendungen eindeutig sein, um die Verbindungen zur Kommunikation zwischen Datenbankpartitionen zu unterstützen.

Für jeden Hostnamen muss ein *logischer-port* entweder 0 (null) oder leer sein (was standardmäßig dem Wert 0 entspricht). Die Datenbankpartition, der dieser *logische-port* zugeordnet ist, ist der Standardknoten auf dem Host, zum dem Clients die Verbindung herstellen. Diese Einstellung kann mithilfe der Umgebungsvariablen **DB2NODE** im Script db2profile oder mit der API sqlesetc() überschrieben werden.

netzname
Dieser Parameter ist optional und wird zur Unterstützung eines Hosts verwendet, der über mehr als eine aktive TCP/IP-Schnittstelle verfügt, von denen jede ihren eigenen Hostnamen hat.

Das folgende Beispiel zeigt eine mögliche Knotenkonfigurationsdatei für ein System RS/6000 SP, in dem SP2EN1 mehrere TCP/IP-Schnittstellen und zwei logische Datenbankpartitionen hat und SP2SW1 als DB2-Datenbankschnittstelle verwendet. Es zeigt außerdem die Datenbankpartitionsnummern, beginnend bei 1 (und nicht bei 0), sowie einen Sprung in der Folge der *dbpartitionsnummern*:

Tabelle 40. Eine Tabelle mit einem Beispiel für Datenbankpartitionsnummern

dbpartitionsnummer	*hostname*	*logischer-port*	*netzname*
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

Die Datei db2nodes.cfg kann mit einem beliebigen Editor aktualisiert werden. (Ausnahme: Unter Windows sollte kein Editor verwendet werden.) Sie müssen jedoch sorgfältig auf den Schutz der Integrität der Daten in der Datei achten, da die Datenbankpartitionierung erfordert, dass die Knotenkonfigurationsdatei gesperrt wird, wenn der Befehl db2start ausgeführt wird, und entsperrt wird, wenn der Befehl db2stop den Datenbankmanager beendet hat. Der Befehl db2start kann die Datei bei Bedarf aktualisieren, während sie gesperrt ist. Sie können beispielsweise den Befehl db2start mit der Option **RESTART** oder **ADD DBPARTITIONNUM** ausführen.

Anmerkung: Wenn der Befehl db2stop nicht erfolgreich ausgeführt und die Knotenkonfigurationsdatei nicht entsperrt wird, führen Sie den Befehl db2stop **FORCE** aus, um sie zu entsperren.

Ändern der Knoten- und Datenbankkonfigurationsdatei

Führen Sie zum Aktualisieren der Datenbankkonfigurationsdatei den Befehl AUTOCONFIGURE mit den entsprechenden Optionen aus.

Der Konfigurationsadvisor hilft Ihnen bei der Optimierung der Leistung und der ausgewogenen Verteilung des Speicherbedarfs für eine einzelne Datenbank pro Instanz und macht Vorschläge zur Änderung von Konfigurationsparametern und Empfehlungen zu geeigneten Werten.

Wenn Sie Änderungen an bestimmten Datenbankpartitionsgruppen (Hinzufügen oder Löschen von Datenbankpartitionen oder Versetzen vorhandener Datenbankpartitionen) planen, muss die Knotenkonfigurationsdatei aktualisiert werden. Wenn Sie Änderungen an der Datenbank planen, sollten Sie die Werte der Konfigurationsparameter überprüfen. Einige Werte können von Zeit zu Zeit im Rahmen der laufenden Änderungen an der Datenbank je nach Benutzung angepasst werden.

Anmerkung: Wenn Sie Parameter ändern, werden die entsprechenden Werte erst dann aktualisiert, wenn die folgenden Ereignisse stattfinden:

- Bei Datenbankparametern, wenn die erste neue Verbindung zur Datenbank hergestellt wird, nachdem alle Anwendungen getrennt waren.
- Bei Datenbankmanagerparametern, wenn die Instanz das nächste Mal gestoppt und wieder gestartet wird.

In den meisten Fällen führen die vom Konfigurationsadvisor empfohlenen Werte zu einer besseren Leistung als die Standardwerte, weil sie auf den Informationen zur Auslastung Ihres Systems und zu Ihrem speziellen Server basieren. Beachten Sie jedoch, dass die Werte die Leistung Ihres Datenbanksystems verbessern, aber nicht unbedingt optimieren können. Sie sollten deshalb als Ausgangspunkt für weitere Optimierungsmaßnahmen verstanden werden.

In Version 9.1 wird der Konfigurationsadvisor automatisch aufgerufen, wenn Sie eine Datenbank erstellen. Zur Inaktivierung dieser Funktion oder zur expliziten Aktivierung dieser Funktion verwenden Sie den Befehl db2set, bevor Sie die Datenbank erstellen. Beispiele:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

Weitere Funktionen, die standardmäßig aktiviert sind, finden Sie in „Automatische Funktionen” auf Seite 17.

Wenn Sie den Konfigurationsadvisor über die Befehlszeile verwenden wollen, verwenden Sie den Befehl AUTOCONFIGURE.

Geben Sie in die Befehlszeile den folgenden Befehl ein, um einzelne Parameter in der Konfiguration des Datenbankmanagers zu aktualisieren:

```
UPDATE DBM CFG USING <konfig_schlüsselwort>=<wert>
```

Geben Sie in der Befehlszeile Folgendes ein, um einzelne Parameter in der Datenbankkonfiguration zu aktualisieren:

```
UPDATE DB CFG FOR <aliasname_der_datenbank>
   USING <konfig_schlüsselwort>=<wert>
```

Mit einem einzigen Befehl können Sie eine oder mehrere Kombinationen aus <konfig_schlüsselwort>=<wert> ändern. Die meisten Änderungen an der

Konfigurationsdatei des Datenbankmanagers werden erst wirksam, nachdem Sie in den Speicher geladen wurden. Bei einem Konfigurationsparameter für einen Server wird diese Operation während der Ausführung des Befehls START DATABASE MANAGER durchgeführt. Bei einem Konfigurationsparameter für einen Client erfolgt die Ausführung dieser Operation beim Neustart der Anwendung.

Zum Anzeigen oder Ausgeben der aktuellen Konfigurationsparameter für den Datenbankmanager können Sie den Befehl GET DATABASE MANAGER CONFIGURATION verwenden.

Zum Zugriff auf den Konfigurationsadvisor aus einer Clientanwendung heraus rufen Sie die API 'db2AutoConfig' auf. Zur Aktualisierung einzelner Parameter in der Konfigurationsdatei des Datenbankmanagers oder der Datenbank aus einer Clientanwendung heraus rufen Sie die API 'db2CfgSet' auf.

Datenbankrecoveryprotokoll

Ein *Datenbankrecoveryprotokoll* zeichnet Datensätze über alle an einer Datenbank vorgenommenen Änderungen auf, wozu auch das Hinzufügen neuer Tabellen oder das Aktualisieren vorhandener Tabellen gehören.

Dieses Protokoll besteht aus einer Reihe von *Protokollspeicherbereichen*, die jeweils in einer separaten, so genannten *Protokolldatei* enthalten sind.

Das Datenbankrecoveryprotokoll kann verwendet werden, um sicherzustellen, dass eine Datenbank nach einem Fehler (z. B. ein Netzstromausfall im System oder ein Anwendungsfehler) nicht in einem inkonsistenten Zustand bleibt. Im Fall eines Fehlers werden bereits durchgeführte Änderungen, die noch nicht mit COMMIT festgeschrieben wurden, mit ROLLBACK rückgängig gemacht; alle festgeschriebenen Transaktionen, die eventuell noch nicht physisch auf die Platte geschrieben wurden, werden wiederhergestellt. Durch diese Maßnahmen wird die Integrität der Datenbank gewährleistet.

Speicherbedarf für Datenbankobjekte

Das Abschätzen der Größe von Datenbankobjekten ist ein ungenaues Unterfangen. Der Systemaufwand, der durch Datenträgerfragmentierung, freien Speicherbereich und die Verwendung von Spalten variabler Länge bedingt ist, macht eine Abschätzung schwierig, weil es eine große Bandbreite an Möglichkeiten für Spaltentypen und Zeilenlängen gibt.

Erstellen Sie nach dem Abschätzen der Größe Ihrer Datenbank eine Testdatenbank, und füllen Sie sie mit repräsentativen Daten. Verwenden Sie dann das Dienstprogramm db2look, um Datendefinitionsanweisungen für die Datenbank zu generieren.

Bei der Abschätzung der Größe einer Datenbank müssen die Auswirkungen der folgenden Elemente berücksichtigt werden:

- Systemkatalogtabellen
- Benutzertabellendaten
- LF-Daten (LF = Long Field; Langfeld)
- LOB-Daten (LOB = Large Object; großes Objekt)
- Indexbereich
- Protokolldateispeicher
- Temporärer Arbeitsspeicherbereich

Berücksichtigen Sie außerdem den Aufwand und den Platzbedarf für folgende Komponenten:

- Datei für das lokale Datenbankverzeichnis
- Datei für das Systemdatenbankverzeichnis
- Bei der Dateiverwaltung entstehender Systemaufwand, den das Betriebssystem benötigt, zum Beispiel:
  - Dateiblockgröße
  - Speicherbereich für die Verzeichnissteuerung

Speicherbedarf für Protokolldateien

Sie benötigen 56 KB Speicherplatz für Protokollsteuerdateien.

Außerdem benötigen Sie mindestens ausreichend Speicherplatz für Ihre aktive Protokollkonfiguration, die Sie nach folgender Formel berechnen können:

```
(logprimary + logsecond) * (logfilsiz + 2 ) * 4096
```

Dabei gilt Folgendes:

- *logprimary* ist die Anzahl der primären Protokolldateien, die in der Konfigurationsdatei der Datenbank definiert sind.
- *logsecond* ist die Anzahl der sekundären Protokolldateien, die in der Konfigurationsdatei der Datenbank definiert sind. In dieser Formel kann *logsecond* nicht auf den Wert *-1* gesetzt werden. (Wenn *logsecond* auf den Wert -1 gesetzt wird, fordern Sie einen unbegrenzten Speicherplatz für aktive Protokolldateien an.)
- *logfilsiz* ist die Anzahl der Seiten in jeder Protokolldatei, die in der Konfigurationsdatei der Datenbank definiert sind.
- 2 ist die Anzahl der Kopfseiten, die für jede Protokolldatei erforderlich ist.
- 4096 ist die Anzahl der Byte einer Seite.

Wenn für die Datenbank die Umlaufprotokollierung aktiviert ist, liefert das Ergebnis dieser Formel den Betrag des gesamten Speicherplatzes, der für die Protokollierung zugeordnet wird; d. h., mehr Speicherplatz wird nicht zugeordnet, und für keine Ihrer Protokolldateien werden Fehler über nicht ausreichenden Plattenspeicherplatz ausgegeben.

Wenn für die Datenbank die aktualisierende Recovery aktiviert ist, sollten spezielle Speicheranforderungen für Protokolle berücksichtigt werden:

- Ist der Konfigurationsparameter *logarchmeth1* auf 'logretain' gesetzt, werden die Protokolldateien im Protokollpfadverzeichnis archiviert. Der Online-Plattenspeicherplatz wird schließlich belegt, sofern Sie die Protokolldateien nicht an eine andere Speicherposition verschieben.
- Ist der Konfigurationsparameter *logarchmeth1* auf 'userexit', DISK oder VENDOR gesetzt, verschiebt ein Benutzerexitprogramm die archivierten Protokolldateien an eine andere Speicherposition. Zusätzlicher Speicherbereich ist weiterhin erforderlich für folgende Dateien:
  - Online archivierte Protokolle, die darauf warten, vom Benutzerexitprogramm verschoben zu werden
  - Neue Protokolle, die für künftige Zwecke formatiert werden

Wenn für die Datenbank eine unendliche Protokollierung aktiviert ist (d. h., wenn Sie den Parameter *logsecond* auf den Wert *-1* setzen), muss der Konfigurationsparameter *logarchmeth1* auf einen anderen Wert als OFF oder LOGRETAIN gesetzt

werden, damit die Archivprotokollierung aktiviert werden kann. Der Datenbankmanager behält mindestens die durch den Parameter *logprimary* definierte Anzahl aktiver Protokolldateien im Protokollpfad; Sie sollten daher in der obigen Formel für *logsecond* nicht den Wert -1 einsetzen. Stellen Sie sicher, dass Sie zusätzlichen Speicherplatz bereitstellen, um der durch die Archivierung von Protokolldateien verursachten Verzögerung Rechnung zu tragen.

Wenn Sie den Protokollpfad spiegeln, müssen Sie den Schätzwert für den Protokolldateispeicherbedarf verdoppeln.

LDAP-Verzeichnisservice (Lightweight Directory Access Protocol)

Bei einem Verzeichnisservice handelt es sich um ein Repository mit Ressourceninformationen zu mehreren Systemen und Services innerhalb einer verteilten Umgebung. Er stellt den Client- und Serverzugriff auf diese Ressourcen bereit.

Clients und Server verwenden den Verzeichnisservice in der Regel, um festzustellen, wie sie auf andere Ressourcen zugreifen können. Informationen zu diesen anderen Ressourcen in der verteilten Umgebung müssen in das Repository des Verzeichnisservice eingegeben werden.

Lightweight Directory Access Protocol (LDAP) ist eine dem Branchenstandard entsprechende Methode für den Zugriff auf Verzeichnisservices. Jede Datenbankserverinstanz veröffentlicht Informationen über ihre Existenz auf einem LDAP-Server und stellt dem LDAP-Verzeichnis Datenbankinformationen zur Verfügung, wenn die Datenbanken erstellt werden. Wenn ein Client eine Verbindung zur Datenbank herstellt, können die Kataloginformationen für den Server aus dem LDAP-Verzeichnis abgerufen werden. Die einzelnen Clients müssen die Katalog· informationen nun nicht mehr lokal auf den verschiedenen Computern speichern. Clientanwendungen durchsuchen das LDAP-Verzeichnis nach den erforderlichen Informationen für die Herstellung der Verbindung zur Datenbank.

Anmerkung: Das Veröffentlichen der Datenbankserverinstanz auf dem LDAP-Server ist kein automatischer Prozess, sondern muss vom Administrator manuell ausgeführt werden.

Als Administrator eines DB2-Systems können Sie einen Verzeichnisservice einrichten und verwalten. Zur Pflege dieses Verzeichnisservice können Sie den Konfigurationsassistenten zu Hilfe nehmen. Der Verzeichnisservice wird für den DB2-Datenbankmanager durch LDAP-Verzeichnisservices (Lightweight Directory Access Protocol) verfügbar gemacht. Zur Verwendung von LDAP-Verzeichnisservices muss zunächst ein LDAP-Server vorhanden sein, der von einem DB2-Datenbankmanager unterstützt wird, sodass dort Verzeichnisinformationen gespeichert werden können.

Anmerkung: In einer Domänenumgebung unter Windows steht bereits ein LDAP-Server zur Verfügung, da er in Windows Active Directory integriert ist. Infolgedessen kann jeder Computer, der mit Windows arbeitet, LDAP verwenden.

Ein LDAP-Verzeichnis ist in einer Unternehmensumgebung nützlich, in der es aufgrund einer großen Anzahl von Clients schwierig ist, lokale Verzeichniskataloge auf jedem Client-Computer zu aktualisieren. In einer solchen Situation wird empfohlen, die Verzeichniseinträge auf einem LDAP-Server zu speichern, sodass die Verwaltung von Katalogeinträgen zentral über einen Punkt, d. h. über den LDAP-Server, erfolgen kann.

Erstellen von Datenbanken

Sie können eine Datenbank mit dem Befehl CREATE DATABASE erstellen. Zur Erstellung einer Datenbank aus einer Clientanwendung heraus rufen Sie die API sqlecrea auf.

Vor der Erstellung einer Datenbank sollten Sie genügend Zeit auf die Planung des Inhalts, des Layouts, des potenziellen Anwachsens sowie auf die gewünschte Nutzung der Datenbank verwenden.

Die folgenden Datenbankzugriffsrechte werden der Gruppe PUBLIC automatisch erteilt: CREATETAB, BINDADD, CONNECT, IMPLICIT_SCHEMA und SELECT für die Systemkatalogsichten. Wenn jedoch die Option RESTRICTIVE angegeben wird, werden der Gruppe PUBLIC keine Zugriffsrechte automatisch erteilt. Weitere Informationen zur Option RESTRICTIVE finden Sie in den Informationen zum Befehl CREATE DATABASE.

Beim Erstellen einer Datenbank werden die folgenden Tasks für Sie ausgeführt:

- Einrichten aller Systemkatalogtabellen, die von der Datenbank benötigt werden
- Zuordnen des Datenbankrecoveryprotokolls
- Erstellen der Datenbankkonfigurationsdatei und Definieren der Standardwerte
- Binden der Datenbankdienstprogramme an die Datenbank

Geben Sie in den Befehlszeilenprozessor Folgendes ein, um eine Datenbank zu erstellen:

```
CREATE DATABASE <datenbankname>
```

Mit dem folgenden Befehl wird zum Beispiel eine Datenbank namens PERSON1 an der Standardspeicherposition mit dem zugeordneten Kommentar ″Personnel DB for BSchiefer Co″ erstellt:

```
CREATE DATABASE personl
  WITH "Personnel DB for BSchiefer Co"
```

Konfigurationsadvisor

Der Konfigurationsadvisor hilft Ihnen bei der Optimierung der Leistung und der ausgewogenen Verteilung des Speicherbedarfs für eine einzelne Datenbank pro Instanz, indem er Konfigurationsparameter zur Änderung vorschlägt und geeignete Werte für sie empfiehlt. Der Konfigurationsadvisor wird automatisch aufgerufen, wenn Sie eine Datenbank erstellen. Zur Inaktivierung dieser Funktion oder zur expliziten Aktivierung dieser Funktion verwenden Sie den Befehl db2set, bevor Sie die Datenbank erstellen. Beispiele:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

Weitere Funktionen, die standardmäßig aktiviert sind, finden Sie in „Automatische Funktionen” auf Seite 17.

Ereignismonitor

Bei der Erstellung einer Datenbank wird gleichzeitig auch ein detaillierter Ereignismonitor für Deadlocks erstellt. Wie bei jedem Monitor fällt für diesen Ereignismonitor etwas Systemaufwand an. Wenn der detaillierte Ereignismonitor für Deadlocks nicht aktiviert sein soll, kann er mit dem folgenden Befehl gelöscht werden:

```
DROP EVENT MONITOR db2detaildeadlock
```

Zur Begrenzung der Größe des Plattenspeicherplatzes, den dieser Ereignismonitor beansprucht, wird der Ereignismonitor inaktiviert und eine Nachricht wird in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben, wenn die maximale Anzahl von Ausgabedateien erreicht ist. Durch das Entfernen von Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

Ferne Datenbanken

Sie haben die Möglichkeit, eine Datenbank in einer anderen, möglicherweise fernen Instanz zu erstellen. Um eine Datenbank in einem anderen (fernen) Datenbankpartitionsserver zu erstellen, müssen Sie zuerst eine Verbindung zu diesem Server herstellen. Eine Datenbankverbindung wird während der Verarbeitung mit dem folgenden Befehl temporär hergestellt:

```
CREATE DATABASE <datenbankname> AT DBPARTITIONNUM <optionen>
```

In diesem Typ von Umgebung können Sie Verwaltungsoperationen auf Instanzebene an einer anderen Instanz als Ihrer Standardinstanz, einschließlich ferner Instanzen, ausführen. Anweisungen zur Vorgehensweise finden Sie in den Informationen zum Befehl db2iupdt (Instanz aktualisieren).

Datenbankcodepages

Standardmäßig werden Datenbanken im codierten Zeichensatz UTF-8 (Unicode) erstellt.

Wenn Sie die Standardcodepage für die Datenbank überschreiben möchten, muss der gewünschte codierte Zeichensatz und das entsprechende Gebiet bei der Erstellung der Datenbank angegeben werden. Informationen zur Angabe des codierten Zeichensatzes und des Gebiets finden Sie in den Informationen zum Befehl CREATE DATABASE bzw. zur API sqlecrea.

Datenbanken mit dynamischem Speicher

Der Datenbankmanager erstellt standardmäßig alle Datenbanken *mit dynamischem Speicher*. Wenn Sie eine Datenbank „ohne dynamischen Speicher" erstellen wollen, müssen Sie im Befehl CREATE DATABASE die Klausel AUTOMATIC STORAGE NO angeben.

Datenbanken, für die der dynamische Speicher aktiviert ist, besitzen einen zugeordneten Speicherpfad bzw. eine Gruppe zugeordneter Speicherpfade. Auf der Grundlage dieser Speicherpfade können Sie Tabellenbereiche als *vom dynamischen Speicher verwaltet* und die zugehörigen Container als durch den Datenbankmanager zugeordnet definieren.

Sie können den dynamischen Speicher für eine Datenbank nur bei der Erstellung der Datenbank aktivieren. Ebenso können Sie den dynamischen Speicher für eine Datenbank nicht inaktivieren, für die dessen Verwendung ursprünglich definiert wurde.

Alle Datenbanken werden standardmäßig mit dynamischem Speicher erstellt. Wenn Sie eine Datenbank ohne dynamischen Speicher erstellen wollen, müssen Sie die Option **AUTOMATIC STORAGE NO** im Befehl CREATE DATABASE angeben.

Beispielbefehle für die Erstellung von Datenbanken ohne dynamischen Speicher:

```
CREATE DATABASE ASNODB1 AUTOMATIC STORAGE NO
CREATE DATABASE ASNODB2 AUTOMATIC STORAGE NO ON X:
```

Beispielbefehle für die Erstellung von Datenbanken mit expliziter oder impliziter Aktivierung des dynamischen Speichers:

```
CREATE DATABASE DB1
CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:
CREATE DATABASE DB3 ON /data/pfad1, /data/pfad2
CREATE DATABASE DB4 ON D:\SpeicherPfad DBPATH ON C:
```

Auf der Basis der verwendeten Syntax extrahiert der Datenbankmanager die beiden folgenden Informationen, die sich auf Speicherpositionen beziehen:

- Den Datenbankpfad (in dem der Datenbankmanager eine Reihe von Steuerdateien für die Datenbank speichert):
  - Wenn Sie **DBPATH ON** angeben, geben Sie damit den Datenbankpfad an.
  - Wenn Sie **DBPATH ON** nicht angeben, gibt der erste Pfad in **ON** den Datenbankpfad (und den Speicherpfad) an.
  - Wenn Sie weder **DBPATH ON** noch **ON** angeben, wird der Konfigurationsparameter **dftdbpath** des Datenbankmanagers zur Bestimmung des Datenbankpfades verwendet.
- Die Speicherpfade (in denen der Datenbankmanager Tabellenbereichscontainer mit dynamischem Speicher erstellt):
  - Wenn Sie **ON** angeben, sind alle aufgeführten Pfade Speicherpfade.
  - Wenn Sie **ON** nicht angeben, wird nur ein Speicherpfad verwendet, der dem Wert des Konfigurationsparameters **dftdbpath** des Datenbankmanagers entspricht.

In der folgenden Tabelle sind die verwendeten Datenbank- und Speicherpfade für die oben gezeigten Beispiele zusammengefasst:

Tabelle 41. Dynamischer Speicher - Datenbankpfade und Speicherpfade

Befehl CREATE DATABASE	Datenbankpfad	Speicherpfade
`CREATE DATABASE DB1 AUTOMATIC STORAGE YES`	Wert des Konfigurationsparameters **dftdbpath**	Wert des Konfigurationsparameters **dftdbpath**
`CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:`	X:	X:
`CREATE DATABASE DB3 ON /data/pfad1, /data/pfad2`	/data/pfad1	/data/pfad1, /data/pfad2
`CREATE DATABASE DB4 ON D:\SpeicherPfad DBPATH ON C:`	C:	D:\SpeicherPfad

Die angegebenen Speicherpfade müssen vorhanden und zugänglich sein. In einer Umgebung mit partitionierten Datenbanken werden in jeder Datenbankpartition die gleichen Speicherpfade verwendet. Sie können keine eindeutige Gruppe von Speicherpfaden für eine bestimmte Datenbankpartition angeben, sofern Sie nicht Datenbankpartitionsausdrücke als Teil der Speicherpfadnamen verwenden. Dieses Verfahren bietet die Möglichkeit, die Datenbankpartitionsnummer in die Speicherpfade einzufügen, sodass die sich ergebenden Pfadnamen für jede Datenbankpartition unterschiedlich ist.

Verwenden Sie das Argument `$N` (mit einem Leerzeichen vor `$N`!), um einen Datenbankpartitionsausdruck anzugeben. Sie können einen Datenbankpartitionsausdruck an einer beliebigen Stelle im Speicherpfad verwenden, und Sie können

mehrere Datenbankpartitionsausdrücke angeben. Schließen Sie den Datenbankpartitionsausdruck mit einem Leerzeichen ab. Alle weiteren Zeichen nach dem Leerzeichen werden an den Speicherpfad angehängt, nachdem der Datenbankpartitionsausdruck ausgewertet wurde. Wenn nach dem Datenbankpartitionsausdruck im Speicherpfad kein Leerzeichen folgt, wird der Rest der Zeichenfolge als Teil des Ausdrucks interpretiert. In der folgenden Tabelle sind die einzigen gültigen Formen des Arguments `$N` aufgeführt. Operatoren werden von links nach rechts ausgewertet, und das Zeichen % stellt den Modulusoperator dar. Die Datenbankpartitionsnummer in den Beispielen ist 10.

Tabelle 42. Datenbankpartitionsausdrücke

Syntax	Beispiel	Wert
[Leerzeichen]$N	`" $N"`	10
[Leerzeichen]$N+[zahl]	`" $N+100"`	110
[Leerzeichen]$N%[zahl]	`" $N%5"`	0
[Leerzeichen]$N+[zahl]%[zahl]	`" $N+1%5"`	1
[Leerzeichen]$N%[zahl]+[zahl]	`" $N%4+2"`	4

Das folgende Beispiel zeigt die Verwendung von Datenbankpartitionsausdrücken:

```
CREATE DATABASE TESTDB ON "/pfad1FuerKnoten $N",
  "/pfad2FuerKnoten $N" DBPATH ON "/dbpfadFuerKnoten"
```

Das folgende Beispiel zeigt einen in der Mitte eines Pfads eingebetteten Datenbankpartitionsausdruck:

```
CREATE DATABASE TESTDB ON "/pfad1FuerKnoten $N",
  "/pfad2FuerKnoten $N suffix" DBPATH ON "/dbpfadFuerKnoten"
```

Anmerkung: Datenbankpartitionsausdrücke sind in Datenbankpfaden nicht gültig, unabhängig davon, ob sie explizit mit **DBPATH ON** oder implizit durch einen Datenbankpartitionsausdruck im ersten Speicherpfad angegeben werden.

Bei der Berechnung des freien Speicherbereichs für einen Speicherpfad für eine gegebene Datenbankpartition prüft der Datenbankmanager, ob die folgenden Verzeichnisse (bzw. Mountpunkte) innerhalb des Speicherpfads vorhanden sind, und verwendet das erste Verzeichnis, das er findet:

```
speicherpfad/instanzname/NODE####/datenbankname
speicherpfad/instanzname/NODE####
speicherpfad/instanzname
speicherpfad
```

Dabei gilt Folgendes:

speicherpfad
: Ein der Datenbank zugeordneter Speicherpfad

instanzname
: Die Instanz, unter der sich die Datenbank befindet

NODE####
: Die Datenbankpartitionsnummer (z. B. NODE0000 oder NODE0001)

datenbankname
: Der Name der Datenbank

Dateisysteme können an einem Punkt unterhalb des Speicherpfads angehängt sein. In diesem Fall erkennt der Datenbankmanager, dass die tatsächliche Kapazität an

freiem Speicherbereich, die für Tabellenbereichscontainer verfügbar ist, möglicherweise nicht die gleiche Kapazität ist, die dem Speicherpfadverzeichnis selbst zugeordnet ist.

Betrachten Sie ein Beispiel, in dem zwei logische Datenbankpartitionen auf einem physischen Computer angelegt sind und nur ein Speicherpfad vorhanden ist: /db2data. Beide Datenbankpartitionen können diesen einen Speicherpfad verwenden. Sie möchten jedoch möglicherweise die Daten jeder Datenbankpartition isolieren, indem Sie für jede jeweils ein eigenes Dateisystem erstellen. Das Dateisystem wird an /db2data/*instanz*/NODE#### angehängt. Beim Erstellen von Containern im Speicherpfad und beim Ermitteln des freien Speicherbereichs ruft der Datenbankmanager nicht die Informationen über den freien Speicherbereich im Pfad /db2data, sondern die Informationen für das entsprechende Verzeichnis /db2data/*instanz*/NODE#### ab.

Jedes Mal, wenn Sie eine Datenbank erstellen, werden drei Standardtabellenbereiche erstellt. Wenn Sie im Befehl CREATE DATABASE keine expliziten Tabellenbereichsdefinitionen angeben, werden die Tabellenbereiche als Tabellenbereiche mit dynamischem Speicher erstellt.

Nach der Erstellung einer Datenbank können Sie ihr neue Speicherpfade hinzufügen, indem Sie die Klausel ADD STORAGE der Anweisung ALTER DATABASE wie im folgenden Beispiel verwenden:

```
ALTER DATABASE ADD STORAGE ON '/data/pfad3', '/data/pfad4'
```

Einschränkungen für dynamischen Speicher

Bei der Entscheidung, ob eine Datenbank mit dynamischem Speicher erstellt werden soll, sind einige Einschränkungen zu berücksichtigen.

- Sie können den dynamischen Speicher für eine Datenbank nach ihrer Erstellung nicht inaktivieren oder aktivieren.
- Speicherpfade müssen absolute Pfadnamen sein. Unter dem Windows-Betriebssystem sind Pfade oder Laufwerkbuchstaben möglich. Der Datenbankpfad muss ein Laufwerkbuchstabe sein. Die maximal zulässige Pfadlänge beträgt 175 Zeichen.
- Für partitionierte Datenbanken müssen Sie in jeder Datenbankpartition die gleiche Gruppe von Speicherpfaden verwenden (sofern Sie keine Datenbankpartitionsausdrücke verwenden).
- Datenbankpartitionsausdrücke sind in Datenbankpfaden nicht gültig, unabhängig davon, ob Sie sie explizit mit der Option **DBPATH ON** des Befehls CREATE DATABASE oder implizit durch einen Datenbankpartitionsausdruck im ersten Speicherpfad angeben.

Hinzufügen von Pfaden für den dynamischen Speicher in Datenbanken mit aktiviertem dynamischen Speicher

Mit der Anweisung ALTER DATABASE können Sie einer Datenbank, die für den dynamischen Speicher eingerichtet ist, einen Pfad für den dynamischen Speicher hinzufügen. Sie können den dynamischen Speicher für eine Datenbank nur bei der Erstellung der Datenbank aktivieren.

Wenn Sie einen Speicherpfad für eine Umgebung mit mehreren Datenbankpartitionen hinzufügen, muss der Speicherpfad in jeder Datenbankpartition vorhanden sein. Wenn der angegebene Pfad nicht in jeder Datenbankpartition vorhanden ist, wird die Anweisung rückgängig gemacht.

Geben Sie die Anweisung ALTER DATABASE wie folgt ein, um einer vorhandenen Datenbank einen Speicherpfad hinzuzufügen:

```
ALTER DATABASE PATH pfadname
```

Überwachen der Speicherpfade

Eine Momentaufnahme der Datenbank enthält die Liste der Speicherpfade, die der Datenbank zugeordnet sind.

Beträgt die Anzahl der Pfade für den dynamischen Speicher 0, ist der dynamische Speicher für diese Datenbank nicht aktiviert:

```
Anzahl Pfade für dynamischen Speicher         = ##
   Pfad für dynamischen Speicher              = <erster Pfad>
   Pfad für dynamischen Speicher              = <zweiter Pfad>
...
```

Wenn der Monitorschalter für den Pufferpool aktiviert ist, werden die folgenden Elemente ebenfalls mit Werten belegt:

```
Dateisystem-ID                                  = 12345
   Freier Speicherbereich des Dateisystems (Byte)  = 20000000000
   Belegter Speicherplatz des Dateisystems (Byte)  = 40000000000000
   Gesamter Speicherplatz des Dateisystems (Byte)  = 40020000000000
```

Diese Daten werden pro Pfad definiert: in einem Datenbanksystem mit Einzeldatenbankpartition pro Pfad, in einer Umgebung mit mehreren Datenbankpartitionen für jede Datenbankpartition.

Darüber hinaus werden in einer Momentaufnahme für einen Tabellenbereich folgende Informationen definiert. Die Informationen geben an, ob der Tabellenbereich als Tabellenbereich mit dynamischem Speicher erstellt wurde:

```
Dynamischen Speicher verwenden             = Ja oder Nein
```

Auswirkungen des Befehls RESTORE DATABASE

Der Befehl RESTORE DATABASE dient dazu, eine Datenbank von einem Backup-Image wiederherzustellen.

Während einer Restoreoperation ist es möglich, die Position des Datenbankpfads zu wählen und darüber hinaus die Speicherpfade erneut zu definieren, die der Datenbank zugeordnet sind. Der Datenbankpfad und die Speicherpfade werden definiert, indem im Befehl RESTORE DATABASE eine Kombination der Klauseln TO, ON und DBPATH ON verwendet wird.

Nachfolgend sind einige Beispiele für gültige Befehle RESTORE für Datenbanken aufgeführt, für die dynamischer Speicher aktiviert ist:

```
RESTORE DATABASE TEST1
RESTORE DATABASE TEST2 TO X:
RESTORE DATABASE TEST3 DBPATH ON D:
RESTORE DATABASE TEST3 ON /pfad1, /pfad2, /pfad3
RESTORE DATABASE TEST4 ON E:\neupfad1, F:\neupfad2 DBPATH ON D:
```

Wie auch beim Befehl CREATE DATABASE extrahiert der Datenbankmanager die beiden folgenden Informationen, die sich auf Speicherpositionen beziehen:

- Den **Datenbankpfad** (in dem der Datenbankmanager eine Reihe von Steuerdateien für die Datenbank speichert)
  - Wenn die Klauseln TO oder DBPATH ON angegeben werden, geben diese Klauseln den Datenbankpfad an.

– Wird die Klausel ON verwendet, DBPATH ON jedoch nicht zusammen mit dieser angegeben, wird der erste in der Klausel ON aufgeführte Pfad als Datenbankpfad verwendet (zusätzlich zu seiner Eigenschaft als Speicherpfad).
– Wird keine der Klauseln TO, ON oder DBPATH ON angegeben, bestimmt der Konfigurationsparameter *dftdbpath* des Datenbankmanagers den Datenbankpfad.

Anmerkung: Wenn eine Datenbank mit demselben Namen bereits auf der Platte vorhanden ist, wird der Datenbankpfad ignoriert, und die Datenbank wird im selben Verzeichnis gespeichert wie die vorhandene Datenbank.

- Die **Speicherpfade** (in denen der Datenbankmanager Tabellenbereichscontainer mit dynamischem Speicher erstellt)
  – Wenn die Klausel ON angegeben wird, werden alle aufgeführten Pfade als Speicherpfade interpretiert, und diese Pfade werden anstelle der im Backup-Image gespeicherten Pfade verwendet.
  – Wird die Klausel ON nicht angegeben, werden an den Speicherpfaden keine Änderungen vorgenommen (die im Backup-Image gespeicherten Speicherpfade werden beibehalten).

Zur Verdeutlichung dieses Konzepts werden die fünf oben gezeigten Beispiele für den Befehl RESTORE in der folgenden Tabelle mit den entsprechenden Speicherpfaden dargestellt:

Tabelle 43. Auswirkungen des Befehls RESTORE auf Datenbank- und Speicherpfade

Befehl RESTORE DATABASE	Keine Datenbank mit demselben Namen auf der Platte vorhanden		Datenbank mit demselben Namen auf der Platte vorhanden	
	Datenbankpfad	**Speicherpfade**	**Datenbankpfad**	**Speicherpfade**
`RESTORE DATABASE TEST1`	*<dftdbpath>*	Die im Backup-Image definierten Speicherpfade werden verwendet.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Die im Backup-Image definierten Speicherpfade werden verwendet.
`RESTORE DATABASE TEST2 TO X:`	X:	Die im Backup-Image definierten Speicherpfade werden verwendet.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Die im Backup-Image definierten Speicherpfade werden verwendet.
`RESTORE DATABASE TEST3 DBPATH ON /db2/databases`	/db2/databases	Die im Backup-Image definierten Speicherpfade werden verwendet.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Die im Backup-Image definierten Speicherpfade werden verwendet.
`RESTORE DATABASE TEST4 ON /pfad1, /pfad2, /pfad3`	/pfad1	/pfad1, /pfad2, /pfad3	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	/pfad1, /pfad2, /pfad3
`RESTORE DATABASE TEST5 ON E:\neupfad1, F:\neupfad2 DBPATH ON D:`	D:	E:\neupfad1, F:\neupfad2	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	E:\neupfad1, F:\neupfad2

In den Fällen, in denen Speicherpfade als Teil der Restoreoperation erneut definiert wurden, werden die Tabellenbereiche, für die die Verwendung von dynamischem Speicher definiert wurde, automatisch an die neuen Pfade umgeleitet. Es ist jedoch nicht möglich, Container, die Tabellenbereichen mit dynamischem Speicher zugeordnet sind, mit dem Befehl SET TABLESPACE CONTAINERS explizit umzuleiten. Eine solche Aktion ist nicht zulässig.

Verwenden Sie die Option `-s` des Befehls db2ckbkp, um zu prüfen, ob der dynamische Speicher für eine Datenbank in einem Backup-Image aktiviert ist. Wenn der dynamische Speicher aktiviert ist, werden die der Datenbank zugeordneten Speicherpfade angezeigt.

Für Datenbanken mit mehreren Datenbankpartitionen, für die der dynamische Speicher aktiviert ist, hat der Befehl RESTORE DATABASE einige zusätzliche Auswirkungen:

1. Die Datenbank muss die gleiche Gruppe von Speicherpfaden in allen Datenbankpartitionen verwenden.
2. Die Ausführung eines Befehls RESTORE mit neuen Speicherpfaden ist nur in der Katalogdatenbankpartition möglich. Sie versetzt die Datenbank in den Status 'Restore anstehend' (RESTORE_PENDING) in allen anderen Datenbankpartitionen (d. h. Nichtkatalogpartitionen).

Tabelle 44. Auswirkungen eines Befehls RESTORE für Mehrpartitionsdatenbanken

Befehl RESTORE DATABASE	Datenbankpartition der Ausführung	Keine Datenbank mit demselben Namen auf der Platte vorhanden		Datenbank mit demselben Namen auf der Platte vorhanden (einschließlich Entwurfsdatenbanken)	
		Ergebnis in anderen Datenbankpartitionen	Speicherpfade	Ergebnis in anderen Datenbankpartitionen	Speicherpfade
RESTORE DATABASE TEST1	Katalogdatenbankpartition	Eine Entwurfsdatenbank wird mit den Speicherpfaden aus dem Backup-Image in der Katalogdatenbankpartition erstellt. Alle anderen Datenbankpartitionen werden in den Status 'Restore anstehend' (RESTORE_PENDING) versetzt.	Die im Backup-Image definierten Speicherpfade werden verwendet.	Keines. Speicherpfade wurden nicht geändert, sodass es zu keinen Auswirkungen auf andere Datenbankpartitionen kommt.	Die im Backup-Image definierten Speicherpfade werden verwendet.
	Nichtkatalogdatenbankpartition	SQL2542N oder SQL2551N wird zurückgegeben. Wenn keine Datenbank vorhanden ist, muss die Katalogdatenbankpartition zuerst mit Restore wiederhergestellt werden.	N/V	Keines. Speicherpfade wurden nicht geändert, sodass es zu keinen Auswirkungen auf andere Datenbankpartitionen kommt.	Die im Backup-Image definierten Speicherpfade werden verwendet.

Tabelle 44. Auswirkungen eines Befehls RESTORE für Mehrpartitionsdatenbanken (Forts.)

Befehl RESTORE DATABASE	Datenbankpartition der Ausführung	Keine Datenbank mit demselben Namen auf der Platte vorhanden		Datenbank mit demselben Namen auf der Platte vorhanden (einschließlich Entwurfsdatenbanken)	
		Ergebnis in anderen Datenbankpartitionen	Speicherpfade	Ergebnis in anderen Datenbankpartitionen	Speicherpfade
`RESTORE DATABASE TEST2 ON /pfad1, /pfad2, /pfad3`	Katalogdatenbankpartition	Eine Entwurfsdatenbank wird mit den im RESTORE-Befehl angegebenen Speicherpfaden erstellt. Alle anderen Datenbankpartitionen werden in den Status 'Restore anstehend' (RESTORE_PENDING) versetzt.	`/pfad1, /pfad2, /pfad3`		`/pfad1, /pfad2, /pfad3`
	Nichtkatalogdatenbankpartition	SQL1174N wird zurückgegeben. Wenn keine Datenbank vorhanden ist, muss die Katalogdatenbankpartition zuerst mit Restore wiederhergestellt werden. Speicherpfade können im RESTORE-Befehl einer Nichtkatalogdatenbank partition nicht angegeben werden.	N/V	SQL1172N wird zurückgegeben. Neue Speicherpfade können im RESTORE-Befehl einer Nichtkatalogdatenbankpartition nicht angegeben werden.	N/V

Katalogisieren von Datenbanken

Wenn Sie eine neue Datenbank erstellen, wird sie automatisch in der Datei für das Systemdatenbankverzeichnis katalogisiert. Sie können außerdem den Befehl CATALOG DATABASE verwenden, um eine Datenbank explizit in der Datei für das Systemdatenbankverzeichnis zu katalogisieren.

Mit dem Befehl CATALOG DATABASE können Sie eine Datenbank mit einem anderen Aliasnamen oder auch einen Datenbankeintrag katalogisieren, der zuvor mit dem Befehl UNCATALOG DATABASE gelöscht wurde.

Obwohl Datenbanken bei ihrer Erstellung automatisch katalogisiert werden, kann es erforderlich werden, eine Datenbank zu katalogisieren. Wenn dieser Fall eintritt, muss die Datenbank vorhanden sein.

Standardmäßig werden Verzeichnisdateien (einschließlich des Datenbankverzeichnisses) im Hauptspeicher zwischengespeichert. Hierzu wird der Konfigurationsparameter für die Verzeichniscacheunterstützung (*dir_cache*) verwendet. Wenn der Verzeichniscache aktiviert ist, wird eine Änderung an einem Verzeichnis (z. B. durch den Befehl CATALOG DATABASE oder UNCATALOG DATABASE) durch eine andere Anwendung möglicherweise erst wirksam, wenn Ihre Anwendung erneut gestartet wird. Um den Verzeichniscache, der von einer Sitzung des Befehlszeilenprozessors verwendet wird, zu aktualisieren, führen Sie den Befehl TERMINATE aus.

In einer partitionierten Datenbank wird in jeder Datenbankpartition ein Cache für Verzeichnisdateien erstellt.

Neben dem Cache auf Anwendungsebene wird auch ein Cache auf Datenbankmanagerebene für interne Suchfunktionen des Datenbankmanagers verwendet. Um diesen „gemeinsamen" Cache zu aktualisieren, führen Sie die Befehle db2stop und db2start aus.

Verwenden Sie den Befehl CATALOG DATABASE über den Befehlszeilenprozessor, um eine Datenbank unter einem anderen Aliasnamen zu katalogisieren. Zum Beispiel wird durch den folgenden Befehl des Befehlszeilenprozessors die Datenbank PERSON1 als HUMANRES katalogisiert:

```
CATALOG DATABASE personl AS humanres
  WITH "Human Resources Database"
```

In diesem Beispiel enthält der Datenbankverzeichniseintrag des Systems den Aliasnamen HUMANRES für die Datenbank, der sich vom Datenbanknamen (PERSON1) unterscheidet.

Zum Katalogisieren einer Datenbank im Systemdatenbankverzeichnis aus einer Clientanwendung heraus rufen Sie die API 'sqlecadb' auf.

Verwenden Sie den Befehl CATALOG DATABASE über den Befehlszeilenprozessor, um eine Datenbank in einer anderen Instanz als der Standardinstanz zu katalogisieren. Im folgenden Beispiel werden Verbindungen zur Datenbank B nun zur Instanz `INSTNC_C` hergestellt. Die Instanz `instnc_c` muss bereits als lokaler Knoten katalogisiert sein, bevor dieser Befehl ausgeführt wird.

```
CATALOG DATABASE b as b_on_ic AT NODE instnc_c
```

Anmerkung: Der Befehl CATALOG DATABASE wird auch auf Clientknoten verwendet, um Datenbanken zu katalogisieren, die sich auf Datenbankservercomputern befinden.

Binden von Dienstprogrammen an die Datenbank

Bei der Erstellung einer Datenbank versucht der Datenbankmanager, die in den Dateien db2ubind.lst und db2cli.lst aufgeführten Dienstprogramme an die Datenbank zu binden. Diese Dateien befinden sich im Unterverzeichnis bnd im entsprechenden Verzeichnis sqllib.

Durch das Binden (BIND) eines Dienstprogramms wird ein *Paket* (d. h., ein Objekt) erstellt, das alle Informationen enthält, die zur Verarbeitung bestimmter SQL- und XQuery-Anweisungen aus einer einzigen Quellendatei benötigt werden.

Anmerkung: Wenn diese Dienstprogramme von einem Client ausgeführt werden sollen, müssen sie explizit gebunden werden. Sie müssen sich in dem Verzeichnis befinden, in dem diese Dateien gespeichert sind, um die Pakete in der Datenbank 'sample' zu erstellen. Die Bindedateien befinden sich im Unterverzeichnis bnd des Verzeichnisses sqllib. Sie müssen auch die Datei db2schema.bnd binden, wenn Sie die Datenbank erstellen oder ein Upgrade der Datenbank von einem Client durchführen. Detaillierte Informationen finden Sie im Abschnitt "DB2-CLI-Bindedateien und Paketnamen".

Rufen Sie über die Befehlszeile die folgenden Befehle auf, um Dienstprogramme an eine Datenbank zu binden bzw. neu zu binden. Hierbei steht `sample` für den Namen der Datenbank:

```
connect to sample
bind @db2ubind.lst
```

Erstellen von Aliasnamen der Datenbank

Ein *Aliasname* ist eine indirekte Methode, auf eine Tabelle, einen Kurznamen oder eine Sicht zu verweisen, sodass eine SQL- oder XQuery-Anweisung vom qualifizierten Namen dieser Tabelle oder Sicht unabhängig sein kann.

Es muss lediglich die Definition des Aliasnamens geändert werden, wenn die Tabelle oder Sicht umbenannt wird. Ein Aliasname kann für andere Aliasnamen erstellt werden. Ein Aliasname kann in einer Sicht- oder Triggerdefinition und in jeder SQL- oder XQuery-Anweisung außer in Definitionen von Prüfungen auf Integritätsbedingungen in Tabellen (Table Check Constraints) verwendet werden, in denen ein vorhandener Tabellen- oder Sichtname angegeben werden kann.

Ein Aliasname kann für eine Tabelle, eine Sicht oder einen Aliasnamen definiert werden, die bzw. der zum Zeitpunkt der Definition noch nicht existiert. Wenn die SQL- oder XQuery-Anweisung, die diesen Aliasnamen enthält, kompiliert wird, muss die Tabelle, Sicht oder der Aliasname vorhanden sein.

Ein Aliasname kann überall dort verwendet werden, wo ein vorhandener Tabellenname verwendet werden kann, und kann auf einen anderen Aliasnamen verweisen, sofern keine rückbezüglichen oder sich wiederholenden Verweise in der Kette der Aliasnamen auftreten.

Als Aliasname kann kein bereits vorhandener Tabellen-, Sicht- oder Aliasname angegeben werden, und der Name darf sich nur auf eine Tabelle in derselben Datenbank beziehen. Der Name einer Tabelle oder Sicht, der in einer Anweisung CREATE TABLE oder CREATE VIEW verwendet wird, darf nicht mit einem Aliasnamen im selben Schema übereinstimmen.

Sie benötigen keine besondere Berechtigung zur Erstellung eines Aliasnamens, sofern der Aliasname sich nicht in einem anderen Schema als dem befindet, das Ihrer aktuellen Berechtigungs-ID eigen ist. In diesem Fall benötigen Sie die Berechtigung DBADM.

Wenn ein Aliasname oder das Objekt, auf das ein Aliasname verweist, gelöscht wird, werden alle von dem Aliasnamen abhängigen Pakete als ungültig und alle von dem Aliasnamen abhängigen Sichten und Trigger als funktionsunfähig markiert.

Geben Sie in der Befehlszeile Folgendes ein, um einen Aliasnamen zu erstellen:

```
CREATE ALIAS <aliasname> FOR <tabellenname>
```

Der Aliasname wird bei der Kompilierung der Anweisung durch den entsprechenden Tabellen- oder Sichtnamen ersetzt. Wenn der Aliasname oder die Aliasnamenskette nicht in einen Tabellen- oder Sichtnamen aufgelöst werden kann, wird ein Fehler zurückgegeben. Wenn zum Beispiel WORKERS ein Aliasname für die Tabelle EMPLOYEE ist, geschieht bei der Kompilierung Folgendes:

```
SELECT * FROM WORKERS
```

wird effektiv zu

```
SELECT * FROM EMPLOYEE
```

Mit der folgenden SQL-Anweisung wird ein Aliasname WORKERS für die Tabelle EMPLOYEE erstellt:

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

Anmerkung: DB2 für OS/390 oder z/Series implementiert zwei verschiedene Konzepte von Aliasnamen: ALIAS und SYNONYM. Diese beiden Konzepte weichen auf folgende Weise von der DB2-Datenbank ab:

- Für ALIASnamen in DB2 für OS/390 oder z/Series gilt:
  - Sie machen eine spezielle Berechtigung oder ein bestimmtes Zugriffsrecht für ihre Ersteller erforderlich.
  - Sie können nicht auf andere Aliasnamen verweisen.
- Für SYNONYMe in DB2 für OS/390 oder z/Series gilt:
  - Sie können nur von ihrem Ersteller verwendet werden.
  - Sie sind immer ohne Qualifikationsmerkmal.
  - Sie werden gelöscht, wenn die Bezugstabelle gelöscht wird.
  - Sie benutzen keinen Namensbereich mit Tabellen oder Sichten gemeinsam.

Herstellen einer Verbindung zu verteilten relationalen Datenbanken

Verteilte relationale Datenbanken basieren auf formalen Requester-Server-Protokollen und -Funktionen.

Ein *Anwendungsrequester* unterstützt die Anwendung, die sich am einen Ende der Verbindung befindet. Er setzt eine Datenbankanforderung der Anwendung in das Format der Kommunikationsprotokolle um, die in einem Netz verteilter Datenbanken verwendet werden können. Diese Anforderungen werden von einem *Datenbankserver* empfangen und verarbeitet, der sich am anderen Endpunkt der Verbindung befindet. Gemeinsam steuern der Anwendungsrequester und der Datenbankserver die Kommunikation und die standortbedingten Faktoren der Verarbeitung, sodass die Anwendung in gleicher Weise arbeiten kann wie beim Zugriff auf eine lokale Datenbank.

Ein Anwendungsprozess muss eine Verbindung zum Anwendungsserver eines Datenbankmanagers herstellen, bevor SQL-Anweisungen ausgeführt werden kön-

nen, die auf Tabellen oder Sichten verweisen. Die Anweisung CONNECT stellt eine Verbindung zwischen einem Anwendungsprozess und dem zugehörigen Server her.

Es gibt zwei Arten von CONNECT-Anweisungen:

- CONNECT (Typ 1) unterstützt die Semantik mit einer Einzeldatenbank pro Arbeitseinheit (RUOW = Remote Unit of Work, ferne Arbeitseinheit).
- CONNECT (Typ 2) unterstützt die Semantik mit mehreren Datenbanken pro Arbeitseinheit (anwendungsgesteuerte DUOW = Application-Directed Distributed Unit of Work, anwendungsgesteuerte verteilte Arbeitseinheit).

DB2 Call Level Interface (CLI) und das eingebettete SQL unterstützen einen Verbindungsmodus, der als Modus für *gleichzeitig ablaufende Transaktionen* bezeichnet wird und der mehrere Verbindungen ermöglicht, die jeweils als unabhängige Transaktion behandelt werden. Eine Anwendung kann über mehrere gleichzeitig bestehende Verbindungen zur selben Datenbank verfügen.

Der Anwendungsserver kann sich relativ zu der Umgebung, in der der Prozess eingeleitet wird, auf einem lokalen oder fernen System befinden. Ein Anwendungsserver ist immer vorhanden. Dies gilt auch dann, wenn die Umgebung keine verteilten relationalen Datenbanken verwendet. Diese Umgebung umfasst ein lokales Verzeichnis, in dem die Anwendungsserver, die in einer Anweisung CONNECT angegeben werden können, beschrieben sind.

Der Anwendungsserver führt die gebundene Form einer statischen SQL-Anweisung aus, mit der auf Tabellen oder Sichten verwiesen wird. Die gebundene Anweisung wird aus einem Paket entnommen, das der Datenbankmanager zuvor über eine Bindeoperation erstellt hat.

In den meisten Fällen kann eine an einen Anwendungsserver angeschlossene Anwendung Anweisungen und Klauseln verwenden, die vom Datenbankmanager des Anwendungsservers unterstützt werden. Dies gilt auch dann, wenn eine Anwendung über den Anwendungsrequester eines Datenbankmanagers ausgeführt wird, der einige dieser Anweisungen und Klauseln *nicht* unterstützt.

Ferne Arbeitseinheit für verteilte relationale Datenbanken

Die *Funktion für ferne Arbeitseinheiten* (RUOW = Remote Unit of Work) ermöglicht die Vorbereitung und Ausführung von SQL-Anweisungen über eine ferne Einheit.

Ein Anwendungsprozess auf Computersystem „A" kann eine Verbindung zu einem Anwendungsserver auf Computersystem „B" herstellen und innerhalb einer oder mehrerer Arbeitseinheiten (UOWs) eine beliebige Anzahl von statischen oder dynamischen SQL-Anweisungen ausführen, die auf Objekte auf Computersystem „B" zugreifen. Nach Beendigung einer Arbeitseinheit auf Computersystem B kann der Anwendungsprozess eine Verbindung zu einem Anwendungsserver auf Computersystem C herstellen, usw.

Die meisten SQL-Anweisungen können remote vorbereitet und ausgeführt werden. Hierbei gelten allerdings die folgenden Einschränkungen:

- Alle Objekte, auf die in einer einzelnen SQL-Anweisung verwiesen wird, müssen vom selben Anwendungsserver verwaltet werden.
- Alle SQL-Anweisungen in einer Arbeitseinheit müssen vom selben Anwendungsserver ausgeführt werden.

Zu einem bestimmten Zeitpunkt befindet sich ein Anwendungsprozess in einem der vier möglichen *Verbindungsstatus*:

- Verbindung möglich und verbunden

  Ein Anwendungsprozess ist mit einem Anwendungsserver verbunden und CONNECT-Anweisungen können ausgeführt werden.

  Wenn die implizite Verbindung möglich ist:

  – Der Anwendungsprozess nimmt diesen Status an, wenn eine Anweisung CONNECT TO oder CONNECT ohne Operanden im Status 'Verbindung möglich und nicht verbunden' erfolgreich ausgeführt wird.

  – Der Anwendungsprozess kann aus dem Status 'Implizite Verbindung möglich' in diesen Status versetzt werden, wenn eine beliebige SQL-Anweisung mit Ausnahme von CONNECT RESET, DISCONNECT, SET CONNECTION oder RELEASE eingegeben wird.

  Unabhängig davon, ob die implizite Verbindung möglich ist, wird dieser Status angenommen, wenn Folgendes gilt:

  – Eine Anweisung CONNECT TO wird im Status 'Verbindung möglich und nicht verbunden' erfolgreich ausgeführt.

  – Eine Anweisung COMMIT oder ROLLBACK wird erfolgreich abgesetzt oder im Status 'Verbindung nicht möglich und verbunden' wird ein erzwungener Rollback ausgeführt.

- Verbindung nicht möglich und verbunden

  Ein Anwendungsprozess ist mit einem Anwendungsserver verbunden, eine Anweisung CONNECT TO kann jedoch nicht erfolgreich ausgeführt werden, um die Anwendungsserver zu ändern. Der Anwendungsprozess kann aus dem Status 'Verbindung möglich und verbunden' in diesen Status versetzt werden, wenn eine beliebige SQL-Anweisung mit Ausnahme von CONNECT TO, CONNECT ohne Operand, CONNECT RESET, DISCONNECT, SET CONNECTION, RELEASE, COMMIT oder ROLLBACK ausgeführt wird.

- Verbindung möglich und nicht verbunden

  Ein Anwendungsprozess ist nicht mit einem Anwendungsserver verbunden. CONNECT TO ist die einzige SQL-Anweisung, die ausgeführt werden kann. Andernfalls wird ein Fehler (SQLSTATE 08003) ausgegeben.

  Unabhängig davon, ob eine implizite Verbindung möglich ist, wird der Anwendungsprozess in diesen Status versetzt, wenn bei der Eingabe einer Anweisung CONNECT TO ein Fehler auftritt oder wenn in einer Arbeitseinheit ein Fehler auftritt, der zu einer Verbindungsunterbrechung und einem Rollback führt. Ein Fehler, der darauf zurückzuführen ist, dass der Anwendungsprozess sich nicht im Status 'Verbindung möglich' befindet, oder dass der Servername nicht im lokalen Verzeichnis aufgeführt ist, verursacht keine Änderung in diesen Status.

  Wenn die implizite Verbindung nicht möglich ist, gilt Folgendes:

  – Der Anwendungsprozess befindet sich zu Beginn in diesem Status.

  – Die Anweisungen CONNECT RESET und DISCONNECT bewirken eine Änderung in diesen Status.

- Implizite Verbindung möglich (wenn die implizite Verbindung verfügbar ist).
  Wenn die implizite Verbindung verfügbar ist, stellt dieser Status den Anfangsstatus eines Anwendungsprozesses dar. Die Anweisung CONNECT RESET bewirkt eine Änderung in diesen Status. Dieser Status wird auch angenommen, wenn eine Anweisung COMMIT oder ROLLBACK eingegeben wird, während sich das System im Status 'Verbindung nicht möglich und verbunden' befindet, und anschließend im Status 'Verbindung möglich und verbunden' eine Anweisung DISCONNECT eingegeben wird.

Die Verfügbarkeit der impliziten Verbindung wird mithilfe von Installationsoptionen, Umgebungsvariablen und Authentifizierungseinstellungen festgelegt.

Die Ausführung aufeinanderfolgender CONNECT-Anweisungen stellt keinen Fehler dar, weil CONNECT selbst den Anwendungsprozess nicht aus dem Status 'Verbindung möglich' in einen anderen Status versetzt. Die Ausführung aufeinanderfolgender CONNECT RESET-Anweisungen stellt jedoch einen Fehler dar. Außerdem ist es auch ein Fehler, eine andere SQL-Anweisung als CONNECT TO, CONNECT RESET, CONNECT ohne Operand, SET CONNECTION, RELEASE, COMMIT oder ROLLBACK und anschließend eine Anweisung CONNECT TO auszuführen. Um diesen Fehler zu vermeiden, muss vor der Anweisung CONNECT TO eine Anweisung CONNECT RESET, DISCONNECT (mit vorausgehender Anweisung COMMIT oder ROLLBACK), COMMIT oder ROLLBACK ausgeführt werden.

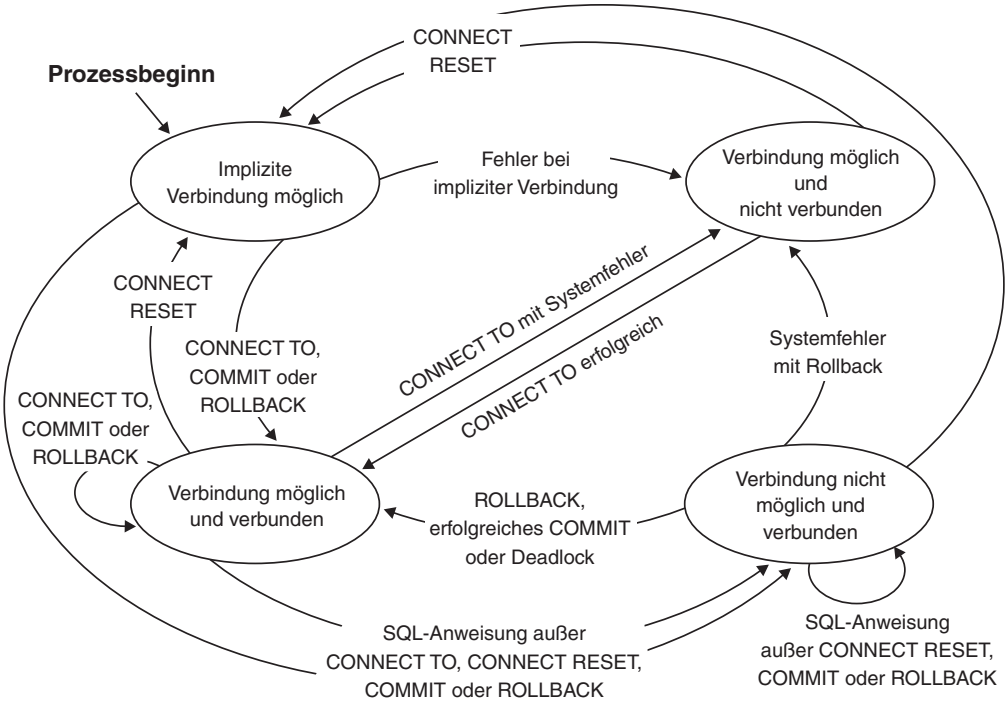


Abbildung 3. Verbindungsstatusänderungen bei Verfügbarkeit einer impliziten Verbindung

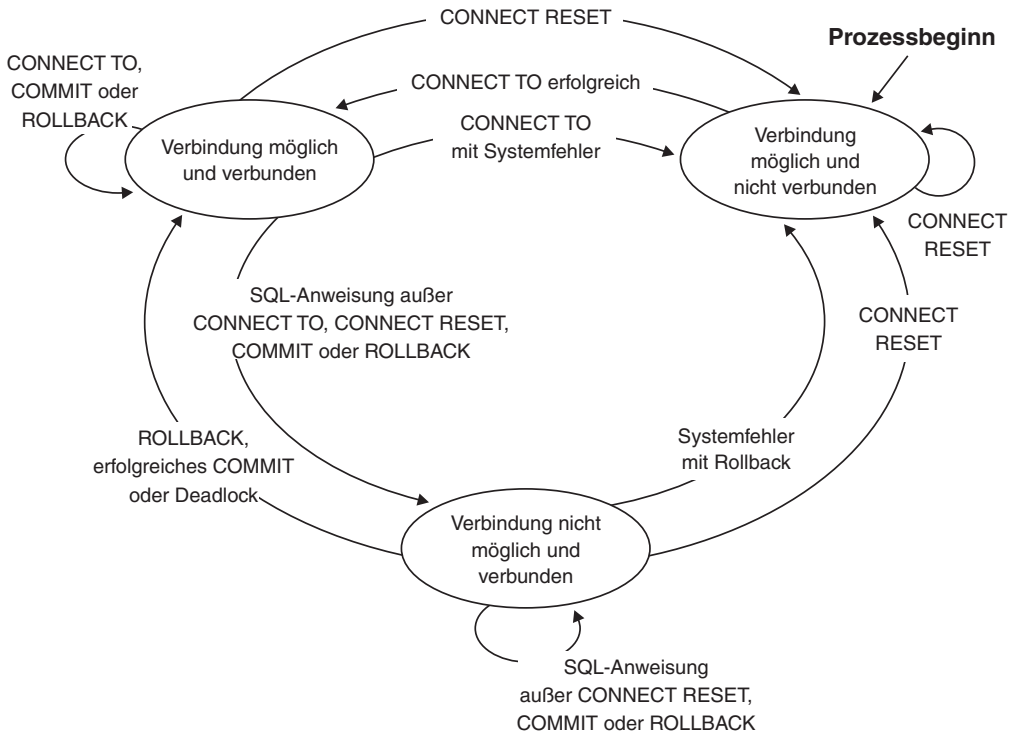


Abbildung 4. Verbindungsstatusänderungen bei Nichtverfügbarkeit einer impliziten Verbindung

Anwendungsgesteuerte verteilte Arbeitseinheit

Die *Funktion für anwendungsgesteuerte verteilte Arbeitseinheiten* (DUOW = Distributed Unit of Work) ermöglicht die ferne Vorbereitung und Ausführung von SQL-Anweisungen.

Ein Anwendungsprozess auf dem Computersystem „A" kann eine Verbindung zu einem Anwendungsserver auf Computersystem „B" herstellen, indem er eine Anweisung CONNECT oder SET CONNECTION absetzt. Der Anwendungsprozess kann dann eine beliebige Anzahl statischer und dynamischer SQL-Anweisungen ausführen, die auf Objekte auf dem Computersystem „B" zugreifen, bevor er die UOW beendet. Alle Objekte, auf die in einer einzelnen SQL-Anweisung verwiesen wird, müssen vom selben Anwendungsserver verwaltet werden. Anders als bei der Funktion für ferne Arbeitseinheiten (RUOWs) kann eine beliebige Anzahl von Anwendungsservern an derselben UOW beteiligt werden. Eine Commit- oder Rollbackoperation beendet die UOW.

Eine anwendungsgesteuerte DUOW verwendet eine Verbindung des Typs 2. Eine Verbindung vom *Typ 2* verbindet einen Anwendungsprozess mit dem angegebenen Anwendungsserver und legt die Regeln für anwendungsgesteuerte DUOWs fest.

Für einen Typ 2-Anwendungsprozess gilt Folgendes:

- Der Anwendungsprozess weist immer den Status 'Verbindung möglich' auf.
- Der Anwendungsprozess befindet sich entweder im Status 'Verbunden' oder im Status 'Nicht verbunden'.
- Der Anwendungsprozess verfügt über null oder mehr Verbindungen.

Jede Verbindung eines Anwendungsprozesses wird für die Verbindung durch den Aliasnamen der Datenbank des Anwendungsservers eindeutig identifiziert.

Eine einzelne Verbindung weist immer einen der folgenden Verbindungsstatus auf:

- Aktuell und angehalten
- Aktuell und Freigabe anstehend
- Ruhend und angehalten
- Ruhend und Freigabe anstehend

Ein Anwendungsprozess des Typs 2 befindet sich zu Beginn im Status 'Nicht verbunden' und verfügt nicht über Verbindungen. Eine Verbindung befindet sich zu Beginn im Status 'Aktuell und angehalten'.

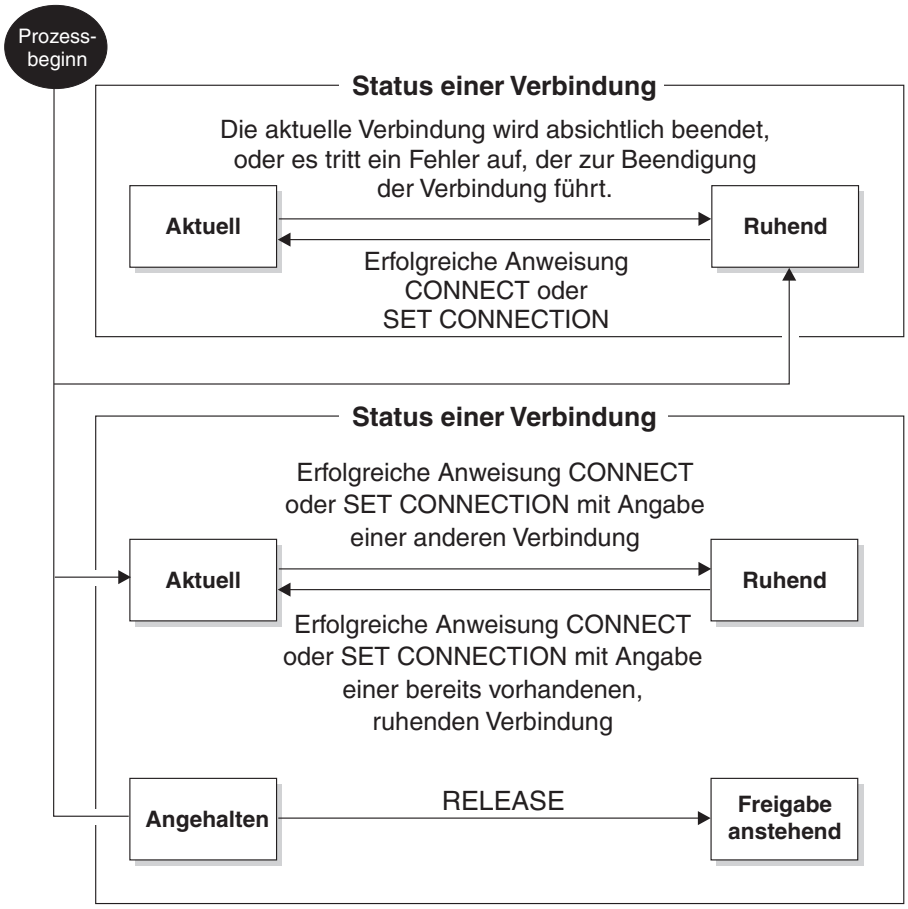


Abbildung 5. Verbindungsstatusänderungen bei anwendungsgesteuerten verteilten DUOWs

Verbindungsstatus von Anwendungsprozessen

Für die Ausführung einer Anweisung CONNECT gelten bestimmte Regeln.

Diese Regeln lauten wie folgt:

- Ein Kontext kann nicht gleichzeitig über mehrere Verbindungen zum selben Anwendungsserver verfügen.
- Wenn ein Anwendungsprozess eine Anweisung SET CONNECTION ausführt, muss als Positionsname eine bereits vorhandene Verbindung aus der Gruppe der Verbindungen für den Anwendungsprozess angegeben werden.

- Wenn ein Anwendungsprozess eine Anweisung CONNECT ausführt und die Option SQLRULES(STD) aktiviert wurde, darf der angegebene Servername *nicht* mit der Angabe für eine bereits vorhandene Verbindung aus der Gruppe der Verbindungen für den Anwendungsprozess übereinstimmen. Eine Beschreibung der Option SQLRULES finden Sie unter „Optionen zur Regelung der Semantik von Arbeitseinheiten" auf Seite 154.

Wenn ein Anwendungsprozess über eine aktuelle Verbindung verfügt, befindet sich der Anwendungsprozess im Status *Verbunden*. Das Sonderregister CURRENT SERVER enthält den Namen des Anwendungsservers für die aktuelle Verbindung. Der Anwendungsprozess kann SQL-Anweisungen ausführen, die auf Objekte verweisen, die von dem Anwendungsserver verwaltet werden.

Ein Anwendungsprozess, der sich im Status 'Nicht verbunden' befindet, wird in den Status 'Verbunden' überführt, wenn eine Anweisung CONNECT oder SET CONNECTION erfolgreich ausgeführt werden konnte. Wenn keine Verbindung vorhanden ist, jedoch SQL-Anweisungen ausgegeben werden, wird eine implizite Verbindung hergestellt. Dies gilt allerdings nur, wenn in der Umgebungsvariablen DB2DBDFT der Name einer Standarddatenbank angegeben wurde.

Wenn ein Anwendungsprozess über keine aktuelle Verbindung verfügt, befindet sich der Anwendungsprozess im Status *Nicht verbunden*. Als einzige SQL-Anweisungen können die Anweisungen CONNECT, DISCONNECT ALL, DISCONNECT (Angabe einer Datenbank), SET CONNECTION, RELEASE, COMMIT, ROLLBACK sowie lokale SET-Anweisungen ausgeführt werden.

Ein Anwendungsprozess im Status *Verbunden* wird in den Status *Nicht verbunden* überführt, wenn die momentan vorhandene, aktuelle Verbindung absichtlich beendet wird oder wenn eine SQL-Anweisung fehlschlägt und zu einer Rollbackoperation auf dem Anwendungsserver und dem Verlust der Verbindung führt. Verbindungen werden nach der erfolgreichen Ausführung einer Anweisung DISCONNECT oder nach der Ausführung einer Anweisung COMMIT absichtlich beendet, wenn die Verbindung sich im Status 'Freigabe anstehend' befindet. (Wenn die Precompileroption DISCONNECT auf den Wert AUTOMATIC eingestellt ist, werden alle Verbindungen beendet. Wenn sie auf den Wert CONDITIONAL eingestellt ist, werden hingegen alle Verbindungen beendet, die nicht über geöffnete WITH HOLD-Cursor verfügen.)

Verbindungsstatus

Es gibt zwei Typen von Verbindungsstatus: Die Status „Angehalten und Freigabe anstehend" und die Status „Aktuell und Ruhend".

Wenn ein Anwendungsprozess eine Anweisung CONNECT ausführt und der Servername für den Anwendungsrequester definiert wurde, sich jedoch nicht in der Gruppe der vorhandenen Verbindungen für den Anwendungsprozess befindet, gilt Folgendes: (i) Die aktuelle Verbindung wird in den Verbindungsstatus *Ruhend* versetzt, der Servername wird zur Gruppe der Verbindungen hinzugefügt und die neue Verbindung wird sowohl in den Verbindungsstatus *Aktuell* als auch in den Verbindungsstatus *Angehalten* versetzt.

Wenn der Servername sich bereits in der Gruppe der vorhandenen Verbindungen für den Anwendungsprozess befindet und die Anwendung mit der Option SQLRULES(STD) vorkompiliert wurde, dann wird ein Fehler (SQLSTATE 08002) ausgegeben.

Status 'Angehalten' und 'Freigabe anstehend'. Mit der Anweisung RELEASE kann angegeben werden, ob eine Verbindung sich im Status 'Angehalten' oder 'Freigabe anstehend' befindet. Der Status *Freigabe anstehend* bedeutet, dass nach der nächsten erfolgreichen Ausführung einer Commitoperation eine Verbindungsunterbrechung durchgeführt wird. (Eine Rollbackoperation hat keine Auswirkungen auf die Verbindungen.) Der Status *Angehalten* bedeutet, dass nach der nächsten Ausführung einer Commitoperation *keine* Verbindungsunterbrechung durchgeführt wird.

Alle Verbindungen befinden sich zu Beginn im Status 'Angehalten' und können mit der Anweisung RELEASE in den Status 'Freigabe anstehend' versetzt werden. Nachdem sie sich im Status 'Freigabe anstehend' befindet, kann eine Verbindung nicht wieder zurück in den Status 'Angehalten' versetzt werden. Eine Verbindung bleibt über Arbeitseinheiten hinweg im Status 'Freigabe anstehend', wenn eine Anweisung ROLLBACK eingegeben wird oder wenn eine fehlgeschlagene Commitoperation zu einer ROLLBACK-Operation führt.

Selbst wenn eine Verbindung nicht explizit für die Freigabe markiert ist, kann sie trotzdem aufgrund einer Commitoperation unterbrochen werden, wenn diese die Bedingungen der Precompileroption DISCONNECT erfüllt.

Status 'Aktuell' und 'Ruhend'. Unabhängig davon, ob eine Verbindung sich im Status 'Angehalten' oder im Status 'Freigabe anstehend' befindet, kann sie sich außerdem im Status 'Aktuell' oder im Status 'Ruhend' befinden. Eine Verbindung im Status *Aktuell* wird zur Ausführung von SQL-Anweisungen verwendet, solange sich dieser Status nicht ändert. Eine Verbindung im Status *Ruhend* ist eine Verbindung, die momentan nicht aktuell ist.

Die einzigen SQL-Anweisungen, die über eine ruhende Verbindung ausgeführt werden können, sind COMMIT, ROLLBACK, DISCONNECT und RELEASE. Die Anweisungen SET CONNECTION und CONNECT ändern den Verbindungsstatus des angegebenen Servers in 'Aktuell'. Alle vorhandenen Verbindungen werden in den Status 'Ruhend' versetzt oder bleiben weiterhin in diesem Status. Zu einem bestimmten Zeitpunkt kann sich immer nur eine Verbindung im Status 'Aktuell' befinden. Wenn eine ruhende Verbindung innerhalb derselben Arbeitseinheit aktiviert wird, dann entspricht der Status aller Sperren, Cursor und vorbereiteten Anweisungen dem Status, in dem sich diese befanden, als die Verbindung zum letzten Mal den Status 'Aktuell' aufwies.

Beendigung einer Verbindung

Wenn eine Verbindung beendet wird, werden alle Ressourcen, die vom Anwendungsprozess über die Verbindung belegt wurden, und alle Ressourcen, die zum Herstellen und Verwalten der Verbindung verwendet wurden, freigegeben. Wenn beispielsweise der Anwendungsprozess eine Anweisung RELEASE ausführt, werden alle geöffneten Cursor geschlossen, wenn die Verbindung während der nächsten Commitoperation beendet wird.

Eine Verbindung kann auch aufgrund eines Kommunikationsfehlers beendet werden. Wenn diese Verbindung sich im Status 'Aktuell' befindet, dann wird der Anwendungsprozess in den Status 'Nicht verbunden' versetzt.

Alle Verbindungen für einen Anwendungsprozess werden beendet, wenn der Prozess beendet wird.

Optionen zur Regelung der Semantik von Arbeitseinheiten

Die Semantik für die Verwaltung von Verbindungen vom Typ 2 wird von einer Reihe von Precompileroptionen bestimmt. Diese Optionen werden im Folgenden zusammengefasst. Die jeweiligen Standardwerte sind in Fettdruck hervorgehoben und unterstrichen.

- CONNECT (**1** | 2). Gibt an, ob CONNECT-Anweisungen als Anweisungen des Typs 1 oder Typs 2 verarbeitet werden sollen.
- SQLRULES (**DB2** | STD). Gibt an, ob CONNECT-Anweisungen des Typs 2 auf der Basis der DB2-Regeln verarbeitet werden sollen, die für CONNECT eine Umschaltung zu einer ruhenden Verbindung zulassen, oder ob die Regeln des SQL92-Standards verwendet werden sollen, die diese Vorgehensweise nicht zulassen.
- DISCONNECT (**EXPLICIT** | CONDITIONAL | AUTOMATIC). Gibt an, welche Datenbankverbindungen unterbrochen werden sollen, wenn eine Commitoperation ausgeführt wird:
  - Verbindungen, die mit der SQL-Anweisung RELEASE (EXPLICIT) explizit zur Freigabe markiert wurden.
  - Verbindungen, die nicht über geöffnete WITH HOLD-Cursor verfügen und Verbindungen, die für die Freigabe (CONDITIONAL) markiert sind.
  - Alle Verbindungen (AUTOMATIC).
- SYNCPOINT (**ONEPHASE** | TWOPHASE | NONE). Gibt an, wie COMMIT- und ROLLBACK-Operationen zwischen mehreren Datenbankverbindungen koordiniert werden sollen. Diese Option wird ignoriert und dient nur zur Gewährleistung der Abwärtskompatibilität.
  - Aktualisierungen können in der Arbeitseinheit nur in einer Datenbank durchgeführt werden. Alle anderen Datenbanken sind schreibgeschützt (ONEPHASE). Alle Aktualisierungsversuche in anderen Datenbanken führen zu einem Fehler (SQLSTATE 25000).
  - Zur Laufzeit wird ein Transaktionsmanager (TM) verwendet, um zweiphasige Commitoperationen in den Datenbanken zu koordinieren, die dieses Protokoll (TWOPHASE) unterstützen.
  - Zur Ausführung des zweiphasigen Commits wird kein Transaktionsmanager (TM) verwendet und die Verwendung einer einzelnen Aktualisierungskomponente und mehrerer Eingabekomponenten wird nicht erzwungen (NONE). Wenn eine COMMIT- oder ROLLBACK-Operation ausgeführt wird, werden für alle Datenbanken individuelle COMMIT- oder ROLLBACK-Operationen übergeben. Wenn mindestens eine der ROLLBACK-Operationen fehlschlägt, wird ein Fehler (SQLSTATE 58005) ausgegeben. Wenn mindestens eine COMMIT-Operation fehlschlägt, wird ein weiterer Fehler (SQLSTATE 40003) ausgegeben.

Um während der Laufzeit eine der hier aufgeführten Optionen außer Kraft zu setzen, können Sie den Befehl SET CLIENT oder die Anwendungsprogrammierschnittstelle (API) sqlesetc verwenden. Ihre aktuellen Einstellungen können mit dem Befehl QUERY CLIENT oder der API sqleqryc abgerufen werden. Beachten Sie hierbei, dass es sich dabei nicht um SQL-Anweisungen, sondern um APIs handelt, die in verschiedenen Hostprogrammiersprachen und im Befehlszeilenprozessor (CLP) definiert wurden.

Wichtige Hinweise zur Datendarstellung

In verschiedenen Systemen werden Daten auf unterschiedliche Weise dargestellt. Wenn Daten von einem System auf ein anderes verschoben werden, muss in bestimmten Fällen eine Datenkonvertierung durchgeführt werden.

Produkte mit DRDA-Unterstützung führen erforderliche Konvertierungsoperationen auf dem Empfangssystem automatisch durch.

Zum Konvertieren numerischer Daten muss das System über Angaben zum Datentyp und dazu verfügen, wie die Daten vom sendenden System dargestellt werden. Zum Konvertieren von Zeichenfolgen sind weitere Informationen erforderlich. Die Zeichenfolgekonvertierung hängt von der verwendeten Codepage der Daten und von der Operation ab, die für die Daten ausgeführt werden soll. Zeichenkonvertierungen werden auf der Basis von IBM Character Data Representation Architecture (CDRA) durchgeführt. Weitere Informationen zur Zeichenkonvertierung finden Sie im Handbuch *Character Data Representation Architecture: Reference & Registry* (IBM Form SC09-2190-00).

Anzeigen des Inhalts der Datei des lokalen oder des Systemdatenbankverzeichnisses

Mit dem Befehl LIST DATABASE DIRECTORY können Sie die Informationen zu den Datenbanken anzeigen, die sich auf Ihrem System befinden.

Sie können den Inhalt der Dateien des lokalen bzw. des Systemdatenbankverzeichnisses erst anzeigen, wenn Sie eine Instanz und eine Datenbank erstellt haben.

Wenn Sie den Inhalt der Datei des lokalen Datenbankverzeichnisses anzeigen wollen, geben Sie den folgenden Befehl ein, in dem `<position>` die Speicherposition der Datenbank bezeichnet:

```
LIST DATABASE DIRECTORY ON <position>
```

Wenn Sie den Inhalt der Datei des Systemdatenbankverzeichnisses anzeigen wollen, setzen Sie den Befehl LIST DATABASE DIRECTORY *ohne* Angabe der Speicherposition der Datei für das Datenbankverzeichnis ab.

Löschen von Datenbanken

Das Löschen einer Datenbank kann weit reichende Auswirkungen haben, da alle Objekte der Datenbank, Container und zugehörige Dateien gelöscht werden. Die gelöschte Datenbank wird aus den Datenbankverzeichnissen entfernt (entkatalogisiert).

Geben Sie in der Befehlszeile Folgendes ein, um eine Datenbank zu löschen:

```
DROP DATABASE <name>
```

Mit dem folgenden Befehl wird die Beispieldatenbank SAMPLE gelöscht:

```
DROP DATABASE SAMPLE
```

Anmerkung: Wenn Sie die Beispieldatenbank SAMPLE löschen, sie jedoch erneut benötigen, können Sie sie erneut erstellen.

Zum Löschen einer Datenbank aus einer Clientanwendung heraus rufen Sie die API 'sqledrpd' auf. Zum Löschen einer Datenbank auf einem angegeben Datenbankpartitionsserver rufen Sie die API 'sqledpan' auf.

Löschen von Aliasnamen

Wenn Sie einen Aliasnamen löschen, wird seine Beschreibung aus dem Katalog gelöscht, alle vorhandenen Pakete und zwischengespeicherten dynamischen Abfragen, die auf den betreffenden Aliasnamen verweisen, werden ungültig, und alle Sichten und Trigger, die von diesem Aliasnamen abhängig sind, werden als funktionsunfähig markiert.

Geben Sie zum Löschen von Aliasnamen über die Befehlszeile die Anweisung DROP ein:

```
DROP ALIAS EMPLOYEE-ALIAS
```

Kapitel 7. Datenbankpartitionen

Eine *Datenbankpartition* ist ein Teil einer Datenbank, der aus eigenen Daten, Indizes, Konfigurationsdateien und Transaktionsprotokollen besteht. Eine Datenbankpartition wird manchmal auch als Knoten oder Datenbankknoten bezeichnet. Eine Umgebung mit partitionierten Datenbanken ist eine Datenbankinstallation, die die Verteilung von Daten für Datenbankpartitionen unterstützt.

Vollständige Informationen zu Datenbankpartitionen finden Sie im Handbuch *Partitionierung und Clustering*.

Kapitel 8. Pufferpools

Ein *Pufferpool* ist ein Bereich im Hauptspeicher, der dem Datenbankmanager zum Zwischenspeichern von Tabellen- und Indexdaten beim Lesen von Daten vom Datenträger zugeordnet wurde. Jede DB2-Datenbank muss einen Pufferpool haben.

Jede neue Datenbank hat einen definierten Standardpufferpool mit dem Namen IBMDEFAULTBP. Zusätzliche Pufferpools können mithilfe der Anweisungen CREATE BUFFERPOOL, DROP BUFFERPOOL und ALTER BUFFERPOOL erstellt, gelöscht und geändert werden. Die Katalogsicht SYSCAT.BUFFERPOOLS greift auf die Informationen zu den Pufferpools zu, die in der Datenbank definiert sind.

Verwendung von Pufferpools

Wenn auf eine Zeile von Daten in einer Tabelle zum ersten Mal zugegriffen wird, liest der Datenbankmanager die Seite, die diese Daten enthält, in einen Pufferpool ein. Seiten verbleiben im Pufferpool, bis die Datenbank gestoppt wird oder der von einer Seite belegte Speicherbereich im Pufferpool von einer anderen Seite benötigt wird.

Seiten im Pufferpool können entweder im Gebrauch sein oder nicht, und sie können genutzt (geändert) oder sauber (ungeändert) sein:

- Seiten, die im Gebrauch sind, werden momentan gelesen oder aktualisiert. Zur Erhaltung der Datenkonsistenz lässt der Datenbankmanager nur einen Agenten zu, der eine bestimmte Seite in einem Pufferpool zu einer bestimmten Zeit aktualisiert. Wenn eine Seite aktualisiert wird, wird auf sie ausschließlich durch einen einzigen Agenten zugegriffen. Wenn sie gelesen wird, kann sie von mehreren Agenten gleichzeitig gelesen werden.
- Genutzte Seiten enthalten Daten, die geändert, jedoch noch nicht auf die Platte geschrieben wurden.
- Wenn eine Seite auf die Platte geschrieben wurde, ist sie sauber und kann im Pufferpool bleiben.

Ein großer Teil der Optimierung einer Datenbank betrifft die Einstellung von Konfigurationsparametern, die das Einlesen von Daten in den Pufferpool und das Schreiben von Daten aus dem Pufferpool auf die Platte steuern. Wenn eine Seite nicht von einem zuletzt aktiven Agenten benötigt wurde, kann sie für neue Seitenanforderungen aus neuen Anwendungen verwendet werden. Die Leistung des Datenbankmanagers kann durch zusätzliche Platten-E/A-Operationen beeinträchtigt werden.

Mithilfe des Snapshot Monitors können Sie die Effektivität der Zugriffe auf Pufferpools berechnen, um die Verwendung Ihrer Pufferpools zu optimieren.

Entwerfen von Pufferpools

Die Größen aller Pufferpools können eine große Auswirkung auf die Leistung Ihrer Datenbank haben.

Vor der Erstellung eines neuen Pufferpools müssen die folgenden Punkte geklärt werden:

- Welcher Pufferpoolname verwendet werden soll.

- Ob der Pufferpool sofort oder erst im Anschluss an die nächste Inaktivierung und erneute Aktivierung der Datenbank erstellt werden soll.
- Ob der Pufferpool für alle Datenbankpartitionen oder nur für eine Untergruppe der Datenbankpartitionen vorhanden sein soll.
- Welche Seitengröße für den Pufferpool verwendet werden soll. Siehe „Seitengrößen von Pufferpools" unten.
- Ob der Pufferpool eine feste Größe haben soll oder ob der Datenbankmanager die Größe des Pufferpools als Reaktion auf Ihre Auslastung automatisch anpassen soll. Es wird empfohlen, den Pufferpool automatisch vom Datenbankmanager optimieren zu lassen, indem Sie den Parameter SIZE bei der Pufferpoolerstellung nicht angeben. Detaillierte Informationen finden Sie in der Beschreibung der Anweisung „CREATE BUFFERPOOL" sowie in „Hinweise zum Pufferpoolspeicher" auf Seite 161.
- Ob ein Teil des Pufferpools für die blockbasierte E/A reserviert werden soll. Detaillierte Informationen finden Sie in „Blockbasierte Pufferpools für besseren sequenziellen Vorablesezugriff".

Beziehung zwischen Tabellenbereichen und Pufferpools

Beim Entwerfen von Pufferpools müssen Sie mit der Beziehung zwischen Tabellenbereichen und Pufferpools vertraut sein. Jeder Tabellenbereich wird einem bestimmten Pufferpool zugeordnet. IBMDEFAULTBP ist der Standardpufferpool. Der Datenbankmanager ordnet darüber hinaus auch die folgenden Systempufferpools zu: IBMSYSTEMBP4K, IBMSYSTEMBP8K, IBMSYSTEMBP16K und IBMSYSTEMBP32K (bisher als „verdeckte Pufferpools" bezeichnet). Wenn ein anderer Pufferpool einem Tabellenbereich zugeordnet werden soll, muss der Pufferpool existieren und die gleiche Seitengröße aufweisen. Die Zuordnung wird bei der Erstellung des Tabellenbereichs (mit der Anweisung CREATE TABLESPACE) definiert. Zu einem späteren Zeitpunkt kann sie (mit der Anweisung ALTER TABLESPACE) geändert werden.

Durch die Verwendung mehrerer Pufferpools haben Sie die Möglichkeit, die Verwendung von Hauptspeicher durch die Datenbank zu konfigurieren, um die allgemeine Leistung zu verbessern. Zum Beispiel kann bei Tabellenbereichen mit einer oder mehreren umfangreichen Tabellen (d. h. größeren als der verfügbare Hauptspeicher), auf die die Benutzer wahlfrei zugreifen, die Größe des Pufferpools begrenzt werden, da ein Zwischenspeichern (Caching) der Datenseiten vielleicht nicht vorteilhaft ist. Dem Tabellenbereich für eine Online-Transaktionsanwendung kann ein größerer Pufferpool zugeordnet werden, sodass die Datenseiten, die von der Anwendung genutzt werden, länger im Cache zwischengespeichert und so schnellere Antwortzeiten erzielt werden können. Bei der Konfiguration neuer Pufferpools ist jedoch ein sorgfältiges Vorgehen geboten.

Seitengrößen von Pufferpools

Die Seitengröße für den Standardpufferpool wird bei der Verwendung des Befehls CREATE DATABASE definiert. Dieser Standardwert stellt die Standardseitengröße für alle zukünftigen Anweisungen CREATE BUFFERPOOL und CREATE TABLESPACE dar. Wenn Sie die Seitengröße beim Erstellen der Datenbank nicht angeben, wird die Standardseitengröße von 4 KB verwendet.

Anmerkung: Wenn Sie ermittelt haben, dass für Ihre Datenbank eine Seitengröße von 8 KB, 16 KB oder 32 KB erforderlich ist, müssen Sie mindestens einen Pufferpool mit der entsprechenden Seitengröße definiert und dem Tabellenbereich in Ihrer Datenbank zugeordnet haben.

Möglicherweise benötigen Sie jedoch einen Pufferpool, dessen Merkmale vom Systempufferpool abweichen. Sie können neue Pufferpools erstellen, die der Datenbankmanager verwendet. Sie müssen die Datenbank möglicherweise erneut starten, um die Änderungen an Tabellenbereichen und Pufferpools in Kraft zu setzen. Die Seitengrößen, die Sie für die Tabellenbereiche angeben, sollten die Seitengrößen bestimmen, die Sie für die Pufferpools auswählen. Die Auswahl der Seitengröße, die für einen Pufferpool angegeben wird, ist wichtig, da die Seitengröße nach der Erstellung des Pufferpools nicht mehr geändert werden kann.

Hinweise zum Pufferpoolspeicher

Speicherbedarf

Beim Entwerfen von Pufferpools müssen Sie auch den Speicherbedarf im Hinblick auf die auf Ihrem System installierte Speicherkapazität und im Verhältnis zu dem Speicher, der von anderen Anwendungen, die gleichzeitig mit dem Datenbankmanager auf demselben System ausgeführt werden, berücksichtigen. Datenauslagerungen (Swapping) durch das Betriebssystem treten auf, wenn nicht ausreichend Speicher zur Aufnahme aller Daten verfügbar ist, auf die zugegriffen wird. Dabei werden einige Daten in einen temporären Plattenspeicherbereich geschrieben, um im Hauptspeicher Platz für andere Daten zu schaffen. Wenn die Daten in dem temporären Plattenspeicherbereich benötigt werden, werden sie wieder zurück in den Hauptspeicher geladen.

Zugriffsschutz für Pufferpoolspeicher

In Version 9.5 werden Datenseiten im Pufferpoolspeicher mithilfe von Speicherschlüsseln geschützt. Diese Speicherschlüssel sind nur verfügbar, wenn sie durch die Registrierdatenbankvariable DB2_MEMORY_PROTECT explizit aktiviert werden. Dies gilt nur für das Betriebssystem AIX (5.3 TL06 5.4) bei Einsatz auf einem POWER6-Prozessor.

Pufferpoolspeicherschutz arbeitet auf Agentenbasis. Ein bestimmter Agent erhält nur dann Zugriff auf die Pufferpoolseiten, wenn dieser Agent den Zugriff benötigt. Der Speicherschutz funktioniert in der Weise, dass bestimmt wird, zu welchen Zeiten die Threads der DB2-Steuerkomponente Zugriff auf den Pufferpoolspeicher haben sollen und zu welchen sie keinen Zugriff haben sollen. Detaillierte Informationen finden Sie in „Zugriffsschutz für Pufferpoolspeicher (AIX auf POWER6)".

Address Windowing Extensions (AWE) und Extended Storage (ESTORE)

Anmerkung: Die AWE- und ESTORE-Komponenten, einschließlich der ESTORE-bezogenen Schlüsselwörter, Monitorelemente und Datenstrukturen, werden nicht mehr unterstützt. Wenn Sie über mehr Speicher verfügen wollen, müssen Sie auf ein 64-Bit-Hardwarebetriebssystem mit den entsprechenden DB2-Produkten aufrüsten. Sie sollten außerdem Anwendungen und Scripts ändern, um Verweise auf diese nicht mehr unterstützte Funktionalität zu entfernen.

Zugriffsschutz für Pufferpoolspeicher (AIX auf POWER6)

Der Datenbankmanager verwendet den Pufferpool, um hinzugefügte, geänderte und gelöschte Daten auf viele der Datenbankdaten anzuwenden. Unter AIX 5.3 TL06+ auf POWER6-Prozessoren können Sie *Speicherschlüssel* zum Schutz des Pufferpoolspeichers verwenden.

Speicherschlüssel sind ein neues Funktionsmerkmal von IBM® Power6-Prozessoren und des AIX®-Betriebssystems, das den Schutz von Speicherbereichen über Hardwareschlüssel auf der Kernel-Thread-Ebene ermöglicht. Der Schutz durch Speicherschlüssel reduziert Probleme aufgrund von Datenverlusten im Pufferpool und begrenzt Fehler, durch die die Datenbank gestoppt werden könnte. Unzulässige Versuche, durch Programmiertechniken auf den Pufferpool zuzugreifen, verursachen eine Fehlerbedingung, die der Datenbankmanager erkennen und behandeln kann.

Anmerkung: Der Pufferpoolspeicherschutz arbeitet auf Agentenbasis. Ein bestimmter Agent erhält nur dann Zugriff auf die Pufferpoolseiten, wenn dieser Agent den Zugriff benötigt.

Der Datenbankmanager schützt Pufferpools, indem er den Zugriff auf den Pufferpoolspeicher einschränkt. Wenn ein Agent Zugriff auf die Pufferpools benötigt, um seine Arbeit zu verrichten, wird ihm der Zugriff auf den Pufferpoolspeicher temporär erteilt. Wenn der Agent den Zugriff auf die Pufferpools nicht länger benötigt, wird der Zugriff entzogen. Dadurch wird sichergestellt, dass Agenten nur dann berechtigt werden, den Inhalt von Pufferpools zu ändern, wenn dies absolut erforderlich ist. Dies verringert die Wahrscheinlichkeit von Datenverlusten in Pufferpools. Jeder unzulässige Zugriff auf den Pufferpoolspeicher führt zu einem Segmentierungsfehler. Zur Diagnose solcher Fehler stehen Tools zur Verfügung, wie zum Beispiel die Befehle db2diag, db2fodc, db2pdcfg und db2support.

Für eine vollständige Aktivierung dieser Pufferpoolspeicherschutzfunktion zur Erhöhung der Ausfallsicherheit der Datenbanksteuerkomponente sollten Sie sowohl die Registrierdatenbankvariable DB2_MEMORY_PROTECT als auch die Registrierdatenbankvariable DB2_THREAD_SUSPENSION aktivieren:

Registrierdatenbankvariable DB2_MEMORY_PROTECT
Diese Registrierdatenbankvariable aktiviert und inaktiviert die Schutzfunktion für den Pufferpoolspeicher. Wenn DB2_MEMORY_PROTECT aktiviert (auf YES gesetzt) ist und ein Thread der DB2-Steuerkomponente versucht, in unzulässiger Weise auf den Pufferpoolspeicher zuzugreifen, wird der Thread der Steuerkomponente abgefangen (Trap). Der Standardwert ist NO.

Registrierdatenbankvariable DB2_THREAD_SUSPENSION
Diese Registrierdatenbankvariable aktiviert und inaktiviert die Aussetzfunktion für DB2-Threads. Mithilfe dieser Funktion können Sie steuern, ob eine DB2-Instanz einen Trap auffangen kann, indem sie einen fehlerhaften Thread der Steuerkomponente (d. h. einen Thread, der in unzulässiger Weise versucht hat, auf den durch Speicherschlüssel geschützten Speicher zuzugreifen) aussetzt.

Hinweis:
Die Schutzfunktion für den Pufferpoolspeicher hängt von der Implementierung der AIX Storage Protect Keys (Speicherschutzschlüssel) ab und funktioniert mit dem fixierten gemeinsam genutzten Speicher möglicherweise nicht. Wenn DB2_MEMORY_PROTECT mit der Einstellung DB2_PINNED_BP oder DB2_LARGE_PAGE_MEM angegeben wird, werden AIX Storage Protect Keys möglicherweise nicht aktiviert. Weitere Informationen zu AIX Storage Protect Keys finden Sie über den folgenden Link: http://publib.boulder.ibm.com/infocenter/systems/scope/aix/index.jsp?topic=/com.ibm.aix.genprogc/doc/genprogc/storage_protect_keys.htm

Die Variable DB2_THREAD_SUSPENSION kann nur aktiviert werden, wenn DB2_MEMORY_PROTECT aktiviert ist. Wenn DB2_THREAD_SUSPENSION aktiviert ist, gilt Folgendes:

- Unabhängig davon, welche Art von Fehler in einem Thread auftritt und ob er durch einen Versuch verursacht wurde, auf Speicher zuzugreifen, der durch Speicherschlüssel geschützt wird und auf den der Thread keinen Zugriff hat, garantiert der Datenbankmanager, wenn der Fehler an einem Punkt auftritt, an dem der Thread keinen Zugriff auf diesen durch Speicherschlüssel geschützten Speicher hat, dass der geschützte Speicher nicht beschädigt wird, und lässt infolgedessen zu, dass die DB2-Steuerkomponente weiterhin ausgeführt wird.
- Wenn bei der Ausführung einer benutzerdefinierten Funktion (UDF) ohne SQL im nicht abgeschirmten Modus ein Versuch, auf den geschützten Pufferpoolspeicher zuzugreifen, erkannt wird, gibt der Datenbankmanager einen Fehler an den Aufrufer der UDF zurück und die Datenbank wird davon unbeschadet weiterhin ausgeführt.

Erstellen von Pufferpools

Mit der Anweisung CREATE BUFFERPOOL können Sie einen neuen Pufferpool definieren, der vom Datenbankmanager verwendet werden soll.

Ein Beispiel für eine grundlegende Anweisung CREATE BUFFERPOOL sieht folgendermaßen aus:

```
CREATE BUFFERPOOL <pufferpoolname>
  PAGESIZE 4096
```

Der Pufferpool kann sofort aktiv werden, wenn ausreichend Speicher verfügbar ist. Standardmäßig werden neue Pufferpools mit dem Schlüsselwort IMMEDIATE erstellt. Auf den meisten Plattformen wird der Datenbankmanager in der Lage sein, mehr Speicher zuzuordnen. Das erwartete Rückgabeergebnis ist eine erfolgreiche Speicherzuordnung. Nur wenn der Datenbankmanager den zusätzlichen Speicher nicht zuordnen kann, gibt er eine Warnung zurück, die besagt, dass der Pufferpool nicht gestartet werden konnte und dass die Zuordnung des Pufferpools beim nächsten Start der Datenbank erfolgt. Für sofortige Anforderungen (IMMEDIATE) braucht die Datenbank nicht erneut gestartet zu werden. Wenn diese Anweisung festgeschrieben wird, wird der Pufferpool in die Systemkatalogtabellen eingetragen. Er wird jedoch erst aktiv, wenn die Datenbank das nächste Mal gestartet wird. Weitere Informationen zu dieser Anweisung, einschließlich weiterer Optionen, finden Sie in der Beschreibung der Anweisung „CREATE BUFFERPOOL".

Wenn Sie eine Anweisung CREATE BUFFERPOOL DEFERRED absetzen, wird der Pufferpool nicht sofort aktiviert, sondern erst beim nächsten Start der Datenbank erstellt. Bis zum erneuten Start der Datenbank verwenden alle neuen Tabellenbereiche einen vorhandenen Pufferpool, auch wenn dieser Tabellenbereich so erstellt wird, dass er den verzögert (DEFERRED) erstellten Pufferpool explizit verwendet.

Es muss ausreichend Realspeicher auf dem System für die Gesamtgröße aller von Ihnen erstellten Pufferpools zur Verfügung stehen. Darüber hinaus benötigt auch das Betriebssystem einigen Speicher für seinen Betrieb.

Gehen Sie wie folgt vor, um einen Pufferpool über die Befehlszeile zu erstellen:

1. Rufen Sie die Liste der Pufferpoolnamen, die in der Datenbank bereits vorhanden sind, mit der folgenden SQL-Anweisung ab:

```
SELECT BPNAME FROM SYSCAT.BUFFERPOOLS
```

2. Wählen Sie einen Pufferpoolnamen aus, der sich momentan nicht in der Ergebnisliste befindet.
3. Legen Sie die Merkmale des zu erstellenden Pufferpools fest.
4. Vergewissern Sie sich, dass Sie die korrekte Berechtigungs-ID zum Ausführen der Anweisung CREATE BUFFERPOOL besitzen.
5. Führen Sie die Anweisung CREATE BUFFERPOOL aus.

Bei partitionierten Datenbanken können auch definieren, dass der Pufferpool in der Datenbankpartition unterschiedlich, zum Beispiel mit unterschiedlichen Größen, erstellt wird. Mit der Standardklausel ALL DBPARTITIONNUMS wird angegeben, dass dieser Pufferpool in allen Datenbankpartitionen der Datenbank erstellt werden soll.

Im folgenden Beispiel gibt die optionale Klausel DATABASE PARTITION GROUP die Datenbankpartitionsgruppe (bzw. -gruppen) an, für die die Pufferpooldefinition gilt:

```
CREATE BUFFERPOOL <pufferpoolname>
  PAGESIZE 4096
  DATABASE PARTITION GROUP <db_partitionsgruppenname>
```

Wenn dieser Parameter angegeben wird, wird der Pufferpool nur in Datenbankpartitionen in diesen Datenbankpartitionsgruppen erstellt. Jede Datenbankpartitionsgruppe muss zu dem gegebenen Zeitpunkt in der Datenbank vorhanden sein. Wenn die Klausel DATABASE PARTITION GROUP nicht angegeben wird, wird dieser Pufferpool in allen Datenbankpartitionen (sowie in allen Datenbankpartitionen, die der Datenbank nachträglich hinzugefügt werden) erstellt.

Weitere Informationen finden Sie in der Beschreibung der Anweisung „CREATE BUFFERPOOL".

Ändern von Pufferpools

Es gibt eine Reihe von Gründen, aus denen es sinnvoll sein kann, einen Pufferpool zu ändern. Dazu gehört zum Beispiel die Aktivierung von Speicher mit automatischer Leistungsoptimierung. Zum Ändern von Pufferpools verwenden Sie die Anweisung ALTER BUFFERPOOL.

Die Berechtigungs-ID der Anweisung muss über die Berechtigung SYSCTRL oder SYSADM verfügen.

Wenn Sie mit Pufferpools arbeiten, müssen Sie möglicherweise eine der folgenden Tasks ausführen:

- Aktivieren der automatischen Optimierung für einen Pufferpool, sodass der Datenbankmanager die Größe des Pufferpools als Reaktion auf die Anforderungen Ihrer Auslastung anpassen kann
- Ändern des Blockbereichs des Pufferpools zur Unterstützung der blockbasierten Ein-/Ausgabe
- Hinzufügen dieser Pufferpooldefinition zu einer neuen Datenbankpartitionsgruppe
- Ändern der Größe des Pufferpools in einigen oder allen Datenbankpartitionen

Zum Ändern eines Pufferpools über die Befehlszeile gehen Sie wie folgt vor:

1. Zum Abrufen der Liste der Pufferpoolnamen, die in der Datenbank bereits vorhanden sind, geben Sie die folgende Anweisung ein:

   ```
   SELECT BPNAME FROM SYSCAT.BUFFERPOOLS
   ```

2. Wählen Sie den Pufferpoolnamen in der Ergebnisliste aus.
3. Legen Sie fest, welche Änderungen vorgenommen werden müssen.
4. Vergewissern Sie sich, dass die korrekte Berechtigungs-ID zum Ausführen der Anweisung ALTER BUFFERPOOL vorhanden ist.

   Anmerkung: Bei den Parametern `IMMEDIATE` und `DEFERRED` handelt es sich um Schlüsselparameter. Mit IMMEDIATE wird die Pufferpoolgröße geändert, ohne dass die nächste Datenbankaktivierung abgewartet werden muss, um die Änderung wirksam zu machen. Wenn nicht ausreichend gemeinsam verwendeter Datenbankspeicher vorhanden ist, um neuen Speicherbereich zuzuordnen, wird die Anweisung verzögert (DEFERRED) ausgeführt.

   Bei Angabe von DEFERRED werden die Änderungen an dem Pufferpool erst angewendet, wenn die Datenbank erneut aktiviert wird. Reservierter Speicherbereich ist nicht erforderlich. Der Datenbankmanager ordnet den erforderlichen Speicher bei der Aktivierung automatisch aus dem System zu.
5. Mit der Anweisung ALTER BUFFERPOOL können Sie ein einzelnes Attribut des Pufferpoolobjekts ändern. Beispiel:

   ```
   ALTER BUFFERPOOL pufferpoolname SIZE anzahl_seiten
   ```

- Die Angabe *pufferpoolname* ist ein einteiliger Name, der einen Pufferpool angibt, der in den Systemkatalogen beschrieben ist.
- Die Angabe *anzahl_seiten* ist die neue Anzahl von Seiten, die diesem bestimmten Pufferpool zugeordnet werden soll. Sie können auch den Wert `-1` verwenden, um anzugeben, dass die Größe des Pufferpools dem Wert des Datenbankkonfigurationsparameters **buffpage** entsprechen soll.

Die Anweisung kann außerdem die Klausel DBPARTITIONNUM <datenbankpartitionsnummer> enthalten, um die Datenbankpartition anzugeben, in der die Größe des Pufferpools geändert werden soll. Wenn diese Klausel nicht angegeben wird, wird die Größe des Pufferpools in allen Datenbankpartitionen mit Ausnahme derer, die einen Ausnahmeeintrag in der Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS haben, geändert. Detaillierte Informationen zur Verwendung dieser Klausel für Datenbankpartitionen finden Sie in der Beschreibung der Anweisung ALTER BUFFERPOOL.

Änderungen am Pufferpool durch diese Anweisung werden in die Systemkatalogtabellen eingetragen, wenn die Anweisung festgeschrieben (COMMIT) ist. Die Änderungen am Pufferpool werden jedoch erst wirksam, wenn die Datenbank das nächste Mal gestartet wird, sofern die Anweisung ALTER BUFFERPOOL nicht mit dem Standardschlüsselwort IMMEDIATE erfolgreich ausgeführt wurde.

Es muss ausreichend Realspeicher auf dem System für die Gesamtgröße aller von Ihnen erstellten Pufferpools zur Verfügung stehen. Darüber hinaus muss auch genügend Realspeicher für die übrigen Komponenten des Datenbankmanagers sowie für Ihre Anwendung vorhanden sein.

Löschen von Pufferpools

Stellen Sie beim Löschen von Pufferpools sicher, dass diesen Pufferpools keine Tabellenbereiche zugeordnet sind. Der Pufferpool IBMDEFAULTBP kann nicht gelöscht werden.

Der Plattenspeicher wird möglicherweise erst freigegeben, wenn die nächste Verbindung zur Datenbank hergestellt wird. Der Speicher eines gelöschten Pufferpools wird erst dann tatsächlich freigegeben, wenn die Datenbank gestoppt wird. Der Pufferpoolspeicher wird sofort zur Verwendung durch den Datenbankmanager freigegeben.

Sie können Pufferpools mit der Anweisung DROP BUFFERPOOL wie folgt löschen:

```
DROP BUFFERPOOL <pufferpoolname>
```

Kapitel 9. Tabellenbereiche

Ein *Tabellenbereich* ist eine Speicherstruktur, die Tabellen, Indizes, LOB-Daten und LONG-Daten enthält. Tabellenbereiche befinden sich in Datenbankpartitionsgruppen. Mit ihrer Hilfe können Sie die Speicherposition der Datenbank- und Tabellendaten direkt bestimmten Containern zuweisen. (Ein Container kann ein Verzeichnisname, Einheitenname oder Dateiname sein.) Die möglichen Vorzüge sind eine bessere Leistung und eine flexiblere Konfiguration.

Da sich Tabellenbereiche in Datenbankpartitionsgruppen befinden, bestimmt der zur Speicherung einer Tabelle ausgewählte Tabellenbereich, wie die Daten dieser Tabelle auf die Datenbankpartitionen in einer Datenbankpartitionsgruppe verteilt werden. Ein einzelner Tabellenbereich kann sich über mehrere Container erstrecken. Mehrere Container (eines oder mehrerer Tabellenbereiche) können auf derselben physischen Platte (bzw. auf demselben Laufwerk) erstellt werden. Wenn Sie Tabellenbereiche mit dynamischem Speicher verwenden, wird dies vom Datenbankmanager verwaltet. Wenn Sie *keine* Tabellenbereiche mit dynamischem Speicher verwenden, sollte sich jeder Container zur Erzielung einer besseren Leistung auf einem anderen Datenträger befinden.

Abb. 6 zeigt ein Beispiel für die Beziehung zwischen Tabellen und Tabellenbereichen innerhalb einer Datenbank und den Containern, die dieser Datenbank zugeordnet sind.

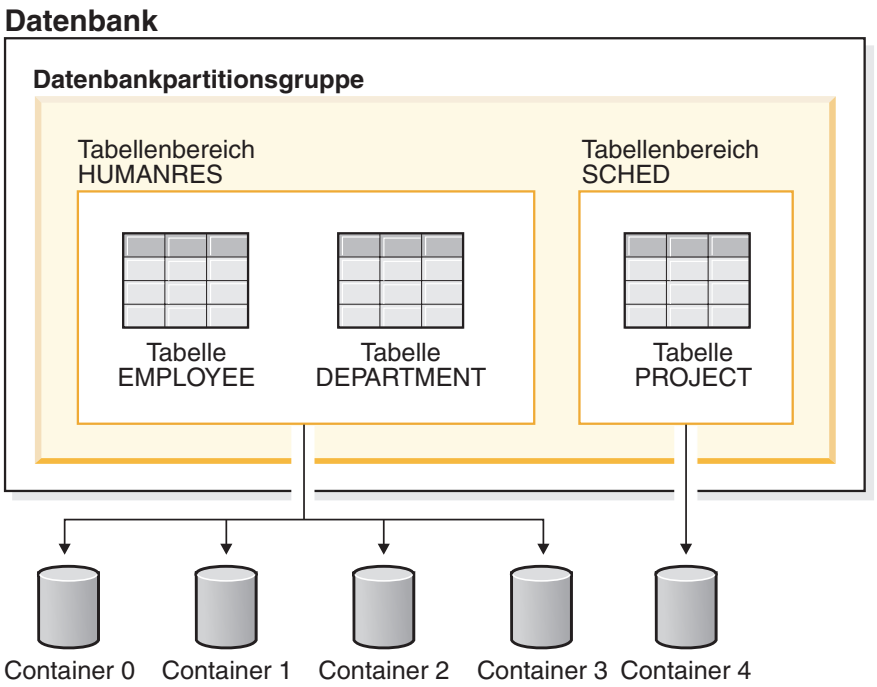


Abbildung 6. Tabellenbereiche und Tabellen in einer Datenbank

Die Tabellen EMPLOYEE und DEPARTMENT befinden sich im Tabellenbereich HUMANRES, der sich über die Container 0, 1, 2 und 3 erstreckt. Die Tabelle PROJECT befindet sich im Tabellenbereich SCHED in Container 4. In diesem Beispiel befindet sich jeder Container auf einer separaten Platte.

Der Datenbankmanager versucht, die Datenmenge möglichst gleichmäßig über die Container zu verteilen. Das heißt, es werden alle Container zum Speichern der Daten verwendet. Die Anzahl der Seiten, die der Datenbankmanager in einen Container schreibt, bevor er einen anderen Container verwendet, wird mit dem Parameter *EXTENTSIZE* definiert. Der Datenbankmanager beginnt beim Speichern der Tabellendaten nicht immer mit dem ersten Container.

Abb. 7 zeigt den Tabellenbereich HUMANRES mit einem Wert von zwei 4-KB-Seiten für EXTENTSIZE und vier Containern mit einer kleinen Anzahl zugeordneter Speicherbereiche. Die Tabellen DEPARTMENT und EMPLOYEE haben jeweils sieben Seiten und verteilen sich über alle vier Container.

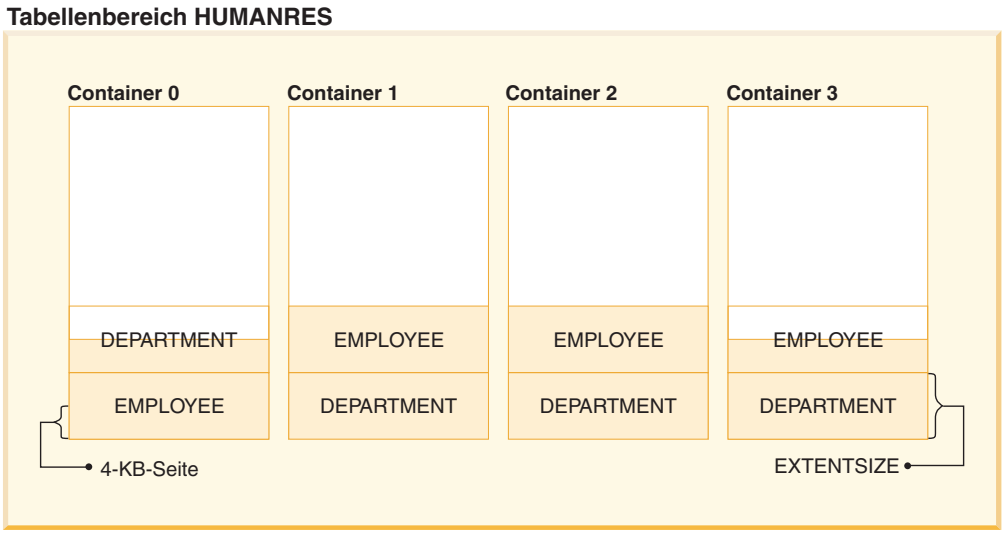


Abbildung 7. Container und EXTENTSIZE große Speicherbereiche in einem Tabellenbereich

Entwerfen von Tabellenbereichen

Mithilfe von Tabellenbereichen wird angegeben, wo die Daten in einer Datenbank physisch auf einem System gespeichert werden, und eine Abstraktionsebene zwischen der Datenbank und den Containerobjekten, in denen sich die tatsächlichen Daten befinden, bereitgestellt.

Für die Erstellung von Tabellenbereichen gibt es zahlreiche Gründe. Dazu gehören zum Beispiel die Sicherstellung der Wiederherstellbarkeit und die Möglichkeit, Objekte in verschiedenen Pufferpools zu isolieren. Bei Verwendung der Funktionalität des dynamischen Speichers brauchen Sie sich um die physische Plattenposition von Tabellenbereichen oder die physische Position von Containern keine Gedanken mehr zu machen. Der Datenbankmanager sorgt automatisch für die Zuordnung oder Erstellung von Containern für Tabellenbereiche.

Eine Datenbank muss mindestens drei Tabellenbereiche enthalten:

- Einen *Katalogtabellenbereich*, der alle Systemkatalogtabellen für die Datenbank enthält. Dieser Tabellenbereich heißt SYSCATSPACE und kann nicht gelöscht werden. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMCATGROUP.

- Einen oder mehrere *Benutzertabellenbereiche*, die alle benutzerdefinierten Tabellen enthalten. Standardmäßig wird ein Tabellenbereich namens USERSPACE1 erstellt. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMDEFAULTGROUP.

  Beim Erstellen einer Tabelle sollten Sie einen Tabellenbereichsnamen angeben, andernfalls kann es zu unerwünschten Ergebnissen kommen.

  Die Seitengröße für eine Tabelle wird entweder durch die Zeilengröße oder durch die Anzahl von Spalten bestimmt. Die für eine Zeile maximal zulässige Länge hängt von der Seitengröße des Tabellenbereichs ab, in dem die Tabelle erstellt wird. Gültige Werte für die Seitengröße sind 4 KB, 8 KB, 16 KB und 32 KB. Vor Version 9.1 war die Standardseitengröße 4 KB. Ab Version 9.1 kann die Standardseitengröße einen der anderen unterstützten Werte haben. Die Standardseitengröße wird bei der Erstellung einer neuen Datenbank deklariert. Auch mit deklarierter Standardseitengröße haben Sie weiterhin die Freiheit, einen Tabellenbereich mit einer Seitengröße für die Tabelle und einen anderen Tabellenbereich mit einer anderen Seitengröße für lange Daten oder LOB-Daten zu erstellen. (Hier ist wiederum zu beachten, dass SMS keine Tabellen unterstützt, die sich über mehrere Tabellenbereiche erstrecken, im Gegensatz zu DMS.) Wenn die Anzahl der Spalten oder die Zeilengröße die Grenze für die Seitengröße eines Tabellenbereichs überschreitet, wird ein Fehler zurückgegeben (SQLSTATE 42997).
- Einen oder mehrere *temporäre Tabellenbereiche*, die temporäre Tabellen enthalten. Temporäre Tabellenbereiche können *temporäre Systemtabellenbereiche* oder *temporäre Benutzertabellenbereiche* sein.

  Temporäre Systemtabellenbereiche enthalten temporäre Daten, die der Datenbankmanager zur Ausführung von Operationen wie Sortierungen und Joins benötigt. Diese Typen von Operationen erfordern zusätzlichen Speicher zur Verarbeitung der Ergebnismengen. Eine Datenbank muss mindestens über einen temporären Tabellenbereich verfügen. Standardmäßig wird ein temporärer Systemtabellenbereich namens TEMPSPACE1 bei der Erstellung der Datenbank erstellt. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMTEMPGROUP.

  Temporäre Benutzertabellenbereiche enthalten temporäre Daten aus Tabellen, die mit einer Anweisung DECLARE GLOBAL TEMPORARY TABLE erstellt wurden. Um die Definition deklarierter temporärer Tabellen zu ermöglichen, muss mindestens ein temporärer Benutzertabellenbereich mit den entsprechenden USE-Zugriffsrechten erstellt werden. USE-Zugriffsrechte werden mithilfe der Anweisung GRANT erteilt. Ein temporärer Benutzertabellenbereich wird *nicht* standardmäßig bei der Erstellung einer Datenbank erstellt.

  Wenn eine Datenbank mehr als einen temporären Tabellenbereich verwendet und ein neues temporäres Objekt benötigt wird, wählt das Optimierungsprogramm eine geeignete Seitengröße für dieses Objekt aus. Anschließend wird dieses Objekt dem temporären Tabellenbereich mit der entsprechenden Seitengröße zugeordnet. Wenn mehr als ein temporärer Tabellenbereich mit dieser Seitengröße vorhanden ist, werden die Tabellenbereiche reihum ausgewählt. In den meisten Fällen ist es nicht empfehlenswert, mehr als einen temporären Tabellenbereich für eine bestimmte Seitengröße zu haben.

  Wenn Abfragen auf Tabellen in Tabellenbereichen ausgeführt werden, die mit größeren Seitengrößen als dem Standardwert definiert sind, schlagen manche dieser Abfragen möglicherweise fehl. Dies geschieht, wenn keine temporären Tabellenbereiche mit einer größeren Seitengröße definiert sind. Möglicherweise müssen Sie einen temporären Tabellenbereich mit einer größeren Seitengröße erstellen. (Wenn der Standardwert zum Beispiel 4 KB beträgt, müssen Sie einen temporären Tabellenbereich mit einer Seitengröße von 8 KB, 16 KB oder 32 KB

erstellen.) Jede DML-Anweisung (Datenbearbeitungssprache) könnte fehlschlagen, sofern kein temporärer Tabellenbereich mit derselben Seitengröße wie die größte in dem Benutzertabellenbereich verwendete Seitengröße vorhanden ist.

Sie sollten einen einzelnen temporären SMS-Tabellenbereich definieren, dessen Seitengröße der Seitengröße entspricht, die in den meisten der Benutzertabellenbereiche verwendet wird. Dies sollte für typische Umgebungen und Auslastungen angemessen sein.

Tabellenbereiche und Datenbankpartitionsgruppen

In einer Umgebung mit partitionierten Datenbanken wird jeder Tabellenbereich einer bestimmten Datenbankpartitionsgruppe zugeordnet. Dadurch können die Merkmale des Tabellenbereichs auf jede Datenbankpartition in der Datenbankpartitionsgruppe angewendet werden.

Die Datenbankpartitionsgruppe muss vorhanden sein (sie wird mit der Anweisung CREATE DATABASE PARTITION GROUP definiert). Die Zuordnung zwischen dem Tabellenbereich und der Datenbankpartitionsgruppe wird bei der Erstellung des Tabellenbereichs über die Anweisung CREATE TABLESPACE definiert.

Die Zuordnung zwischen dem Tabellenbereich und der Datenbankpartitionsgruppe kann mit der Anweisung ALTER TABLESPACE nicht geändert werden. Sie können lediglich die Spezifikation des Tabellenbereichs für einzelne Datenbankpartitionen innerhalb der Datenbankpartitionsgruppe ändern. In einer Umgebung mit einer Einzelpartition wird jeder Tabellenbereich der Standarddatenbankpartitionsgruppe zugeordnet. Die Standarddatenbankpartitionsgruppe bei der Definition eines Tabellenbereichs ist IBMDEFAULTGROUP, sofern kein temporärer Systemtabellenbereich definiert wird. In diesem Fall wird die Standarddatenbankpartitionsgruppe IBMTEMPGROUP verwendet.

Typen von Tabellenbereichen

Eine Datenbank muss mindestens drei Typen von Tabellenbereichen enthalten: einen Katalogtabellenbereich, mindestens einen Tabellenbereich für Benutzertabellen und mindestens einen Tabellenbereich für temporäre Tabellen.

Es gibt zwei Typen von Tabellenbereichen, die beide in einer einzelnen Datenbank verwendet werden können:

- Vom System verwaltete Bereiche (SMS - System Managed Space), bei denen der Dateimanager des Betriebssystems den Speicherbereich steuert
- Von der Datenbank verwaltete Bereiche (DMS - Database Managed Space), bei denen der Datenbankmanager den Speicherbereich steuert

In einer Umgebung mit partitionierten Datenbanken verfügt die Katalogpartition über alle drei Tabellenbereiche, während die anderen Datenbankpartitionen nur die Tabellenbereiche TEMPSPACE1 und USERSPACE1 enthalten.

Darüber hinaus können auch dynamische Speicherbereiche erstellt werden, die mit SMS- oder DMS-Tabellenbereichen als zugrunde liegendem Tabellenbereichstyp arbeiten. Der tatsächliche Typ (SMS oder DMS) wird vom Datenbankmanager in Abhängigkeit vom Typ der Daten gewählt, die in dem jeweiligen Tabellenbereich gespeichert werden sollen (SMS für Tabellenbereiche mit temporären Tabellen, ansonsten DMS).

Vom Betriebssystem verwalteter Speicherbereich (SMS)

In einem vom Betriebssystem verwalteten Tabellenbereich (SMS - System Managed Space) ordnet der Dateisystemmanager des Betriebssystems den Speicherbereich zu, in dem die Tabelle gespeichert wird, und verwaltet diesen Bereich.

Das Speichermodell enthält in der Regel viele Dateien, die Tabellenobjekte darstellen, die im Speicherbereich des Dateisystems gespeichert sind. Als zuständiger Benutzer entscheiden Sie über die Speicherposition der Dateien, der Datenbankmanager verwaltet ihre Namen, und das Dateisystem ist für ihre Verwaltung zuständig. Durch Steuern der Datenmenge, die in jede Datei geschrieben wird, verteilt der Datenbankmanager die Daten gleichmäßig auf die Container der Tabellenbereiche.

Jeder Tabelle ist mindestens eine physische SMS-Datei zugeordnet.

Die Daten in den Tabellenbereichen werden einheitenübergreifend in Speicherbereichen in allen Containern des Systems gespeichert. Ein *Speicherbereich* ist eine Gruppe von aufeinander folgenden Seiten, die für die Datenbank definiert sind. Die Dateierweiterung bezeichnet den Datentyp, der in der betreffenden Datei gespeichert wird. Zur gleichmäßigen Datenverteilung auf alle Container im Tabellenbereich, werden die Anfangsspeicherbereiche für Tabellen reihum in allen Containern angelegt. Eine solche Verteilung der Speicherbereiche ist besonders wichtig, wenn die Datenbank zahlreiche kleinen Tabellen enthält. Das DB2-Striping wird empfohlen, wenn Daten in mehrere Container geschrieben werden. Wenn Sie ein Platten-Striping zusammen mit dem DB2-Striping implementieren, sollten der Wert für EXTENTSIZE des Tabellenbereichs und die Stripegröße der Platte identisch sein.

In einem SMS-Tabellenbereich wird der Speicher für Tabellen bei Bedarf zugeordnet. Die zugeordnete Speichergröße hängt von der Einstellung des Datenbankkonfigurationsparameters *multipage_alloc* ab. Wenn dieser Konfigurationsparameter auf den Wert YES gesetzt wird, wird ein Speicherbereich in voller EXTENTSIZE-Größe (in der Regel zwei oder mehr Seiten) zugeordnet, wenn Speicherbereich benötigt wird. Ansonsten wird Speicherbereich jedes Mal in der Größe von einer Seite zugeordnet.

Die Zuordnung aus mehreren Seiten bestehender Dateien ist standardmäßig aktiviert. Der Wert des Datenbankkonfigurationsparameters 'multipage_alloc' zeigt an, ob die Zuordnung aus mehreren Seiten bestehender Dateien aktiviert ist.

Anmerkung: Die Zuordnung aus mehreren Seiten bestehender Dateien gilt nicht für Tabellenbereiche für temporäre Tabellen.

Die mehrseitige Dateizuordnung betrifft nur die Daten- und Indexteile einer Tabelle. Das heißt, die Dateien .LF, .LB und .LBA werden nicht jedes Mal um eine EXTENTSIZE-Größe erweitert.

Wenn der gesamte Speicherplatz in einem einzelnen Container in einem SMS-Tabellenbereich Tabellen zugeordnet ist, wird der Tabellenbereich als voll betrachtet, selbst wenn noch weiterer Speicherplatz in anderen Containern verblieben ist. Sie können einem SMS-Tabellenbereich Container nur in einer Datenbankpartition hinzufügen, die noch keine Container hat.

Anmerkung: SMS-Tabellenbereiche können die Vorteile des Vorablesezugriffs und der Cachefunktion des Dateisystems nutzen.

SMS-Tabellenbereiche werden über die Option MANAGED BY SYSTEM im Befehl CREATE DATABASE oder in der Anweisung CREATE TABLESPACE definiert. Dabei müssen Sie zwei Schlüsselfaktoren beim Entwurf Ihrer SMS-Tabellenbereiche beachten:

- Container für den Tabellenbereich

  Sie müssen die Anzahl der Container angeben, die Sie für Ihren Tabellenbereich verwenden wollen. Es ist sehr wichtig, alle gewünschten Container zu ermitteln, weil Sie nach dem Erstellen des Tabellenbereichs keine Container löschen oder hinzufügen können. In einer Umgebung mit partitionierten Datenbanken kann die Anweisung ALTER TABLESPACE verwendet werden, um beim Hinzufügen einer neuen Datenbankpartition zu einer Datenbankpartitionsgruppe für einen SMS-Tabellenbereich Container für die neue Datenbankpartition hinzuzufügen.

  Jeder Container, der für einen SMS-Tabellenbereich verwendet wird, gibt einen absoluten oder relativen Verzeichnisnamen an. Jedes dieser Verzeichnisse kann sich auf einem anderen Dateisystem (oder einer anderen physischen Platte) befinden. Die Maximalgröße des Tabellenbereichs kann wie folgt abgeschätzt werden:

  ```
  Anzahl von Containern * (maximale Dateisystemgröße, die vom
     Betriebssystem unterstützt wird)
  ```

  Diese Formel setzt voraus, dass jedem Container ein bestimmtes Dateisystem zugeordnet wird und dass für jedes Dateisystem der maximale Speicherbereich verfügbar ist. In der Praxis ist dies möglicherweise nicht der Fall, und die Maximalgröße des Tabellenbereichs kann sehr viel kleiner sein. Darüber hinaus bestehen auch Einschränkungen durch SQL bezüglich der Größe von Datenbankobjekten, die sich auf die maximale Größe eines Tabellenbereichs auswirken können.

  Anmerkung: Gehen Sie beim Definieren der Container mit besonderer Sorgfalt vor. Wenn die Container bereits Dateien oder Verzeichnisse enthalten, wird eine Fehlernachricht (SQL0298N) zurückgegeben.

- Parameter EXTENTSIZE für den Tabellenbereich

  Der Wert für den Parameter EXTENTSIZE kann nur beim Erstellen des Tabellenbereichs angegeben werden. Da spätere Änderungen daran nicht möglich sind, ist es wichtig, einen geeigneten Wert für EXTENTSIZE anzugeben.

  Wenn Sie beim Erstellen eines Tabellenbereichs für den Parameter EXTENTSIZE keinen Wert angeben, erstellt der Datenbankmanager den Tabellenbereich mit dem Standardwert, der durch den Konfigurationsparameter *dft_extent_sz* der Datenbank definiert ist. Dieser Konfigurationsparameter wird anfangs auf der Grundlage der Informationen gesetzt, die beim Erstellen der Datenbank angegeben werden. Wird der Parameter *dft_extent_sz* nicht mit dem Befehl CREATE DATABASE angegeben, wird der Standardwert auf 32 gesetzt.

Um geeignete Werte für die Anzahl der Container und für den Parameter EXTENTSIZE für den Tabellenbereich festlegen zu können, benötigen Sie folgende Kenntnisse:

- Den oberen Grenzwert, den Ihr Betriebssystem für die Größe eines logischen Dateisystems festlegt

  Zum Beispiel haben einige Betriebssysteme eine obere Begrenzung von 2 GB. Wenn Sie also ein Tabellenobjekt mit einer Größe von 64 GB erstellen möchten, benötigen Sie auf dieser Art System mindestens 32 Container.

  Wenn Sie einen Tabellenbereich erstellen, können Sie Container angeben, die sich auf verschiedenen Dateisystemen befinden, und dadurch die Menge der Daten erhöhen, die in der Datenbank gespeichert werden können.

- Die Art und Weise, wie der Datenbankmanager die einem Tabellenbereich zugeordneten Datendateien und Container verwaltet

  Die erste Datei mit Tabellendaten (SQL00002.DAT) wird im ersten für den Tabellenbereich angegebenen Container erstellt. Diese Datei darf so weit anwachsen, bis sie die durch den Wert für EXTENTSIZE festgelegte Größe erreicht. Nach Erreichen dieser Größe schreibt der Datenbankmanager Daten in die Datei SQL00002.DAT im nächsten Container. Dieser Prozess wird fortgesetzt, bis alle Container Dateien des Namens SQL00002.DAT enthalten. Wenn dies eintritt, kehrt der Datenbankmanager zum ersten Container zurück. Dieser Prozess (der auch als *Striping - einheitenübergreifendes Lesen und Schreiben von Daten* bezeichnet wird) wird über die Containerverzeichnisse fortgesetzt, bis ein Container voll ist (SQL0289N) oder vom Betriebssystem kein weiterer Speicherbereich mehr zugeordnet werden kann (Fehlernachricht: Datenträger voll). Das Striping betrifft die Blockzuordnungsdateien (SQLnnnnn.BKM), Indexobjekte sowie andere Objekte, die zum Speichern von Tabellendaten verwendet werden.

  Anmerkung: Der SMS-Tabellenbereich ist voll, sobald irgendeiner seiner Container voll ist. Daher ist es wichtig, dass jedem Container dieselbe Menge an Speicherbereich zur Verfügung steht.

  Um die Daten gleichmäßiger über die Container zu verteilen, bestimmt der Datenbankmanager den Container, der zuerst verwendet werden soll, indem er den Wert der Tabellenkennung (SQL00002.DAT im obigen Beispiel) und die Anzahl der Container als Faktoren verwendet. Die Container werden beginnend mit dem Wert 0 durchnummeriert.

Von der Datenbank verwalteter Speicherbereich (DMS)

In einem DMS-Tabellenbereich (DMS = Database Managed Space) steuert der Datenbankmanager den Speicherbereich.

Das Speichermodell besteht aus einer begrenzten Anzahl Einheiten, deren Speicherbereich vom Datenbankmanager verwaltet wird. Der Datenbankadministrator entscheidet, welche Einheiten und Dateien zu verwenden sind, und verwaltet den Speicherbereich dieser Einheiten und Dateien. Der Tabellenbereich ist im Wesentlichen die Implementierung eines Dateisystems, das einem bestimmten Zweck dient und dazu entwickelt wurde, die Anforderungen des Datenbankmanagers optimal zu erfüllen.

DMS-Tabellenbereiche unterscheiden sich von SMS-Tabellenbereichen dadurch, dass der Speicherbereich für DMS-Tabellenbereiche bei der Erstellung des Tabellenbereichs zugeordnet wird. Bei SMS-Tabellenbereichen wird der Speicherbereich nach Bedarf zugeordnet. Ein DMS-Tabellenbereich für benutzerdefinierte Tabellen und Daten kann als *regulärer Tabellenbereich* oder als *LOB-Tabellenbereich* zum Speichern aller Tabellen- oder Indexdaten definiert werden.

Beachten Sie beim Entwerfen Ihrer DMS-Tabellenbereiche und Container Folgendes:

- Der Datenbankmanager arbeitet mit einheitenübergreifendem Lesen und Schreiben von Daten (Striping), um eine gleichmäßige Verteilung von Daten auf alle Container sicherzustellen. Dadurch werden die Daten gleichmäßig in alle Container im Tabellenbereich geschrieben, wobei die EXTENTSIZE großen Speicherbereiche für Tabellen reihum in allen Containern angelegt werden. Das DB2-Striping wird empfohlen, wenn Daten in mehrere Container geschrieben werden. Wenn Sie ein Platten-Striping zusammen mit dem DB2-Striping implementieren, sollten der Wert für EXTENTSIZE des Tabellenbereichs und die Stripegröße der Platte identisch sein.

- Die Maximalgröße eines regulären Tabellenbereichs beträgt 512 GB bei 32-KB-Seiten. Die Maximalgröße von LOB-Tabellenbereichen beträgt 16 TB. Informationen zur maximalen Größe regulärer Tabellenbereiche für andere Seitengrößen finden Sie in SQL- und XML-Begrenzungen.
- Im Unterschied zu SMS-Tabellenbereichen müssen die Container, die einen DMS-Tabellenbereich bilden, nicht die gleiche Größe haben. Dies wird im Normalfall jedoch nicht empfohlen, da es zu ungleichmäßiger Verteilung (Striping) auf die Container und nicht optimaler Leistung führt. Wenn ein Container voll ist, wird in DMS-Tabellenbereichen jeder verfügbare freie Speicherbereich anderer Container genutzt.
- Da der Speicherbereich vorab zugeordnet wird, muss er zur Verfügung stehen, bevor der Tabellenbereich erstellt werden kann. Bei Verwendung von Einheitencontainern muss die Einheit ebenfalls mit genügend Speicherbereich für die Definition des Containers verfügbar sein. Auf jeder Einheit kann nur ein Container definiert werden. Um eine Verschwendung von Speicherbereich zu vermeiden, sollten die Größe der Einheit und die Größe des Containers äquivalent sein. Wenn z. B. die Einheit mit 5000 Seiten zugeordnet ist, und der Einheitencontainer zum Zuordnen von 3000 Seiten definiert ist, sind 2000 Seiten der Einheit nicht verwendbar.
- Standardmäßig wird ein EXTENTSIZE großer Speicherbereich in jedem Container für Systemaufwand reserviert. Nur ganze, durch EXTENTSIZE definierte Speicherbereiche werden verwendet. Für eine optimale Speicherverwaltung können Sie daher beim Zuordnen eines Containers eine geeignete Größe anhand der folgenden Formel bestimmen:

  ```
  extent_size * (n + 1)
  ```

  Dabei ist *extent_size* die Größe jedes durch EXTENTSIZE definierten Speicherbereichs im Tabellenbereich, und *n* ist die Anzahl dieser Speicherbereiche, die Sie in dem Container speichern wollen.
- Die Mindestgröße eines DMS-Tabellenbereichs beträgt fünf EXTENTSIZE große Speicherbereiche. Jeder Versuch, einen Tabellenbereich zu erstellen, der kleiner als fünf EXTENTSIZE-Bereiche ist, führt zu einem Fehler (SQL1422N).
  - Drei EXTENTSIZE große Speicherbereiche im Tabellenbereich sind für den Systemaufwand reserviert.
  - Mindestens zwei EXTENTSIZE große Speicherbereiche sind erforderlich, um Tabellendaten des Benutzers zu speichern. (Diese zwei Speicherbereiche sind für die regulären Daten einer Tabelle vorgesehen, und nicht für Index-, Langfeld- oder LOB-Daten, die eigene Speicherbereiche benötigen.)
- Einheitencontainer müssen logische Datenträger mit einer „zeichenspezifischen Schnittstelle" (d. h. keine physischen Datenträger) verwenden.
- Für DMS-Tabellenbereiche können auch Dateien anstelle von Einheiten (devices) verwendet werden. Mit dem Standardtabellenbereichsattribut NO FILE SYSTEM CACHING in Viper 2 können Dateien nahe an Einheiten ausgeführt werden; der Vorteil ist, dass keine Einheiten eingerichtet werden müssen. Weitere Informationen hierzu finden Sie in „Tabellenbereiche ohne Dateisystemcaching" auf Seite 197.
- Wenn LOB- oder LONG VARCHAR-Daten verarbeitet werden, kann die Verwendung des Dateisystemcache Leistungsvorteile erbringen.

  Anmerkung: LOB- und LONG VARCHAR-Daten werden nicht vom Pufferpool des Datenbankmanagers zwischengespeichert (gepuffert).

- Einige Betriebssysteme erlauben den Betrieb physischer Einheiten, die größer als 2 GB sind. Sie sollten in Betracht ziehen, die physische Einheit in logische Einheiten zu unterteilen, damit kein Container größer als die vom Betriebssystem zugelassene Größe ist.

Anmerkung: Ebenso wie SMS-Tabellenbereiche können DMS-Dateicontainer die Vorteile des Vorablesezugriffs und der Cachefunktion des Dateisystems nutzen. DMS-Tabellenbereiche, die mit Containern für Roheinheiten arbeiten, können dies hingegen nicht.

Beim Arbeiten mit DMS-Tabellenbereichen sind zwei Containeroptionen verfügbar: unformatierte Einheiten (Roheinheiten) und Dateien. Beim Arbeiten mit Dateicontainern ordnet der Datenbankmanager den gesamten Container bei der Erstellung des Tabellenbereichs zu. Ein Ergebnis dieser ersten Zuordnung des gesamten Tabellenbereichs besteht darin, dass die physische Zuordnung, zwar nicht garantiert, jedoch normalerweise direkt aufeinander folgend erfolgt, obwohl das Dateisystem die Zuordnung vornimmt. Beim Arbeiten mit Containern für Roheinheiten übernimmt der Datenbankmanager die Steuerung der gesamten Einheit und stellt immer sicher, dass die Seiten in einem Speicherbereich direkt aufeinander folgen.

Beim Arbeiten mit DMS-Tabellenbereichen sollten Sie in Betracht ziehen, jeden Container einer anderen Platte zuzuordnen. Dadurch wird die Tabellenbereichskapazität vergrößert und die Möglichkeit geschaffen, parallele E/A-Operationen zu nutzen.

Die Anweisung CREATE TABLESPACE erstellt einen neuen Tabellenbereich in einer Datenbank, ordnet diesem Tabellenbereich Container zu und trägt die Definition und die Attribute des Tabellenbereichs in den Katalog ein. Wenn Sie einen Tabellenbereich erstellen, wird EXTENTSIZE als Anzahl zusammenhängender Seiten definiert. Der EXTENTSIZE große Speicherbereich ist die Speicherzuordnungseinheit innerhalb eines Tabellenbereichs. Nur jeweils eine Tabelle oder ein Objekt, zum Beispiel ein Index, kann die Seiten in einem einzelnen Speicherbereich verwenden. Allen Objekten, die in dem Tabellenbereich erstellt werden, werden Speicherbereiche in einer logischen Adressenzuordnung des Tabellenbereichs zugeordnet. Die Zuordnung von Speicherbereichen wird über Speicherzuordnungsseiten (SMP = Space Map Pages) verwaltet.

Der erste Speicherbereich in der logischen Adressenzuordnung für den Tabellenbereich besteht aus den Kopfdaten für den Tabellenbereich, die interne Steuerdaten enthalten. Der zweite Speicherbereich ist der erste Speicherbereich der SMP für den Tabellenbereich. SMP-Speicherbereiche sind in regelmäßigen Abständen über den gesamten Tabellenbereich verteilt. Jeder SMP-Speicherbereich besteht aus einer Bitzuordnung der Speicherbereiche vom aktuellen SMP-Speicherbereich bis zum nächsten SMP-Speicherbereich. Die Bitzuordnung dient zur Verfolgung, welche der dazwischenliegenden Speicherbereiche in Gebrauch sind.

Der auf die SMP folgende Speicherbereich ist die Objekttabelle für den Tabellenbereich. Die Objekttabelle ist eine interne Tabelle, die aufzeichnet, welche Benutzerobjekte im Tabellenbereich vorhanden sind und wo sich deren erster EMP-Speicherbereich (EMP = Extent Map Page, Speicherbereichsmaskenseite) befindet. Jedes Objekt verfügt über eigene EMPs, die eine Maske zu allen Seiten des Objekts darstellen, das in der logischen Adressenzuordnung für den Speicherbereich gespeichert ist. Abb. 8 auf Seite 176 zeigt, wie Speicherbereiche in einer logischen Tabellenbereichsadressenzuordnung zugeordnet werden.

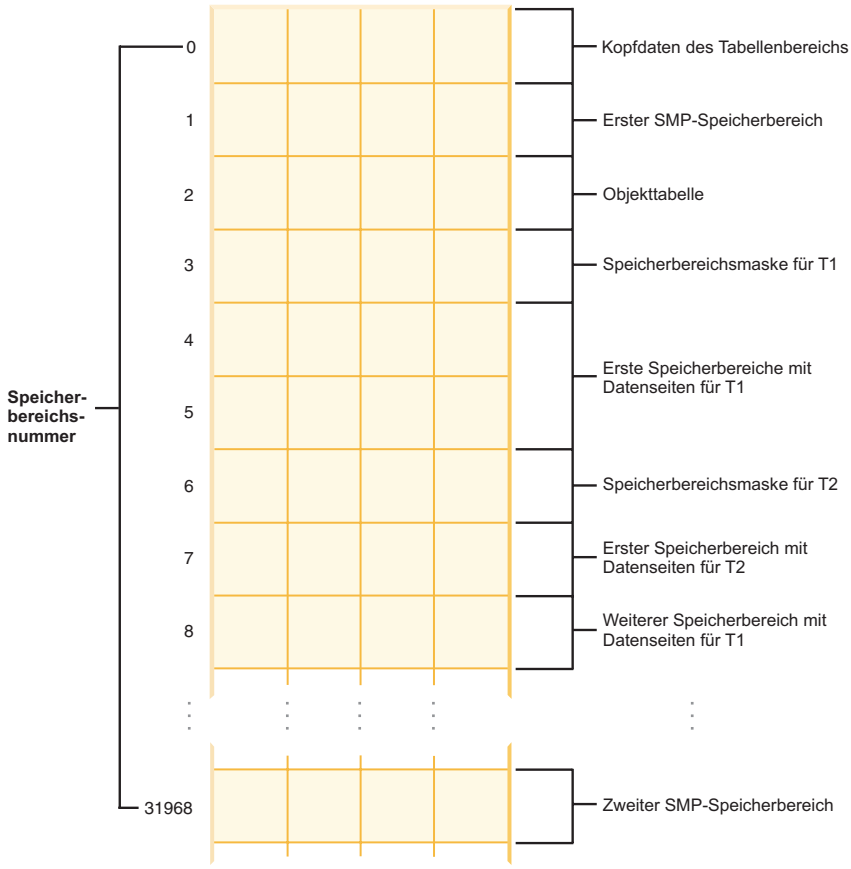


Abbildung 8. Logische Tabellenbereichsadresszuordnung

DMS-Tabellenbereichszuordnungen:

Eine Tabellenbereichszuordnung ist im Datenbankmanager eine interne Darstellung eines DMS-Tabellenbereichs, die die Konvertierung logischer Seitenpositionen in physische Seitenpositionen in einem Tabellenbereich beschreibt. Dieser Abschnitt erläutert die Nützlichkeit einer Tabellenbereichszuordnung und die Herkunft der Informationen in einer Tabellenbereichszuordnung.

In einer partitionierten Datenbank werden Seiten in einem DMS-Tabellenbereich logisch von 0 bis (N-1) durchnummeriert, wobei N die Anzahl verwendbarer Seiten im Tabellenbereich darstellt.

Die Seiten in einem Tabellenbereich werden zu Speicherbereichen gruppiert, deren Größe durch den Parameter EXTENTSIZE definiert ist. Aus Sicht der Tabellenbereichsverwaltung erfolgt jede Zuordnung von Objekten auf der Grundlage der durch EXTENTSIZE definierten Speicherbereiche. Auf diese Weise ist es möglich, dass eine Tabelle vielleicht nur die Hälfte der Seiten in einem EXTENTSIZE-Speicherbereich verwendet, jedoch wird der gesamte Speicherbereich als in Gebrauch und dem jeweiligen Objekt zugeordnet betrachtet. Standardmäßig wird ein EXTENTSIZE großer Speicherbereich zur Aufnahme der Containerkennung verwendet, wobei dieser Speicherbereich nicht zum Speichern von Daten verwendet werden kann. Wenn jedoch die Registrierdatenbankvariable DB2_USE_PAGE_CONTAINER_TAG aktiviert ist, wird nur eine Seite für die Containerkennung verwendet.

In Abb. 9 sehen Sie die logische Adressenzuordnung für einen DMS-Tabellenbereich.

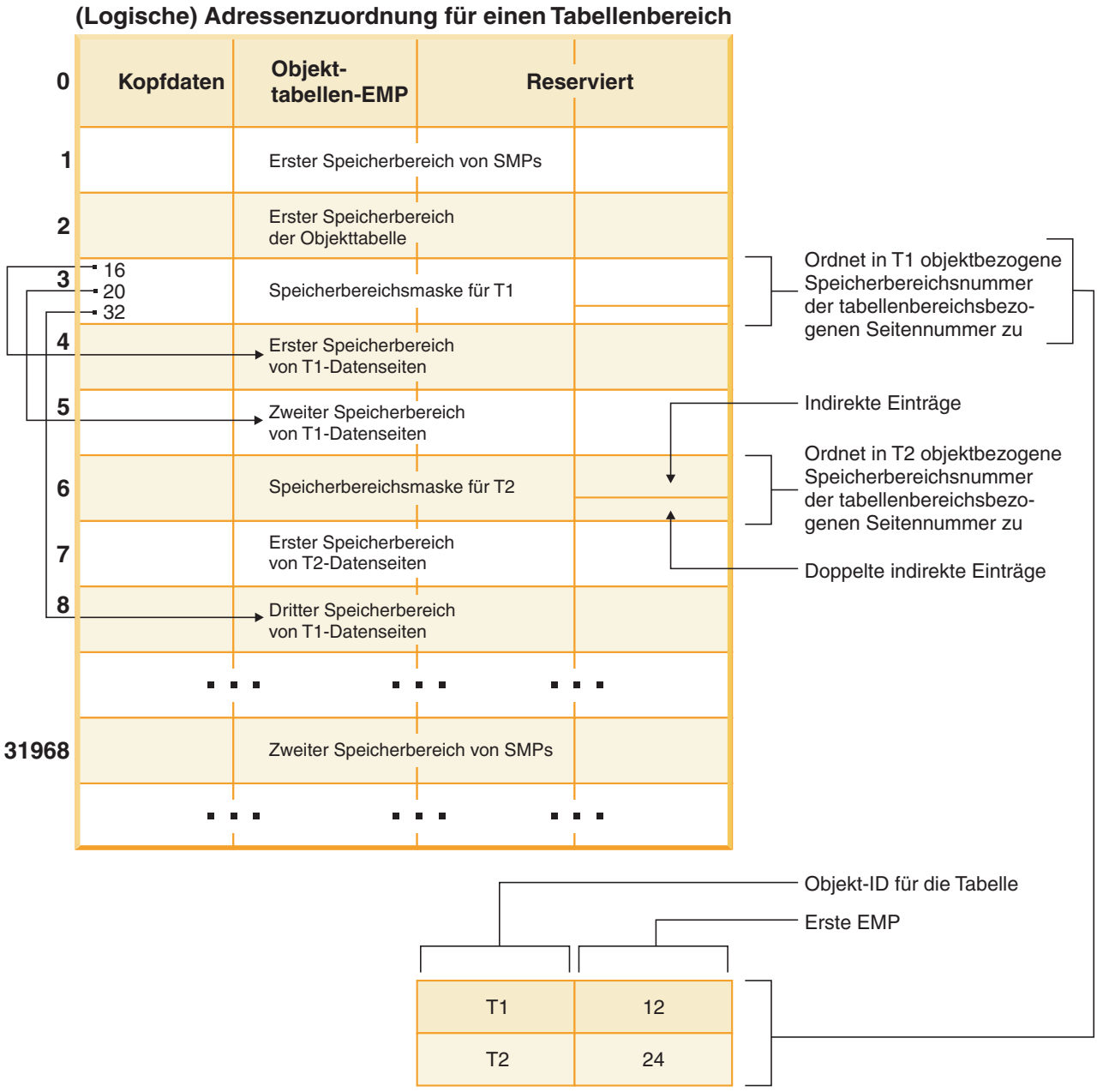


Abbildung 9. DMS-Tabellenbereiche

In der Adressenzuordnung des Tabellenbereichs gibt es zwei Typen von Zuordnungsseiten: EMP-Seiten (Extent Map Pages) und SMP-Seiten (Space Map Pages).

Die Objekttabelle ist eine interne relationale Tabelle, die eine Objektkennung der Position des ersten EMP-Speicherbereichs in der Tabelle zuordnet. Dieser EMP-Speicherbereich bildet, direkt oder indirekt, eine Zuordnung aller Speicherbereiche im betreffenden Objekt. Jede EMP enthält eine Reihe von Einträgen. Jeder Eintrag ordnet eine zum Objekt relative Speicherbereichsnummer einer zum Tabellenbereich relativen Seitennummer zu, an der sich der Objektspeicherbereich befindet.

Direkte EMP-Einträge ordnen Adressen, die zum Objekt relativ sind, direkt Adressen zu, die zum Tabellenbereich relativ sind. Die letzte EMP-Seite im ersten EMP-Speicherbereich enthält indirekte Einträge. Indirekte EMP-Einträge ordnen EMP-Seiten zu, die anschließend Zuordnungen zu Objektseiten herstellen. Die letzten 16 Einträge in der letzten EMP-Seite im ersten EMP-Speicherbereich enthalten doppelt indirekte Einträge.

Die Speicherbereiche der Adressenzuordnung für den logischen Tabellenbereich werden reihum einheitenübergreifend in den Containern gespeichert, die dem Tabellenbereich zugeordnet sind.

Da der Speicher in Containern jeweils in Form eines EXTENTSIZE großen Bereichs zugeordnet wird, werden Seiten, die nicht eine volle EXTENTSIZE-Größe bilden, nicht verwendet. Wenn Sie zum Beispiel einen 205 Seiten großen Container mit einem EXTENTSIZE-Wert 10 haben, sind ein EXTENTSIZE-Speicherbereich für die Kennung und 19 EXTENTSIZE-Speicherbereiche für Daten verfügbar. Die fünf übrigen Seiten werden verschenkt.

Wenn ein DMS-Tabellenbereich einen einzelnen Container enthält, ist die Umwandlung der logischen Seitennummer in die physische Position auf dem Datenträger ein einfacher Prozess, bei dem die Seiten 0, 1, 2 in der gleichen Reihenfolge auf dem Datenträger angeordnet werden.

Ebenfalls recht einfach ist der Prozess, wenn mehr als ein Container vorhanden ist und jeder der Container gleich groß ist. Der erste EXTENTSIZE-Speicherbereich im Tabellenbereich, der die Seiten 0 bis (EXTENTSIZE - 1) enthält, wird im ersten Container angelegt, der zweite EXTENTSIZE-Speicherbereich wird im zweiten Container angelegt usw. Nach dem letzten Container wird der Prozess wiederholt, wobei wieder mit dem ersten Container begonnen wird. Durch diesen zyklischen Prozess werden die Daten gleichmäßig verteilt.

Für Tabellenbereiche, die Container unterschiedlicher Größen enthalten, kann kein einfaches Reihumverfahren angewandt werden, da in diesem Fall der zusätzliche Speicherplatz in den größeren Containern nicht genutzt wird. An dieser Stelle kommt die Tabellenbereichszuordnung (engl. table space map) ins Spiel: Sie gibt an, wie die EXTENTSIZE-Speicherbereiche innerhalb des Tabellenbereichs positioniert sind, und stellt dadurch sicher, dass alle Speicherbereiche in den physischen Containern zur Verwendung verfügbar sind.

Anmerkung: In den folgenden Beispielen wird bei den Containergrößen die Größe der Containerkennung (container tag) nicht berücksichtigt. Die Containergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Containergrößen dar. Die Beispiele zeigen Container unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch wird empfohlen, Container gleicher Größe zu verwenden.

Beispiel 1:

In einem Tabellenbereich sind drei Container vorhanden, jeder Container enthält 80 verwendbare Seiten und der EXTENTSIZE-Wert für den Tabellenbereich beträgt 20. Jeder Container enthält daher vier EXTENTSIZE große Speicherbereiche (80 / 20) mit insgesamt zwölf Speicherbereichen. Diese Speicherbereiche (Extents) befinden sich auf dem Datenträger wie in Abb. 10 auf Seite 179 gezeigt.

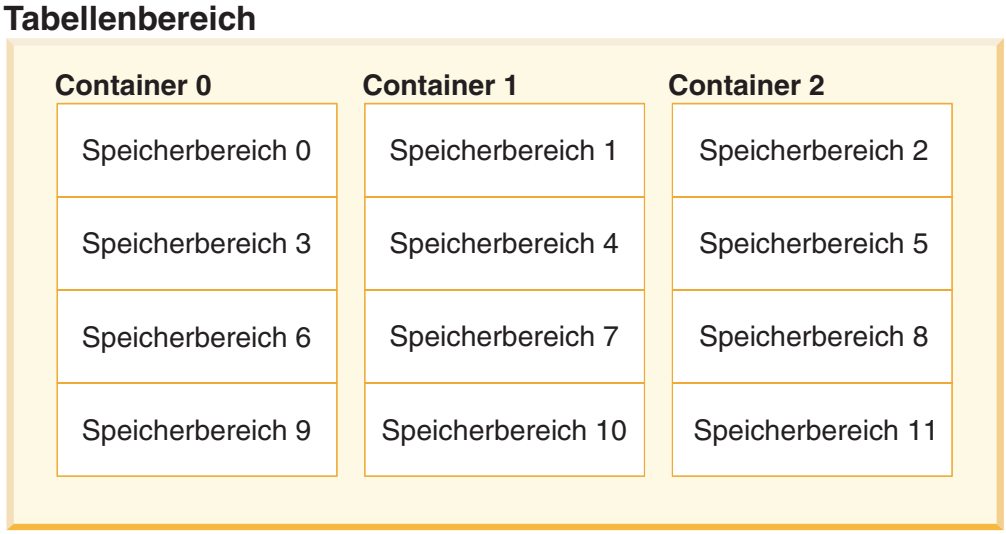


Abbildung 10. Tabellenbereich mit drei Containern und zwölf Speicherbereichen

Wenn Sie sich eine Tabellenbereichszuordnung ansehen wollen, erstellen Sie mithilfe des Snapshot Monitor eine Momentaufnahme des Tabellenbereichs. In Beispiel 1, in dem die drei Container die gleiche Größe besitzen, sieht die Tabellenbereichszuordnung wie folgt aus:

```
Range   Stripe Stripe Max      Max   Start   End         Adj.  Containers
Number  Set    Offset Extent   Page  Stripe  Stripe
  [0]    [0]    0     11       239   0       3           0  3  (0, 1, 2)
```

Ein *Bereich* (engl. *range*) ist der Teil der Zuordnung, in dem ein zusammenhängender Bereich von Stripes jeweils die gleiche Gruppe von Containern enthalten. In Beispiel 1 enthalten alle Stripes (0 - 3) die gleiche Gruppe von drei Containern (0, 1 und 2), sodass dies als ein einzelner Bereich betrachtet wird.

Die Spaltenüberschriften in der Tabellenbereichszuordnung heißen 'Range Number' (Bereichsnummer), 'Stripe Set', 'Stripe Offset', 'Maximum extent number addressed by the range' (höchste Speicherbereichsnummer, die durch den Bereich adressiert wird), 'Maximum page number addressed by the range' (höchste Seitennummer, die durch den Bereich adressiert wird), 'Start Stripe' (Anfangsstripe), 'End Stripe' (Endstripe), 'Range adjustment' (Bereichsanpassung) und 'Container list' (Containerliste). Diese Namen werden in Beispiel 2 detaillierter beschrieben.

Dieser Tabellenbereich kann auch wie in Abb. 11 auf Seite 180 dargestellt werden, wobei jede vertikale Linie einem Container entspricht und jede horizontale Linie als *Stripe* (einheitenübergreifend gespeicherter Datenblock) bezeichnet wird. Jede Zellennummer entspricht einem EXTENTSIZE großen Speicherbereich (Extent).

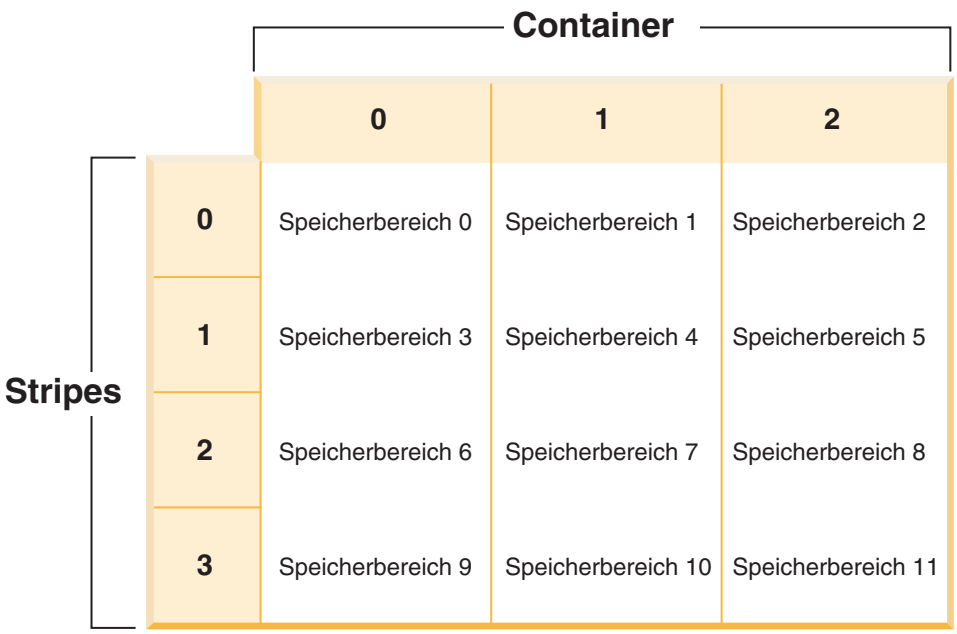


Abbildung 11. Tabellenbereich mit drei Containern und zwölf Speicherbereichen, Stripes hervorgehoben

Beispiel 2:

Im Tabellenbereich sind zwei Container vorhanden: der erste ist 100 Seiten groß, der zweite 50 Seiten, und EXTENTSIZE definiert eine Größe von 25 Seiten. Dies bedeutet, dass der erste Container vier EXTENTSIZE große Speicherbereiche und der zweite Container zwei solche Speicherbereiche besitzt. Der Tabellenbereich lässt sich wie in Abb. 12 darstellen.

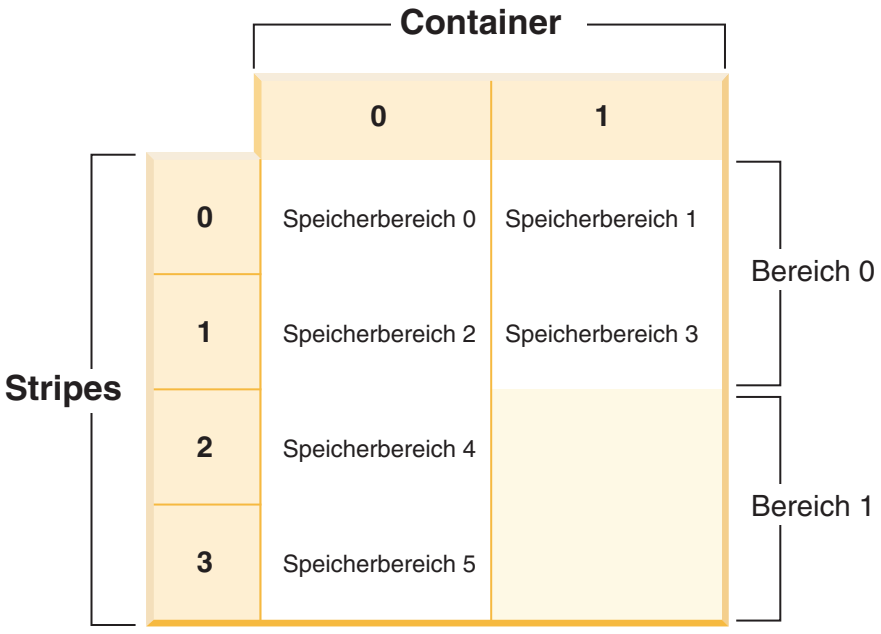


Abbildung 12. Tabellenbereich mit zwei Containern, Bereiche hervorgehoben

Stripes 0 und 1 enthalten beide Container (0 und 1), jedoch enthalten die Stripes 2 und 3 nur den ersten Container (0). Jede dieser Gruppen von Stripes wird als

Bereich (range) bezeichnet. Die Tabellenbereichszuordnung, die in der Momentaufnahme eines Tabellenbereichs gezeigt wird, sieht folgendermaßen aus:

```
Range   Stripe Stripe Max      Max   Start   End      Adj.  Containers
Number  Set    Offset Extent   Page  Stripe  Stripe
  [0]    [0]    0     3         99   0       1        0  2  (0, 1)
  [1]    [0]    0     5        149   2       3        0  1  (0)
```

Im ersten Bereich (Range) befinden sich vier EXTENTSIZE große Speicherbereiche (Extents). Daher ist 3 die höchste Speicherbereichsnummer (Max Extent), die in diesem Bereich adressiert wird. Jeder Speicherbereich ist 25 Seiten groß, sodass sich im ersten Bereich 100 Seiten befinden. Da die Seitennummerierung ebenfalls bei 0 beginnt, ist 99 die höchste Seitennummer (Max Page), die in diesem Bereich adressiert wird. Der erste Stripe (Start Stripe) in diesem Bereich ist 0 und der letzte Stripe (End Stripe) im Bereich ist Stripe 1. Es gibt zwei Container in diesem Bereich, 0 und 1. Das Stripe-Offset ist der erste Stripe im Stripe-Set, in diesem Fall 0, weil nur ein Stripe-Set vorhanden ist. Die Bereichsanpassung (Adj.) ist ein Offsetwert, der verwendet wird, wenn Daten in einem Tabellenbereich neu ausgeglichen werden. (Ein Neuausgleich kann stattfinden, wenn in einem Tabellenbereich Speicherplatz hinzugefügt oder gelöscht wird.) Wenn kein Neuausgleich stattfindet, ist dieser Wert immer 0.

Im zweiten Bereich (Range) befinden sich zwei EXTENTSIZE große Speicherbereiche (Extents), und da 3 die höchste Speicherbereichsnummer ist, die im vorigen Bereich adressiert wurde, ist 5 die höchste Speicherbereichsnummer, die in diesem Bereich adressiert wird. Im zweiten Bereich befinden sich 50 Seiten (2 Speicherbereiche * 25 Seiten), und da 99 die höchste Seitennummer ist, die im vorigen Bereich adressiert wird, ist 149 nun die höchste Seitennummer, die in diesem Bereich adressiert wird. Dieser Bereich beginnt bei Stripe 3 und endet bei Stripe 3.

Tabellenbereiche mit dynamischem Speicher

Bei der Erstellung eines Tabellenbereichs in einer Datenbank, für die der dynamische Speicher nicht aktiviert ist, müssen Sie eine der Klauseln MANAGED BY SYSTEM oder MANAGED BY DATABASE angeben. . Durch die Verwendung dieser Klauseln wird ein vom System verwalteter SMS-Tabellenbereich bzw. ein von der Datenbank verwalteter DMS-Tabellenbereich erstellt. In beiden Fällen müssen Sie eine explizite Liste von Containern angeben.

Wenn für eine Datenbank der dynamische Speicher aktiviert ist, sind weitere Möglichkeiten verfügbar: Sie können die Klausel MANAGED BY AUTOMATIC STORAGE angeben oder die MANAGED BY-Klausel weglassen (was die Verwendung des dynamischen Speichers impliziert). . In diesem Fall brauchen Sie keine Containerdefinitionen anzugeben, weil der Datenbankmanager die Container automatisch zuweist.

Die folgenden Beispiele zeigen einige Anweisungen zur Erstellung von Tabellenbereichen mit dynamischem Speicher:

```
CREATE TABLESPACE TS1
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
CREATE TEMPORARY TABLESPACE TEMPTS
CREATE USER TEMPORARY TABLESPACE USRTMP MANAGED BY AUTOMATIC STORAGE
CREATE LONG TABLESPACE LONGTS
```

Obwohl der Tabellenbereichstyp mit dynamischem Speicher ein anderer Tabellenbereichstyp zu sein scheint, handelt es sich tatsächlich nur um eine Erweiterung

der bereits vorhandenen Tabellenbereichstypen SMS und DMS. Wenn Sie einen regulären (REGULAR) oder großen (LARGE) Tabellenbereich erstellen, wird er als DMS-Tabellenbereich mit Dateicontainern erstellt. Wenn Sie einen Tabellenbereich für temporäre Systemtabellen erstellen, wird er als SMS-Tabellenbereich mit Verzeichniscontainern erstellt.

Anmerkung: In zukünftigen Versionen des Datenbankmanagers wird dieses Verhalten möglicherweise geändert.

Die diesen Containern zugeordneten Namen haben folgendes Format:

speicherpfad/*instanzname*/NODE####/*datenbankname*/T#######/C#######.*ERW*

Dabei gilt:

speicherpfad
: Ein der Datenbank zugeordneter Speicherpfad

instanzname
: Die Instanz, unter der die Datenbank erstellt wurde

datenbankname
: Der Name der Datenbank

NODE####
: Die Datenbankpartitionsnummer (z. B. NODE0000)

T#######
: Die Tabellenbereichs-ID (z. B. T0000003)

C#######
: Die Container-ID (z. B. C0000012)

ERW
: Eine Erweiterung basierend auf dem Typ der zu speichernden Daten:

  CAT Systemkatalogtabellenbereich

  TMP Tabellenbereich für temporäre Systemtabellen

  UTM Tabellenbereich für temporäre Benutzertabellen

  USR Benutzertabellenbereich oder regulärer Tabellenbereich

  LRG LOB-Tabellenbereich

Unterschiede zwischen regulären und großen Tabellenbereichen mit dynamischem Speicher und DMS-Tabellenbereichen

Reguläre und große (LARGE) Tabellenbereiche mit dynamischem Speicher werden als DMS-Tabellenbereiche erstellt, für die alle Regeln und Funktionsweisen von DMS-Tabellenbereichen gelten. In der Verwaltung des Speichers gibt es jedoch Unterschiede, die in der folgenden Tabelle erläutert werden:

Tabelle 45. Verwaltungsunterschiede zwischen Tabellenbereichen ohne dynamischen Speicher und mit dynamischem Speicher

Nicht dynamischer Speicher	Dynamischer Speicher
Sie müssen beim Erstellen des Tabellenbereichs explizit eine Liste von Containern angeben.	Sie können keine Liste von Containern beim Erstellen des Tabellenbereichs angeben, da der Datenbankmanager Container automatisch zuweist und zuordnet.

Tabelle 45. Verwaltungsunterschiede zwischen Tabellenbereichen ohne dynamischen Speicher und mit dynamischem Speicher (Forts.)

Nicht dynamischer Speicher	Dynamischer Speicher
Die Funktion zur automatischen Größenänderung ist standardmäßig inaktiviert (AUTORESIZE ist NO).	Die Funktion zur automatischen Größenänderung ist standardmäßig aktiviert (AUTORESIZE ist YES).
Sie können die Klausel INITIALSIZE nicht verwenden, um die Anfangsgröße für den Tabellenbereich anzugeben.	Sie können die Klausel INITIALSIZE verwenden, um die Anfangsgröße für den Tabellenbereich anzugeben.
Sie können Containeroperationen mit der Anweisung ALTER TABLESPACE (unter Angabe von ADD, DROP, BEGIN NEW STRIPE SET usw.) ausführen.	Sie können keine Containeroperationen ausführen, da der Datenbankmanager den Speicherbereich verwaltet.
Sie können eine umgeleitete Restoreoperation verwenden, um die dem Tabellenbereich zugeordneten Container erneut zu definieren.	Sie können keine umgeleitete Restoreoperation dazu verwenden, die dem Tabellenbereich zugeordneten Container erneut zu definieren, da der Datenbankmanager den Speicherbereich verwaltet.

Wie in obiger Tabelle aufgeführt, können Sie beim Erstellen eines regulären oder großen Tabellenbereichs mit dynamischem Speicher die Anfangsgröße in der Anweisung CREATE TABLESPACE angeben. Beispiel:

```
CREATE TABLESPACE TS1 INITIALSIZE 100 M
```

Wenn Sie keine Anfangsgröße angeben, verwendet der Datenbankmanager den Standardwert von 32 MB.

Zur Erstellung eines Tabellenbereichs mit einer bestimmten Größe erstellt der Datenbankmanager Dateicontainer in den Speicherpfaden. Wenn die Verteilung des Speicherplatzes in den Pfaden ungleichmäßig ist, werden Container möglicherweise mit unterschiedlichen Größen erstellt. Daher ist es wichtig, dass in allen Speicherpfaden eine annähernd gleiche Menge an freiem Speicherplatz zur Verfügung steht.

Wenn Sie die Funktion zur automatischen Größenänderung für den Tabellenbereich aktivieren, erweitert der Datenbankmanager entsprechend dem in diesem Tabellenbereich belegten Speicherplatz automatisch die vorhandenen Container und fügt neue hinzu (durch Stripe-Sets). Weder beim Erweitern noch beim Hinzufügen von Containern findet eine Umverteilung der Daten statt.

Tabellenbereiche des Typs TEMPORARY

Temporäre Systemtabellenbereiche enthalten temporäre Daten, die der Datenbankmanager zur Ausführung von Operationen wie Sortierungen und Joins benötigt.

Diese Typen von Operationen erfordern zusätzlichen Speicher zur Verarbeitung der Ergebnismengen. Eine Datenbank muss über mindestens einen Tabellenbereich für temporäre Systemtabellen verfügen, der dieselbe Seitengröße aufweist wie der Katalogtabellenbereich; standardmäßig wird ein Tabellenbereich für temporäre Systemtabellen mit der Bezeichnung TEMPSPACE1 zum Zeitpunkt der Datenbankerstellung generiert. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMTEMPGROUP.

Temporäre Benutzertabellenbereiche enthalten temporäre Daten aus Tabellen, die mit einer Anweisung DECLARE GLOBAL TEMPORARY TABLE erstellt wurden.

Um die Definition deklarierter temporärer Tabellen zu ermöglichen, muss mindestens ein temporärer Benutzertabellenbereich mit den entsprechenden USE-Zugriffsrechten erstellt werden. USE-Zugriffsrechte werden mithilfe der Anweisung GRANT erteilt. Ein temporärer Benutzertabellenbereich wird *nicht* standardmäßig bei der Erstellung einer Datenbank erstellt.

Es wird empfohlen, einen einzelnen temporären Tabellenbereich zu definieren, dessen Seitengröße der Seitengröße entspricht, die in den meisten Benutzertabellenbereichen verwendet wird. Dies sollte für typische Umgebungen und Auslastungen geeignet sein. Es kann jedoch vorteilhaft sein, mit unterschiedlichen Konfigurationen für temporäre Tabellenbereiche und Auslastungen zu experimentieren. Sie sollten die folgenden Punkte beachten:

- Auf temporäre Tabellen wird meist gruppenweise und sequenziell zugegriffen. Das heißt, eine Gruppe von Zeilen wird eingefügt oder eine Gruppe sequenzieller Zeilen wird abgerufen. Daher führt eine größere Seitengröße in der Regel zu einer besseren Leistung, da weniger Anforderungen logischer und physischer Seiten erforderlich sind, um eine bestimmte Datenmenge einzulesen.
- Wenn Sie mithilfe eines temporären Tabellenbereichs eine Tabelle reorganisieren, muss die Seitengröße des temporären Tabellenbereichs mit der Seitengröße der Tabelle übereinstimmen. Deshalb sollten Sie sicherstellen, dass temporäre Tabellenbereiche vorhanden sind, die für jede Seitengröße definiert sind, die von vorhandenen Tabellen verwendet wird, die Sie vielleicht mithilfe eines temporären Tabellenbereichs reorganisieren.

  Sie können eine Reorganisation auch ohne temporären Tabellenbereich ausführen, indem Sie die Tabelle direkt im selben Tabellenbereich reorganisieren. Natürlich setzt diese Art der Reorganisation voraus, dass im Tabellenbereich bzw. in den Tabellenbereichen der Tabelle zusätzlicher Speicherbereich für den Reorganisationsprozess vorhanden ist.
- Wenn Sie SMS-Tabellenbereiche für temporäre Systemtabellen verwenden, empfiehlt sich gegebenenfalls die Verwendung der Registrierdatenbankvariablen DB2_SMS_TRUNC_TMPTABLE_THRESH. Durch Löschen werden die für die temporären Systemtabellen erstellten Dateien auf die Größe 0 abgeschnitten. Die Registrierdatenbankvariable DB2_SMS_TRUNC_TMPTABLE_THRESH kann dazu verwendet werden, einen Zugriff auf Dateisysteme zu umgehen und die Dateien potenziell bei einer Größe ungleich null zu belassen, um den Leistungsaufwand zu vermeiden, der mit wiederholten Erweiterungen und Verkürzungen der Dateien verbunden ist.
- Im Allgemeinen wählt das Optimierungsprogramm, wenn temporäre Tabellenbereiche mit unterschiedlichen Seitengrößen vorhanden sind, den temporären Tabellenbereich aus, dessen Pufferpool die größte Anzahl von Zeilen aufnehmen kann (d. h. in den meisten Fällen der größte Pufferpool). In solchen Fällen empfiehlt es sich, einem der temporären Tabellenbereiche einen großen Pufferpool und den übrigen einen kleineren Pufferpool zuzuordnen. Eine solche Pufferpoolzuordnung hilft bei der Sicherstellung einer effizienten Auslastung des Hauptspeichers. Wenn z. B. Ihr Katalogtabellenbereich 4-KB-Seiten verwendet und die übrigen Tabellenbereiche 8-KB-Seiten, kann sich folgende Konfiguration für temporäre Tabellenbereiche am besten eignen: ein einzelner temporärer 8-KB-Tabellenbereich mit einem großen Pufferpool und ein einzelner 4-KB-Tabellenbereich mit einem kleinen Pufferpool.
- Es bringt im Allgemeinen keinen Vorteil, mehr als einen temporären Tabellenbereich von jeder Seitengröße zu definieren.

Sicherstellen der erforderlichen Seitengrößen für temporäre Tabellenbereiche auf dem System:

Durch die Verwendung höherer RIDs (Satzkennungen) werden die Zeilen in den Ergebnismengen zu Abfragen und positionierten Aktualisierungen länger. Wenn die Zeilengröße in Ihren Ergebnismengen der zulässigen maximalen Zeilenlänge für Ihre vorhandenen temporären Tabellenbereiche auf dem System nahe kommt, müssen Sie möglicherweise einen Systemtabellenbereich für temporäre Tabellen mit einer größeren Seitengröße erstellen.

Voraussetzung

Stellen Sie sicher, dass Sie über die Berechtigung SYSCTRL oder SYSADM verfügen, wenn Sie einen Systemtabellenbereich für temporäre Tabellen erstellen müssen.

Vorgehensweise

Gehen Sie wie folgt vor, um sicherzustellen, dass die maximale Seitengröße für Ihren Systemtabellenbereich für temporäre Tabellen für Ihre Abfragen und positionierten Aktualisierungen ausreicht:

1. Ermitteln Sie die maximale Zeilengröße in Ihren Ergebnismengen zu Abfragen und positionierten Aktualisierungen. Überwachen Sie die Abfragen, oder ermitteln Sie die maximale Zeilengröße mit der DDL-Anweisung, die Sie zum Erstellen Ihrer Tabellen verwendet haben.
2. Listen Sie Ihre Tabellenbereiche mit dem Befehl LIST TABLESPACES, wie im folgenden Beispiel angegeben, auf:

```
     db2 LIST TABLESPACES SHOW DETAIL...
 Tabellenbereichs-ID                  = 1
 Name                                 = TEMPSPACE1
 Typ                                  = Vom System verwalteter Bereich
 Inhalt                               = Temporäre Systemdaten
 Status                               = 0x0000
     Detaillierte Erläuterung:
         Normal
 Seiten insgesamt                     = 10
 Verwendbare Seiten                   = 10
 Verwendete Seiten                    = 10
 Freie Seiten                         = Nicht zutreffend
 Obere Grenze (Seiten)                = Nicht zutreffend
 Seitengröße (Byte)                   = 4096
 Speicherbereichsgröße (Seiten)       = 32
 Vorablesezugriffsgröße (Seiten)      = 320
 Anzahl Container                     = 10
...
```

   Sie können die temporären Tabellenbereiche auf dem System in der Ausgabe an dem Wert 'Temporäre Systemdaten' im Inhaltsfeld der Ausgabe erkennen. Achten Sie auf die Seitengröße für die einzelnen Systemtabellenbereiche für temporäre Tabellen und die Seitengröße der Tabellenbereiche, in denen die in den Abfragen und Aktualisierungen angegebenen Tabellen erstellt wurden.
3. Überprüfen Sie, ob die Seitengröße für den Systemtabellenbereich für temporäre Tabellen für die größte Zeilengröße in Ihren Ergebnismengen ausreicht:

```
maximale_zeilengröße > maximale_zeilenlänge - 8 Byte (strukturbedingter
                                                 Überhang bei Einzelpartition)
maximale_zeilengröße > maximale_zeilenlänge - 16 Byte (strukturbedingter
                                                 Überhang bei DPF)
```

   'maximale_zeilengröße' gibt die maximale Zeilengröße für Ihre Ergebnismengen an und 'maximale_zeilenlänge' die maximale zulässige Zeilenlänge entspre-

chend der größten Seitengröße bei allen temporären Tabellenbereichen auf dem System. Lesen Sie die Informationen über SQL- und XML-Einschränkungen im Handbuch *SQL Reference, Volume 1*, um die maximale Zeilenlänge für die Seitengrößen der einzelnen Tabellenbereiche zu ermitteln.

Wenn die maximale Zeilengröße unter dem errechneten Wert liegt, werden Ihre Abfragen wie in DB2 UDB Version 8 ausgeführt, und es sind keine weiteren Maßnahmen erforderlich.

4. Erstellen Sie einen temporären Tabellenbereich auf Ihrem System, dessen Seitengröße mindestens der zweifachen Seitengröße des Tabellenbereichs entspricht, in dem die Tabellen erstellt wurden, sofern Sie nicht bereits über einen temporären Tabellenbereich mit dieser Seitengröße verfügen. Wenn Sie z. B. unter Windows eine Tabelle in einem Tabellenbereich mit einer Seitengröße von 4 KB erstellt haben, erstellen Sie nun einen zusätzlichen temporären Tabellenbereich auf dem System mit einer Seitengröße von 8 KB:

```
db2 CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
                PAGESIZE 8K
                MANAGED BY SYSTEM
                USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

Wenn die Seitengröße des Tabellenbereichs 32 KB beträgt, können Sie die über Ihre Abfragen ausgewählten Daten reduzieren oder Abfragen so aufspalten, dass die Seite des Systemtabellenbereichs für temporäre Tabellen ausreicht. Statt z. B. alle Spalten einer Tabelle auszuwählen, können Sie nur die Spalten auswählen, die Sie tatsächlich benötigen, oder eine Unterzeichenfolge bestimmter Spalten, um zu vermeiden, dass die Seitengröße überschritten wird.

SMS- und DMS-Tabellenbereiche im Vergleich

Bei der Entscheidung, welcher Tabellenbereichstyp zum Speichern der Daten verwendet werden soll, sollten Sie das Pro und Kontra gut abwägen.

Vorteile eines SMS-Tabellenbereichs:

- Der Speicher wird vom System erst dann zugeordnet, wenn er benötigt wird.
- Beim Erstellen eines Tabellenbereichs sind weniger vorbereitende Arbeiten erforderlich, weil Sie keine Container vordefinieren müssen.
- Indizes, die für bereichspartitionierte Daten erstellt werden, können in einem anderen Tabellenbereich als die Tabellendaten gespeichert werden.

Vorteile eines DMS-Tabellenbereichs:

- Die Größe eines Tabellenbereichs kann durch Hinzufügen oder Erweitern von Containern erhöht werden (mit der Anweisung ALTER TABLESPACE). Vorhandene Daten können automatisch auf die neue Gruppe von Containern verteilt werden, um die optimale E/A-Effizienz zu erhalten.
- Eine Tabelle kann nach dem Typ der zu speichernden Daten auf mehrere Tabellenbereiche verteilt werden:
  - LF-Daten (LF = Long Field, Langfeld) und LOB-Daten (LOB = Large Object, großes Objekt)
  - Indizes
  - Reguläre Tabellendaten

  Vielleicht sollen die Tabellendaten zur Erhöhung der Leistung oder zur Vergrößerung der für eine Tabelle gespeicherten Datenmenge getrennt gehalten werden. Wenn Sie zum Beispiel große Tabellenbereiche (LARGE) mit einer Seitengröße von 4 KB verwenden, können Sie eine Tabelle mit 2 TB an regulären Tabellendaten, einen separaten Tabellenbereich mit 2 TB an Indexdaten und einen weiteren separaten Tabellenbereich mit 2 TB an Langfelddaten haben. Wenn diese drei Typen von Daten stattdessen in nur einem Tabellenbereich

gespeichert würden, wäre der Gesamtspeicherplatz auf 2 TB begrenzt. Die Verwendung größerer Seitengrößen ermöglicht Ihnen zudem die Speicherung von mehr Daten. Die vollständige Liste der Begrenzungen für Seitengrößen des Datenbankmanagers finden Sie über die zugehörigen Links.
- Indizes, die für bereichspartitionierte Daten erstellt werden, können in einem anderen Tabellenbereich als die Tabellendaten gespeichert werden.
- Es ist möglich, die Speicherposition der Daten auf der Platte zu steuern, sofern das Betriebssystem dies zulässt.
- Im Allgemeinen gilt, dass sich mit einer gut abgestimmten Gruppe von DMS-Tabellenbereichen eine bessere Leistung erzielen lässt als mit SMS-Tabellenbereichen.

Anmerkung: Bei leistungsempfindlichen Anwendungen, insbesondere Anwendungen mit einer großen Anzahl an Einfügeoperationen, ist die Verwendung von DMS-Tabellenbereichen empfehlenswert.

Darüber hinaus kann sich die Platzierung von Daten bei beiden Arten von Tabellenbereichen unterscheiden. Betrachten Sie beispielsweise den Bedarf an effizienten Tabellensuchoperationen: Es ist wichtig, dass die Seiten in einem Speicherbereich physisch aufeinander folgen. Bei SMS entscheidet das Dateisystem des Betriebssystems, wo die logischen Dateiseiten physisch abgelegt werden. Die Seiten können aufeinander folgend zugeordnet werden. Dies hängt vom Umfang anderer Aktivitäten im Dateisystem sowie von dem Algorithmus ab, der zum Bestimmen der Platzierung verwendet wird. Bei DMS jedoch kann der Datenbankmanager sicherstellen, dass die Seiten physisch aufeinander folgen, da er direkt mit dem Plattendatenträger interagiert.

Generell lassen sich kleine Datenbanken, die von einem kleinen Personenkreis genutzt werden, am einfachsten mit SMS-Tabellenbereichen verwalten. Andererseits sollten Sie für umfangreiche, wachsende Datenbanken SMS-Tabellenbereiche nur für temporäre Tabellenbereiche und den Katalogtabellenbereich sowie getrennte DMS-Tabellenbereiche mit mehreren Containern für jede Tabelle verwenden. Außerdem ist es wahrscheinlich sinnvoll, Langfelddaten (LF-Daten) und Indizes in eigenen Tabellenbereichen zu speichern.

Wenn Sie sich entschließen, DMS-Tabellenbereiche mit Einheitencontainern zu verwenden, müssen Sie bereit sein, Ihre Umgebung zu optimieren und zu verwalten.

Überlegungen zur Auslastung beim Entwerfen von SMS- und DMS-Tabellenbereichen

Der primäre Typ der Auslastung, die vom Datenbankmanager in Ihrer Umgebung verwaltet wird, kann Ihre Wahl des zu verwendenden Typs von Tabellenbereich und der anzugebenden Seitengröße beeinflussen.

Eine OLTP-Auslastung (Onlinetransaktionsverarbeitung) ist durch Transaktionen charakterisiert, die einen wahlfreien Zugriff auf Daten benötigen, häufige Einfüge- oder Aktualisierungsaktivitäten verursachen und Abfragen enthalten, die in der Regel nur kleine Datenmengen zurückliefern. In Anbetracht dessen, dass der Zugriff wahlfrei nur auf eine bzw. wenige Seiten erfolgt, ist ein Vorablesezugriff weniger wahrscheinlich.

In diesem Fall zeigen DMS-Tabellenbereiche mit Einheitencontainern die beste Leistung. DMS-Tabellenbereiche mit Dateicontainern oder SMS-Tabellenbereiche sind ebenfalls sinnvolle Möglichkeiten für eine OLTP-Auslastung, wenn es nicht auf maximale Leistung ankommt. Beachten Sie, dass die Verwendung von DMS-Tabellenbereichen mit Dateicontainern (wobei FILE SYSTEM CACHING auf OFF

gesetzt ist) auf einer Ebene durchgeführt werden kann, die mit unformatierten DMS-Tabellenbereichscontainern vergleichbar ist. Wenn nur wenige oder gar keine sequenziellen E/A-Operationen zu erwarten sind, spielen die Werte der Parameter EXTENTSIZE und PREFETCHSIZE in der Anweisung CREATE TABLESPACE keine wichtige Rolle für die E/A-Effizienz. Wichtig ist jedoch, eine ausreichende Anzahl von Seitenlöschfunktionen über den Konfigurationsparameter *chngpgs_thresh* festzulegen.

Eine Abfrageauslastung ist durch Transaktionen charakterisiert, die einen sequenziellen oder teilweise sequenziellen Zugriff auf Daten benötigen und in der Regel große Datenmengen zurückliefern. Ein DMS-Tabellenbereich mit mehreren Einheitencontainern (wobei sich jeder Container auf einer separaten Platte befindet) bietet das größte Potenzial für einen effizienten parallelen Vorablesezugriff. Der Wert des Parameters PREFETCHSIZE in der Anweisung CREATE TABLESPACE sollte das Produkt aus dem Wert des Parameters EXTENTSIZE multipliziert mit der Anzahl der Einheitencontainer sein. Alternativ dazu können Sie eine Vorablesezugriffsgröße von -1 angeben, und der Datenbankmanager wählt automatisch eine entsprechende Vorablesezugriffsgröße aus. Dadurch kann der Datenbankmanager aus allen Containern parallel vorablesen. Wenn sich die Anzahl von Containern ändert oder der Vorablesezugriff aus einem Grund verstärkt oder verringert werden muss, kann der Wert für PREFETCHSIZE über die Anweisung ALTER TABLESPACE entsprechend geändert werden.

Eine sinnvolle Alternative für eine Abfrageauslastung ist die Verwendung von Dateien, wenn das Dateisystem über eine eigene Vorablesefunktion verfügt. Die Dateien können entweder zu DMS-Tabellenbereichen mit Dateicontainern gehören oder Dateien für SMS-Tabellenbereiche sein. Beachten Sie, dass Sie bei Verwendung von SMS sicherstellen müssen, dass die Verzeichniscontainer getrennten physischen Datenträgern zugeordnet sind, um E/A-Parallelität zu erreichen.

Das Ziel für eine gemischte Auslastung besteht darin, einzelne E/A-Anforderungen so effizient wie möglich für OLTP-Auslastungen zu machen und die Effizienz paralleler E/A-Operationen für Abfrageauslastungen zu maximieren.

Beim Ermitteln der Seitengröße für einen Tabellenbereich sind folgende Überlegungen zu berücksichtigen:

- Für OLTP-Anwendungen, die wahlfreie Lese- und Schreiboperationen durchführen, ist eine geringere Seitengröße empfehlenswert, weil dabei kein Pufferspeicher durch unerwünschte Zeilen verschwendet wird.
- Für DSS-Anwendungen (Decision Support System), die jeweils auf eine große Anzahl aufeinander folgender Zeilen gleichzeitig zugreifen, sind größere Seiten empfehlenswert, weil dadurch weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen.
- Durch größere Seiten können Sie möglicherweise die Zahl der Indexstufen reduzieren.
- Größere Seiten unterstützen längere Zeilen.
- Auf 4-KB-Standardseiten sind Tabellen auf 500 Spalten begrenzt, während auf größeren Seiten (8 KB, 16 KB und 32 KB) 1.012 Spalten unterstützt werden.
- Die Maximalgröße des Tabellenbereichs ist proportional zur Seitengröße des Tabellenbereichs.

Hinweise zu SMS- und DMS-Einheiten

Bei der Entscheidung über die Verwendung von Dateisystemdateien bzw. Einheiten für Tabellenbereichscontainer sind zwei Dinge zu berücksichtigen: das Puffern von Daten und die Verwendung von LOB- bzw. LONG-Daten.

- **Puffern von Daten**

  Vom Plattenspeicher gelesene Tabellendaten sind in der Regel im Pufferpool der Datenbank verfügbar. In einigen Fällen kann es geschehen, dass eine Datenseite aus dem Pufferpool entfernt wird, bevor sie von der Anwendung verwendet wurde. Dies ist insbesondere dann möglich, wenn der Pufferpoolbereich für andere Datenseiten benötigt wird. Für Tabellenbereiche, die mit SMS- oder DMS-Dateicontainern arbeiten, kann die oben beschriebene Cachefunktion des Dateisystems die E/A-Operationen überflüssig machen, die in einem solchen Fall ansonsten anfallen.

  Tabellenbereiche, die DMS-Einheitencontainer verwenden, verwenden das Dateisystem oder den Cache des Dateisystems nicht. Daher könnten Sie den Pufferpool der Datenbank vergrößern und den Cache des Dateisystems verkleinern, um den Umstand auszugleichen, dass DMS-Tabellenbereiche, die Einheitencontainer verwenden, keine doppelte Pufferung nutzen.

  Wenn Monitortools auf Systemebene zeigen, dass das Aufkommen an E/A-Operationen für einen DMS-Tabellenbereich im Vergleich zu einem äquivalenten SMS-Tabellenbereich höher liegt, kann sich dieser Unterschied durch die doppelte Pufferung erklären.

- **Verwenden von LOB- oder LONG-Daten**

  Wenn eine Anwendung LOB- oder LONG-Daten abruft, verwendet der Datenbankmanager keine seiner Puffer, um die Daten zwischenzuspeichern. Jedes Mal, wenn eine Anwendung eine dieser Seiten benötigt, muss der Datenbankmanager sie vom Plattenspeicher abrufen. Wenn LOB- oder LONG-Daten jedoch in SMS- oder DMS-Dateicontainern gespeichert werden, kann die Pufferung durch den Dateisystemcache erfolgen und dadurch eine bessere Leistung erreicht werden.

  Da die Systemkataloge einige LOB-Spalten enthalten, sollten Sie sie in SMS-Tabellenbereichen oder in DMS-Dateitabellenbereichen speichern.

Überlegungen zur Auswahl von Tabellenbereichen für Tabellen

Beim Ermitteln der Vorgehensweise beim Zuordnen von Tabellen zu Tabellenbereichen sollten Sie die Verteilung Ihrer Tabellen, die Menge der Daten und den Typ der Daten in der Tabelle sowie verwaltungstechnische Fragen berücksichtigen.

Verteilung Ihrer Tabellen

Sie müssen zumindest sicherstellen, dass sich der Tabellenbereich, den sie auswählen, in einer Datenbankpartitionsgruppe mit der gewünschten Verteilung befindet.

Menge der Daten in der Tabelle

Wenn Sie planen, viele kleine Tabellen in einem Tabellenbereich zu speichern, sollten Sie dazu die Verwendung eines SMS-Tabellenbereichs in Betracht ziehen. Die Vorteile von DMS-Tabellenbereichen im Hinblick auf die Effizienz bei E/A-Operationen und Speicherbereichsverwaltung sind bei kleinen Tabellen nicht so bedeutsam. Die Vorteile von SMS-Tabellenbereichen (und nur bei Bedarf) sind bei kleinen Tabellen attraktiver. Wenn eine Ihrer Tabellen größer ist oder Sie einen schnelleren Zugriff auf die Daten in den Tabellen benötigen, sollte ein DMS-Tabellenbereich mit einem kleinen Wert für EXTENTSIZE in Betracht gezogen werden.

Eventuell empfiehlt es sich, einen separaten Tabellenbereich für jede umfangreiche Tabelle zu verwenden und kleine Tabellen gemeinsam in einem einzigen Tabellenbereich anzulegen. Diese Trennung ermöglicht Ihnen, anhand der Auslastung des Tabellenbereichs einen geeigneten Wert für den Parameter EXTENTSIZE auszuwählen.

Typ der Daten in der Tabelle

Sie könnten beispielsweise über Tabellen mit alten Daten verfügen, die relativ selten verwendet werden und bei denen der Endbenutzer eine längere Antwortzeit für Abfragen, die diese Daten betreffen, vielleicht akzeptiert. In diesem Fall könnten Sie für diese „historischen" Tabellen einen anderen Tabellenbereich verwenden und diesem Tabellenbereich kostensparendere physische Einheiten mit einer langsameren Zugriffsgeschwindigkeit zuordnen.

Auf der anderen Seite können Sie einige wichtige Tabellen identifizieren, für die eine schnelle Verfügbarkeit und schnelle Antwortzeiten erforderlich sind. Diese Tabellen könnten Sie in einen Tabellenbereich stellen, der einer schnellen physischen Einheit zugeordnet ist, die die Anforderungen für diese wichtigen Daten besser erfüllt.

Wenn Sie mit DMS-Tabellenbereichen arbeiten, können Sie Ihre Tabellendaten auf drei verschiedene Tabellenbereiche verteilen: einen für Indexdaten, einen für Daten großer Objekte (LOB) und Langfelddaten (LF) sowie einen für reguläre Tabellendaten. Auf diese Weise können Sie die Tabellenbereichsmerkmale und die physischen Einheiten der Tabellenbereiche auswählen, die für die Daten am besten geeignet sind. Sie könnten z. B. die Indexdaten auf die schnellste verfügbare Einheit stellen und dadurch eine erhebliche Verbesserung der Leistung erzielen. Wenn Sie eine Tabelle auf DMS-Tabellenbereiche aufteilen, sollten Sie in Betracht ziehen, diese Tabellenbereiche zusammen zu sichern und wiederherzustellen, wenn die aktualisierende Recovery (Rollforward) aktiviert ist. SMS-Tabellenbereiche unterstützen diese Art der Datenverteilung auf Tabellenbereiche nicht.

Verwaltungstechnische Fragen

Einige Verwaltungsfunktionen können auf Tabellenbereichsebene anstatt auf Datenbank- oder Tabellenebene ausgeführt werden. Wenn Sie beispielsweise eine Backup-Kopie eines Tabellenbereichs anstelle der Backup-Kopie einer Datenbank erstellen, können Sie Ihre Zeit und Ressourcen besser ausnutzen. Diese Vorgehensweise ermöglicht Ihnen, Tabellenbereiche mit umfangreichen Änderungen häufig zu sichern und von den Tabellenbereichen, die wenig geändert werden, nur gelegentlich neue Backup-Kopien anzulegen.

Sie können eine Datenbank oder einen Tabellenbereich wiederherstellen. Wenn Tabellen, die sich nicht aufeinander beziehen, keine gemeinsamen Tabellenbereiche benutzen, haben Sie die Option, kleinere Teile Ihrer Datenbank wiederherzustellen und den Aufwand verringern.

Ein gutes Verfahren besteht darin, Tabellen, die voneinander abhängig sind, in einer Gruppe von Tabellenbereichen zusammenzufassen. Diese Tabellen könnten über referenzielle Integritätsbedingungen oder andere definierte Geschäftsregeln voneinander abhängig sein.

Falls Sie eine bestimmte Tabelle häufig löschen und erneut definieren müssen, können Sie die Tabelle in einem eigenen Tabellenbereich definieren, weil das Löschen eines DMS-Tabellenbereichs effizienter ist als das Löschen einer Tabelle.

Automatische Änderung der Größe von Tabellenbereichen

Durch die Aktivierung von Tabellenbereichen mit dynamischem Speicher zur automatischen Größenänderung kann der Datenbankmanager die Problembedingung eines vollen Dateisystems automatisch durch Hinzufügen eines neuen Stripe-Sets mit Containern beheben.

In einem Datenbanksystem können zwei Typen von Tabellenbereichen enthalten sein: vom System verwaltete Bereiche (SMS) und von der Datenbank verwaltete Bereiche (DMS). Die Container, die SMS-Tabellenbereichen zugeordnet sind, sind Dateisystemverzeichnisse und die Dateien in diesen Verzeichnissen werden in gleichem Maße größer wie die Objekte im entsprechenden Tabellenbereich. Die Dateien wachsen, bis für einen der Container eine Dateisystembegrenzung oder die Größenbegrenzung der Datenbank für Tabellenbereiche erreicht wird (siehe SQL- und XML-Begrenzungen).

DMS-Tabellenbereiche bestehen aus Dateicontainern oder Containern für (unformatierte und unpartitionierte) Roheinheiten, und ihre Größe wird definiert, wenn die Container dem Tabellenbereich zugeordnet werden. Der Tabellenbereich gilt dann als voll, wenn der gesamte Speicherplatz des Containers belegt ist. Im Gegensatz zu SMS-Tabellenbereichen können hier jedoch Container mit der Anweisung ALTER TABLESPACE hinzugefügt oder erweitert werden, wodurch der Tabellenbereich mehr Speicherplatz erhält. DMS-Tabellenbereiche verfügen darüber hinaus über eine *Funktion zur automatischen Größenänderung (AUTORESIZE)*: Wenn in einem DMS-Tabellenbereich, für den die automatische Größenänderung aktiviert ist, der Speicherplatz erschöpft ist, kann das Datenbanksystem den Tabellenbereich um einen oder mehrere Dateicontainer erweitern. SMS-Tabellenbereiche verfügen über ähnliche Funktionen zur automatischen Erweiterung, doch der Begriff der Funktion zur automatischen Größenänderung (AUTORESIZE) wird ausschließlich für DMS verwendet.

Die automatische Größenänderung von Tabellenbereichen hat die folgenden Auswirkungen:

- Tabellenbereichen, für die die automatische Größenänderung aktiviert ist, sind Metadaten zugeordnet, die von Version 8.2.1 oder früheren Releases nicht erkannt werden. Jeder Versuch, eine Datenbank mit Tabellenbereichen, für die die automatische Größenänderung aktiviert ist, mit diesen Versionen zu verwenden, führt zu einem Fehler (normalerweise SQL0980C oder SQL0902C). Ein Fehler kann bei dem Versuch, eine Verbindung zu einer Datenbank herzustellen, oder bei dem Versuch, einen Restore der Datenbank auszuführen, zurückgegeben werden. Wenn Sie die automatische Größenänderung für Tabellenbereiche aktiviert haben, werden durch die Inaktivierung der Funktion zur automatischen Größenänderung für diese Tabellenbereiche die Metadaten entfernt, sodass die Datenbank mit Version 8.2.1 oder früheren Releases verwendet werden kann.
- Wenn Sie die Funktion zur automatischen Größenänderung inaktivieren, gehen die Werte, die INCREASESIZE und MAXSIZE zugeordnet sind, verloren und stehen bei einer späteren Aktivierung der Funktion nicht mehr zur Verfügung.
- Sie können diese Funktion nicht für Tabellenbereiche aktivieren, die mit Containern für Roheinheiten arbeiten, und Sie können einem Tabellenbereich, für den die automatische Größenänderung aktiviert ist, keine Container für Roheinheiten hinzufügen. Versuche, solche Operationen auszuführen, schlagen mit Fehlern (SQL0109N) fehl. Wenn Sie Container für Roheinheiten hinzufügen müssen, müssen Sie diese Funktion zuvor inaktivieren.

- Eine umgeleitete Restoreoperation kann die Containerdefinitionen nicht so ändern, dass ein Container für eine Roheinheit verwendet werden kann. Ein Versuch, diese Art von Operation auszuführen, verursacht einen Fehler (SQL0109N).
- Da der Wert für die Maximalgröße die automatische Vergrößerung eines Tabellenbereichs durch den Datenbankmanager begrenzt, legt dieser Maximalwert auch fest, bis zu welcher Größe Sie einen Tabellenbereich vergrößern können. Dies bedeutet, dass bei einer von Ihnen ausgeführten Operation, die einem Tabellenbereich Speicherbereich hinzufügt, die resultierende Größe kleiner oder gleich der Maximalgröße sein muss. Speicherbereich kann mit der Klausel ADD, EXTEND, RESIZE oder BEGIN NEW STRIPE SET der Anweisung ALTER TABLESPACE hinzugefügt werden.

Aktivieren und Inaktivieren der Funktion zur automatischen Größenänderung (AUTORESIZE)

Standardmäßig ist die Funktion zur automatischen Größenänderung für einen DMS-Tabellenbereich nicht aktiviert. Mit der folgenden Anweisung wird ein DMS-Tabellenbereich ohne Aktivierung der automatischen Größenänderung erstellt:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
```

Zum Aktivieren der Funktion zur automatischen Größenänderung geben Sie die Klausel AUTORESIZE YES in der Anweisung CREATE TABLESPACE an:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M) AUTORESIZE YES
```

Sie können die Funktion zur automatischen Größenänderung auch nach der Erstellung eines DMS-Tabellenbereichs aktivieren oder inaktivieren, indem Sie die Klausel AUTORESIZE in der Anweisung ALTER TABLESPACE verwenden:

```
ALTER TABLESPACE DMS1 AUTORESIZE YES
ALTER TABLESPACE DMS1 AUTORESIZE NO
```

Darüber hinaus sind zwei weitere Attribute, MAXSIZE und INCREASESIZE, für die automatische Größenänderung von Tabellenbereichen relevant:

Maximalgröße (MAXSIZE)

Mit der Klausel MAXSIZE der Anweisung CREATE TABLESPACE wird die maximal mögliche Größe für den Tabellenbereich definiert. So wird beispielsweise durch die folgende Anweisung ein Tabellenbereich erstellt, der bis zu einem Maximalwert von 100 MB vergrößert werden kann (pro Datenbankpartition, falls die Datenbank mehrere Datenbankpartitionen besitzt):

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES MAXSIZE 100 M
```

Die Klausel MAXSIZE NONE gibt an, dass für den Tabellenbereich kein Maximalwert festgelegt ist. Der Tabellenbereich kann wachsen, bis eine Dateisystembegrenzung oder die Größenbegrenzung für Tabellenbereiche erreicht wird (siehe SQL- und XML-Begrenzungen). Wenn Sie die Klausel MAXSIZE nicht angeben, gibt es keine Maximalgröße, wenn die Funktion zur automatischen Größenänderung aktiviert wird.

Mit der Anweisung ALTER TABLESPACE können Sie den Wert von MAXSIZE für einen Tabellenbereich ändern, für den die Funktion zur automatischen Größenänderung bereits aktiviert ist, wie in den folgenden Beispielen gezeigt:

```
ALTER TABLESPACE DMS1 MAXSIZE 1 G
ALTER TABLESPACE DMS1 MAXSIZE NONE
```

Wenn Sie eine Maximalgröße angeben, kann der tatsächlich vom Datenbankmanager umgesetzte Wert geringfügig niedriger sein als der angegebene Wert, da der Datenbankmanager versucht, die Vergrößerung von Containern konsistent zu halten.

Vergrößerungsvolumen (INCREASESIZE)

Mit der Klausel INCREASESIZE der Anweisung CREATE TABLESPACE wird die Menge an Speicherplatz definiert, um die der Tabellenbereich vergrößert wird, wenn keine freien Speicherbereiche im Tabellenbereich mehr vorhanden sind, jedoch eine Anforderung für einen oder mehrere Speicherbereiche erfolgt ist. Sie können den Wert als explizite Größe oder als Prozentsatz wie in den folgenden Beispielen angeben:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 5 M

CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 50 PERCENT
```

Ein Prozentsatz bedeutet, dass das Volumen, um das entsprechend dem Wert von INCREASESIZE zu vergrößern ist, jedes Mal, wenn der Tabellenbereich vergrößert werden muss, neu berechnet wird. Dabei basiert das jeweilige Vergrößerungsvolumen auf einem Prozentsatz der Tabellenbereichsgröße zu dem betreffenden Zeitpunkt. Beispiel: Wenn der Tabellenbereich 20 MB groß ist und für INCREASESIZE ein Wert von 50 % angegeben wird, wird der Tabellenbereich bei der ersten Vergrößerung um 10 MB erweitert (auf eine Größe von 30 MB) und bei der nächsten Vergrößerung um 15 MB.

Wenn Sie die Klausel INCREASESIZE bei der Aktivierung der Funktion zur automatischen Größenänderung nicht angegeben, ermittelt der Datenbankmanager einen angemessenen Wert, der sich während des Lebenszyklus des Tabellenbereichs ändern kann. Wie bei AUTORESIZE und MAXSIZE können Sie den Wert für INCREASESIZE mithilfe der Anweisung ALTER TABLESPACE ändern.

Wenn Sie einen Wert für INCREASESIZE angeben, kann der tatsächlich vom Datenbankmanager verwendete Wert geringfügig vom angegebenen Wert abweichen. Diese Anpassung des verwendeten Wertes dient dazu, die Vergrößerungen für alle Container im Tabellenbereich konsistent zu halten.

Erweiterung von Tabellenbereichen

Bei Tabellenbereichen, deren Größe automatisch geändert werden kann, versucht der Datenbankmanager, den Tabellenbereich zu vergrößern, wenn der gesamte vorhandene Speicherbereich belegt ist und eine Anforderung für zusätzlichen Speicherbereich abgesetzt wurde. Der Datenbankmanager bestimmt, welche Container im Tabellenbereich erweitert werden können, sodass kein Neuausgleich stattfindet. Der Datenbankmanager erweitert nur die Container, die sich im letzten Bereich der Tabellenbereichszuordnung befinden (diese Zuordnung beschreibt die Speicherbelegung für den Tabellenbereich), und erweitert sie um dasselbe Volumen.

Betrachten Sie z. B. die folgende Anweisung:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
  USING (FILE 'C:\TS1CONT' 1000, FILE 'D:\TS1CONT' 1000,
         FILE 'E:\TS1CONT' 2000, FILE 'F:\TS1CONT' 2000)
  EXTENTSIZE 4
  AUTORESIZE YES
```

Unter Berücksichtigung der Tatsache, dass der Datenbankmanager einen kleinen Teil jedes Containers (einen EXTENTSIZE großen Speicherbereich) für Metadaten verwendet, ist nachfolgend die Tabellenbereichszuordnung dargestellt, die für den Tabellenbereich auf der Basis der Anweisung CREATE TABLESPACE erstellt wird. (Die Tabellenbereichszuordnung ist Teil der Ausgabe einer Momentaufnahme für einen Tabellenbereich.)

```
Tabellenbereichszuordnung:

 Range  Stripe Stripe  Max         Max  Start  End    Adj.   Containers
 Number Set    Offset  Extent      Page Stripe Stripe
 [   0] [   0]      0     995      3983      0    248   0    4 (0,1,2,3)
 [   1] [   0]      0    1495      5983    249    498   0    2 (2,3)
```

Die Tabellenbereichszuordnung zeigt, dass die Container mit der ID 2 und 3 (`E:\TS1CONT` und `F:\TS1CONT`) die einzigen Container im letzten Bereich (Range) der Zuordnung sind. Wenn der Datenbankmanager die Container in diesem Tabellenbereich automatisch erweitert, werden daher nur diese beiden Container erweitert.

Anmerkung: Wenn Sie einen Tabellenbereich erstellen, bei dem alle Container dieselbe Größe haben, enthält die Zuordnung nur einen Bereich (Range). In diesem Fall erweitert der Datenbankmanager jeden Container. Um zu verhindern, dass die Erweiterung nur auf eine Untermenge der Container begrenzt wird, müssen Sie einen Tabellenbereich mit Containern gleicher Größe erstellen.

Wie zuvor erläutert, können Sie eine Begrenzung für die Maximalgröße des Tabellenbereichs angeben. Sie können auch den Wert NONE angeben, sodass keine Größenbegrenzung besteht. Wenn Sie NONE oder keinen Grenzwert angeben, wird die Obergrenze durch den Grenzwert des Dateisystems oder des Tabellenbereichs definiert. Der Datenbankmanager versucht nicht, die Tabellenbereichsgröße über die Obergrenze hinaus zu erweitern. Es kann jedoch vorkommen, dass ein Versuch, einen Container zu erweitern, fehlschlägt, bevor dieser Grenzwert erreicht ist, wenn das Dateisystem voll ist. In diesem Fall vergrößert der Datenbankmanager den Tabellenbereich nicht weiter und gibt eine Fehlerbedingung wegen nicht ausreichenden Speicherplatzes an die Anwendung zurück. In dieser Situation haben Sie zwei Möglichkeiten zur Lösung des Problems:

- Stellen Sie mehr verfügbaren Speicherplatz in dem vollen Dateisystem bereit.
- Führen Sie Containeroperationen für den Tabellenbereich aus, die bewirken, dass sich der betreffende Container nicht mehr im letzten Bereich (Range) der Tabellenbereichszuordnung befindet. Die einfachste Methode, dies zu erreichen, besteht darin, dem Tabellenbereich ein neues Stripe-Set mit einer neuen Gruppe von Containern hinzuzufügen, wobei nach Möglichkeit sichergestellt werden sollte, dass alle Container dieselbe Größe haben. Sie können neue Stripe-Sets mit der Klausel BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE hinzufügen. Durch das Hinzufügen eines neuen Stripe-Sets wird der Tabellenbereichszuordnung ein neuer Bereich hinzugefügt. Durch einen neu hinzugefügten Bereich befinden sich die Container, die der Datenbankmanager automatisch zu erweitern versucht, in diesem neuen Stripe-Set, und die älteren Container bleiben unverändert.

  Anmerkung: Wenn eine vom Benutzer eingeleitete Containeroperation ansteht oder gerade ein nachfolgender Neuausgleich ausgeführt wird, wird die Funktion

zur automatischen Größenänderung so lange inaktiviert, bis die Operation festgeschrieben bzw. der Neuausgleich abgeschlossen ist.

Ein Beispiel für DMS-Tabellenbereiche: Nehmen Sie einmal an, dass ein Tabellenbereich drei Container umfasst, die dieselbe Größe haben und sich jeweils in einem eigenen Dateisystem befinden. Während der Ausführung von Operationen im Tabellenbereich erweitert der Datenbankmanager diese drei Container automatisch. Schließlich ist eines der Dateisysteme voll, und der zugehörige Container kann nicht mehr erweitert werden. Wenn es nicht möglich ist, im Dateisystem zusätzlichen freien Speicherplatz verfügbar zu machen, müssen Sie Containeroperationen im Tabellenbereich durchführen, die bewirken, dass sich der betreffende Container nicht mehr im letzten Bereich (Range) der Tabellenbereichszuordnung befindet. In diesem Fall könnten Sie beispielsweise ein neues Stripe-Set hinzufügen und dabei zwei Container angeben (jeweils einen für jedes Dateisystem, in dem noch Speicherplatz zur Verfügung steht); oder Sie könnten eine größere oder geringere Anzahl von Containern angeben (und dabei sicherstellen, dass jeder der hinzugefügten Container dieselbe Größe hat und in jedem der verwendeten Dateisysteme ausreichend Speicherplatz für mögliche Erweiterungen verfügbar ist). Beim Vergrößern des Tabellenbereichs versucht der Datenbankmanager nun, anstelle der älteren Container die Container in diesem neuen Stripe-Set zu erweitern.

Überwachung

Informationen zur automatischen Größenänderung von DMS-Tabellenbereichen sind Teil der Momentaufnahmenausgabe des Tabellenbereichsmonitors. Die Werte für das Vergrößerungsvolumen (INCREASESIZE) und für die Maximalgröße (MAXSIZE) werden ebenfalls in der Ausgabe wie im folgenden Beispiel angezeigt:

```
Autom. Größenänderung aktiviert               = Ja oder Nein
Aktuelle Größe des Tabellenbereichs (Byte)    = ###
Maximale Größe des Tabellenbereichs (Byte)    = ### oder NONE
Vergrößerungsvolumen (Byte)                   = ###
Vergrößerungsvolumen (Prozent)                = ###
Zeit der letzten erfolgreichen Größenänderung = TT/MM/JJJJ HH:MM:SS.SSSSSS
Letzte fehlgeschlagene Größenänderung         = Ja oder Nein
```

Automatische Anpassung von PREFETCHSIZE nach Hinzufügen oder Löschen von Containern

Der Datenbankmanager ist so konfiguriert, dass die automatische Bestimmung des Werts für PREFETCHSIZE die Standardeinstellung für alle Tabellenbereiche ist, die unter Version 8.2 (und späteren Versionen) erstellt werden.

Sie sollten sich nicht darum zu kümmern brauchen, die Vorablesezugriffsgröße (PREFETCHSIZE) nach dem Hinzufügen oder Löschen von Containern anzupassen. Das Standardverhalten des Datenbankmanagers zur automatischen Anpassung der Vorablesezugriffsgröße enthebt Sie dieser Sorge. Wenn Sie mit der Möglichkeit rechnen, dass Sie eventuell vergessen, den Wert für PREFETCHSIZE eines Tabellenbereichs nach dem Hinzufügen oder Löschen von Containern anzupassen, sollten Sie das Standardverhalten nicht ändern, sondern den Wert für PREFETCHSIZE durch den Datenbankmanager automatisch bestimmen lassen. Wenn Sie die automatische Anpassung der Vorablesezugriffsgröße nicht zulassen, indem Sie die Standardeinstellung ändern, und vergessen, die Vorablesezugriffsgröße zu aktualisieren, kann es zu merklichen Leistungseinbußen in der Datenbank kommen.

Es gibt drei Möglichkeiten, die Vorablesezugriffsgröße für einen Tabellenbereich nicht automatisch bestimmen zu lassen:

- Erstellen Sie den Tabellenbereich mit einem bestimmten Wert für PREFETCHSIZE. Durch die manuelle Auswahl eines Werts für PREFETCHSIZE geben Sie an, dass Sie bereit sind, den Wert für PREFETCHSIZE bei Bedarf selbst anzupassen, wenn eine Änderung der dem Tabellenbereich zugeordneten Anzahl von Containern erfolgt.
- Verwenden Sie keinen Wert für PREFETCHSIZE bei der Erstellung des Tabellenbereichs und setzen Sie den Wert des Datenbankkonfigurationsparameters *dft_prefetch_sz* auf einen anderen Wert als AUTOMATIC. Der Datenbankmanager überprüft diesen Parameter, wenn kein expliziter Wert für PREFETCHSIZE bei der Erstellung eines Tabellenbereichs angegeben wird. Wenn ein anderer Wert als AUTOMATIC festgestellt wird, wird dieser als Standardwert für PREFETCHSIZE verwendet. In diesem Fall müssen Sie daran denken, bei Bedarf den Wert für PREFETCHSIZE anzupassen, wenn eine Änderung der dem Tabellenbereich zugeordneten Anzahl von Containern erfolgt.
- Ändern Sie den Wert für PREFETCHSIZE manuell mithilfe einer Anweisung ALTER TABLESPACE.

Verwendung der Registrierdatenbankvariablen DB2_PARALLEL_IO

Vorablesezugriffsanforderungen werden in mehrere kleinere Vorablesezugriffsanforderungen gemäß der Parallelität eines Tabellenbereichs und vor der Übergabe an die Vorablesewarteschlangen aufgeteilt. Die Registrierdatenbankvariable DB2_PARALLEL_IO dient zur Definition der Anzahl physischer Spindeln pro Container sowie zur Beeinflussung der parallelen Ein-/Ausgabe für den Tabellenbereich. Bei inaktivierter paralleler E/A entspricht die Parallelität eines Tabellenbereichs der Anzahl von Containern. Bei aktivierter paralleler E/A ist die Parallelität eines Tabellenbereichs gleich der Anzahl von Containern multipliziert mit dem Wert, der in der Registrierdatenbankvariablen DB2_PARALLEL_IO angegeben ist. (Dies entspricht anders formuliert der Aussage, dass die Parallelität des Tabellenbereichs gleich dem Wert für PREFETCHSIZE dividiert durch den Wert für EXTENTSIZE des Tabellenbereichs ist.)

Im Folgenden wird die Beeinflussung des Werts für PREFETCHSIZE durch die Registrierdatenbankvariable DB2_PARALLEL_IO anhand einiger Beispiele erläutert. (Nehmen Sie an, dass alle Tabellenbereiche mit dem Wert AUTOMATIC für PREFETCHSIZE definiert wurden.)

- DB2_PARALLEL_IO=*
  - Alle Tabellenbereiche verwenden den Standardwert, bei dem die Anzahl von Spindeln für jeden Container gleich 6 ist. Der Wert für PREFETCHSIZE wird bei aktivierter paralleler E/A 6-mal größer sein.
  - Für alle Tabellenbereiche wird die parallele E/A aktiviert. Die Vorablesezugriffsanforderung wird in mehrere kleinere Anforderungen unterteilt, jeweils auf den Wert für PREFETCHSIZE dividiert durch den Wert für EXTENTSIZE (bzw. auf die Anzahl Container multipliziert mit der Anzahl Spindeln).
- DB2_PARALLEL_IO=*:3
  - Alle Tabellenbereiche verwenden 3 als Anzahl Spindeln pro Container.
  - Für alle Tabellenbereiche wird die parallele E/A aktiviert.
- DB2_PARALLEL_IO=*:3,1:1
  - Alle Tabellenbereiche verwenden 3 als Anzahl Spindeln pro Container, mit Ausnahme von Tabellenbereich 1, der den Wert 1 verwendet.
  - Für alle Tabellenbereiche wird die parallele E/A aktiviert.

Tabellenbereiche ohne Dateisystemcaching

Die empfohlene Methode zur Aktivierung bzw. Inaktivierung der ungepufferten Ein-/Ausgabe unter UNIX, Linux und Windows wird auf der Tabellenbereichsebene ausgeführt.

Dadurch können Sie die ungepufferte Ein-/Ausgabe für bestimmte Tabellenbereiche aktivieren bzw. inaktivieren und gleichzeitig jede Abhängigkeit vom physischen Layout der Datenbank umgehen. Darüber hinaus kann der Datenbankmanager in diesem Fall bestimmen, welche Ein-/Ausgabemethode (gepuffert oder ungepuffert) sich für jede einzelne Datei am besten eignet.

Die Klausel NO FILE SYSTEM CACHING dient zur Aktivierung der ungepufferten Ein-/Ausgabe, sodass gleichzeitig das Dateicaching für einen bestimmten Tabellenbereich inaktiviert wird. Wenn sie aktiviert ist, bestimmt der Datenbankmanager abhängig von der Plattform, welcher Typ von Ein-/Ausgabe, d. h. Direct I/O (DIO) oder Concurrent I/O (CIO), zu verwenden ist. Angesichts der Leistungsverbesserung durch CIO verwendet der Datenbankmanager die CIO-Funktionalität immer dann, wenn sie unterstützt wird. Es gibt keine Benutzerschnittstelle, über die angegeben werden könnte, welche von beiden zu verwenden ist.

Zur Nutzung des maximalen Vorteils der ungepufferten Ein-/Ausgabe kann es notwendig werden, Pufferpools zu vergrößern. Wenn jedoch der Speichermanager mit automatischer Leistungsoptimierung aktiviert und die Pufferpoolgröße auf AUTOMATIC gesetzt wird, führt der Datenbankmanager eine automatische Optimierung der Pufferpoolgröße auf bestmögliche Leistung durch. Beachten Sie, dass diese Funktion erst ab Version 9 verfügbar ist.

Zum Inaktivieren bzw. Aktivieren des Dateisystemcachings geben Sie die Klausel NO FILE SYSTEM CACHING bzw. die Klausel FILE SYSTEM CACHING in der Anweisung CREATE TABLESPACE oder ALTER TABLESPACE an. Bei keiner Angabe einer der beiden Klauseln wird die Standardeinstellung verwendet. Wenn die Anweisung ALTER TABLESPACE ausgeführt wird, müssen vorhandene Verbindungen zur Datenbank erst beendet werden, bevor die neue Caching-Einstellung in Kraft gesetzt wird.

Anmerkung: Wenn ein Attribut abweichend vom Standardwert auf FILE SYSTEM CACHING oder NO FILE SYSTEM CACHING gesetzt wird, ist kein Mechanismus verfügbar, der ein Rücksetzen auf den Standardwert ermöglicht.

Diese Methode der Aktivierung und Inaktivierung des Dateisystemcachings ermöglicht eine Steuerung des Ein-/Ausgabemodus (gepuffert oder ungepuffert) auf der Tabellenbereichsebene.

Anmerkung: Der Ein-/Ausgabezugriff auf Langfelddaten (LF) und große Objekte (LOB) wird für SMS- und DMS-Container unabhängig von der Einstellung für den fraglichen Tabellenbereich gepuffert.

Der Befehl GET SNAPSHOT FOR TABLESPACES kann zum Abfragen der aktuellen Einstellung der Klausel für das Dateisystemcaching verwendet werden. Das folgende Beispiel zeigt einen Ausschnitt aus der Ausgabe des Befehls `DB2 GET SNAPSHOT FOR TABLEPSACES ON db1`:

```
  Name des Tabellenbereichs                  = USERSPACE1
  Tabellenbereichs-ID                        = 2
  Tabellenbereichstyp                        = Von der Datenbank verwalteter Bereich
  Inhaltstyp des Tabellenbereichs            = Alle permanenten Daten.
                                               Großer Tabellenbereich.
```

```
Seitengröße für Tabellenbereich (Byte)     = 4096
Speicherbereichsgröße für Tabellenbereich (Seiten)  = 32
Automatische Vorablesezugriffsgröße aktiviert       = Ja
Derzeit verwendete Pufferpool-ID                    = 1
Pufferpool-ID bei nächstem Systemstart              = 1
Dynamischen Speicher verwenden                      = Ja
Autom. Größenänderung aktiviert                     = Ja
Dateisystemcaching                                  = Nein
Tabellenbereichsstatus                              = 0x'00000000'
 Detaillierte Erläuterung:
   Normal
Vorablesezugriffsgröße für Tabellenbereich (Seiten) = 32
Gesamtanzahl Seiten                                 = 256
```

Alternative Methoden zur Aktivierung/Inaktivierung der ungepufferten Ein-/Ausgabe unter UNIX, Linux und Windows

Einige UNIX-Plattformen unterstützen die Inaktivierung des Caching auf Dateisystemebene durch die Option MOUNT. Weitere Informationen zu dieser Option finden Sie in der Dokumentation zu Ihrem Betriebssystem. Es ist jedoch wichtig, sich über den Unterschied zwischen der Inaktivierung des Cachings auf Tabellenbereichsebene und der Inaktivierung des Cachings auf Dateisystemebene im Klaren zu sein. Auf der Tabellenbereichsebene steuert der Datenbankmanager, welche Dateien mit oder ohne Dateisystemcaching zu öffnen sind. Auf der Dateisystemebene wird jede Datei, die sich in diesem speziellen Dateisystem befindet, ohne Dateisystemcaching geöffnet. Für einige Plattformen, wie zum Beispiel AIX müssen bestimmte Voraussetzungen, wie zum Beispiel die Serialisierung des Lese- und Schreibzugriffs, erfüllt sein, bevor diese Funktion verwendet werden kann. Der Datenbankmanager erfüllt zwar diese Voraussetzungen, jedoch sollten Sie vor der Aktivierung dieser Funktion in der Dokumentation zu Ihrem Betriebssystem alle Informationen zu Voraussetzungen nachlesen, wenn das Zieldateisystem Dateien enthält, die nicht zu DB2 gehören.

Anmerkung: Die inzwischen veraltete Registrierdatenbankvariable DB2_DIRECT_IO, die mit Version 8.1 Fixpack 4 eingeführt wurde, aktiviert kein Dateisystemcaching für alle SMS-Container mit Ausnahme der Tabellenbereiche für Langfelddaten, Daten großer Objekte und temporäre Tabellendaten unter AIX JFS2. Die Definition dieser Registrierdatenbankvariablen in Version 9.1 oder späteren Versionen hat dieselbe Wirkung wie eine Änderung aller Tabellenbereiche (SMS und DMS) mit der Klausel NO FILE SYSTEM CACHING. Die Verwendung der Variablen DB2_DIRECT_IO wird jedoch nicht empfohlen, und es ist beabsichtigt, diese Variable in einem späteren Release zu entfernen. Geben Sie stattdessen NO FILE SYSTEM CACHING auf der Tabellenbereichsebene an.

Alternative Methoden zur Aktivierung/Inaktivierung der ungepufferten Ein-/Ausgabe unter Windows

In früheren Releases konnte die leistungsbezogene Registrierdatenbankvariable DB2NTNOCACHE verwendet werden, um das Dateisystemcaching für alle DB2-Dateien zu inaktivieren und dadurch mehr Speicher für die Datenbank zur Verfügung zu stellen, sodass der Pufferpool oder der Sortierspeicher vergrößert werden konnten. In Version 9.5 ist DB2NTNOCACHE veraltet und und wird in einem zukünftigen Release möglicherweise entfernt. Der Unterschied zwischen der Registrierdatenbankvariablen DB2NTNOCACHE und der Klausel NO FILE SYSTEM CACHING ist die Möglichkeit, das Caching für Tabellenbereiche selektiv zu inaktivieren. Mit Version 9.5 braucht diese Registrierdatenbankvariable zur Inaktivierung des Dateisystemcachings in der gesamten Instanz nicht mehr definiert zu werden, sofern die Instanz nur neu erstellte Tabellen-

bereiche enthält, da NO FILE SYSTEM CACHING die Standardeinstellung ist, wenn nicht FILE SYSTEM CACHING explizit angegeben wird.

Leistungsaspekte

Die ungepufferte Ein-/Ausgabe wird im Wesentlichen zur Verbesserung der Leistung verwendet. In einigen Fällen kann jedoch möglicherweise zu Leistungseinbußen kommen, die auf die Kombination einer kleinen Pufferpoolgröße mit einem kleinen Dateisystemcache oder auch auf andere Gründe zurückzuführen sind. Zur Leistungsverbesserung können zum Beispiel folgende Maßnahmen empfohlen werden:

- Wenn der Speichermanager mit automatischer Leistungsoptimierung nicht aktiviert ist, aktivieren Sie ihn und setzen die Pufferpoolgröße auf AUTOMATIC, in dem Sie die Anweisung `ALTER BUFFERPOOL <name> SIZE AUTOMATIC` ausführen. Dies ermöglicht dem Datenbankmanager, die Pufferpoolgröße zu optimieren.
- Wenn der Speichermanager mit automatischer Leistungsoptimierung nicht aktiviert werden soll, vergrößern Sie den Pufferpool in Inkrementen von 10 oder 20 %, bis eine Leistungsverbesserung eintritt.
- Wenn der Speichermanager mit automatischer Leistungsoptimierung nicht aktiviert werden soll, ändern Sie den Tabellenbereich, sodass er „FILE SYSTEM CACHING" verwendet. Dies bewirkt im Wesentlichen, dass die ungepufferte Ein-/Ausgabe inaktiviert wird und der Containerzugriff wieder mit gepufferter Ein-/Ausgabe erfolgt.

Die Leistungsverbesserung sollte vor der Implementierung auf dem Produktionssystem unter kontrollierten Umgebungsbedingungen getestet werden.

Wenn Sie beabsichtigen, Dateisystemdateien anstelle von Einheiten als Tabellenbereichscontainer zu verwenden, sollten Sie das Dateisystemcaching in Betracht ziehen, das wie folgt ausgeführt wird:

- Seiten aus DMS-Dateicontainern (und aus allen SMS-Containern) können vom Betriebssystem im Dateisystemcache zwischengespeichert werden, es sei denn, der Tabellenbereich ist mit NO FILESYSTEM CACHING definiert.
- Seiten aus den Tabellenbereichen von DMS-Einheitencontainern werden vom Betriebssystem nicht im Dateisystemcache zwischengespeichert.

Verwenden von CIO/DIO als Standardmechanismus für das Dateisystemcaching für neue Tabellenbereichscontainer

Der Standard-E/A-Mechanismus für neu erstellte Tabellenbereichscontainer auf den meisten AIX-, Linux-, Solaris- und Windows-Plattformen ist CIO/DIO (Concurrent I/O oder Direct I/O). Diese Standardeinstellung bietet eine Durchsatzsteigerung gegenüber gepufferter Ein-/Ausgabe bei Auslastungen mit intensiver Transaktionsverarbeitung sowie bei Rollbackoperationen.

Das Attribut FILE SYSTEM CACHING bzw. NO FILE SYSTEM CACHING gibt an, ob E/A-Operationen auf der Dateisystemebene im Cache zwischengespeichert werden sollen oder nicht:

- FILE SYSTEM CACHING gibt an, dass alle E/A-Operationen im Zieltabellenbereich auf der Dateisystemebene im Cache zwischengespeichert werden sollen.
- NO FILE SYSTEM CACHING gibt an, dass alle E/A-Operationen den Cache auf Dateisystemebene umgehen sollen.

Anmerkung: Wenn Sie mit DMS-Tabellenbereichen arbeiten, sollten Sie einen separaten Tabellenbereich für Langfelddaten (LF-Daten) und für Daten großer Objekte

(LOB-Daten) verwenden, sodass die regulären Tabellenbereiche nicht betroffen sind. (Für SMS-Tabellenbereiche ist das CIO/DIO-Attribut (NO FILE SYSTEM CACHING) inaktiviert.)

Die folgenden Schnittstellen enthalten das Attribut FILE SYSTEM CACHING:

- Anweisung CREATE TABLESPACE
- Befehl CREATE DATABASE
- API sqlecrea() (Feld *sqlfscaching* der Struktur SQLETSDESC)

Wenn dieses Attribut in der Anweisung CREATE TABLESPACE oder im Befehl CREATE DATABASE nicht angegeben wird, verarbeitet der Datenbankmanager die Anforderung entsprechend dem Standardverhalten für die Plattform und den Dateisystemtyp. Das genaue Verhalten wird in „Dateisystemcaching - Konfigurationen" beschrieben. Bei der API sqlecrea() weist der Wert `0x2` für das Feld *sqlfscaching* den Datenbankmanager an, die Standardeinstellung zu verwenden.

Beachten Sie, dass der Wert für das Attribut FILE SYSTEM CACHING gegenwärtig von den folgenden Tools interpretiert wird:

- Befehl GET SNAPSHOT FOR TABLESPACES
- Befehl db2pd –tablespaces
- Befehl db2look –d <dbname> –l

Für den Befehl db2look gilt, dass bei keiner Angabe des Attributs FILE SYSTEM CACHING die Ausgabe dieses Attribut nicht enthält.

Beispiel

Nehmen Sie an, dass sich die Datenbank und alle zugehörigen Tabellenbereichscontainer in einem JFS-Dateisystem unter AIX befinden und die folgende Anweisung abgesetzt wird:

```
DB2 CREATE TABLESPACE JFS2
```

In früheren Versionen verwendete der Datenbankmanager bei keiner Angabe des Attributs die gepufferte Ein-/Ausgabe (FILE SYSTEM CACHING) als E/A-Mechanismus. Mit Version 9.5 verwendet der Datenbankmanager NO FILE SYSTEM CACHING.

Dateisystemcaching - Konfigurationen

Das Betriebssystem arbeitet standardmäßig mit einem Caching für Dateidaten, die von der Platte gelesen bzw. auf Platte geschrieben werden.

Eine typische Leseoperation führt einen Zugriff auf die physische Platte aus, um Daten von der Platte in den Dateisystemcache einzulesen und die Daten anschließend in den Anwendungspuffer zu kopieren. Analog führt eine Schreiboperation einen Zugriff auf die physische Platte aus, um die Daten aus dem Anwendungspuffer in den Dateisystemcache und anschließend aus dem Cache auf die physische Platte zu kopieren. Auf dieses Verhalten beim Caching von Daten auf der Dateisystemebene bezieht sich die Klausel FILE SYSTEM CACHING der Anweisung CREATE TABLESPACE. Da der Datenbankmanager ein eigenes Datencaching mithilfe von Pufferpools verwaltet, wird das Caching auf Dateisystemebene nicht benötigt, wenn die Größe des Pufferpools entsprechend optimiert ist.

Anmerkung: Der Datenbankmanager verhindert bereits das Caching der meisten DB2-Daten, mit Ausnahme von temporären Daten und LOBs unter AIX; dies geschieht durch die Inaktivierung der Cacheseiten.

In einigen Fällen kann das Caching auf Dateisystemebene und in den Pufferpools zu Leistungseinbußen führen, da für das doppelte Caching zusätzliche CPU-Zyklen benötigt werden. Zur Vermeidung dieses doppelten Cachings besitzen die meisten Dateisysteme eine Funktion, die das Caching auf der Dateisystemebene inaktiviert. Dies wird häufig als *ungepufferte Ein-/Ausgabe* bezeichnet. Unter UNIX wird diese Funktion gemeinhin als *Direct I/O (oder DIO)* bezeichnet. Unter Windows entspricht dies dem Öffnen einer Datei mit der Markierung (Flag) FILE_FLAG_NO_BUFFERING. Darüber hinaus unterstützen einige Dateisysteme wie zum Beispiel IBM JFS2 oder Symantec VERITAS VxFS auch ein erweitertes Direct I/O, nämlich die noch leistungsfähigere Funktion *Concurrent I/O (CIO)*. Der Datenbankmanager unterstützt diese Funktion mit der Tabellenbereichsklausel NO FILE SYSTEM CACHING. Wenn diese definiert ist, nutzt der Datenbankmanager CIO automatisch in Dateisystemen, in denen diese Funktion vorhanden ist. Diese Funktion kann dazu beitragen, den Speicherbedarf des Dateisystemcache zu senken und so mehr Speicher für andere Verwendungszwecke verfügbar zu machen.

Vor Version 9.5 wurde das Schlüsselwort FILE SYSTEM CACHING impliziert, wenn weder NO FILE SYSTEM CACHING noch FILE SYSTEM CACHING angegeben war. Für Version 9.5 gilt, dass bei keiner Angabe eines der Schlüsselwörter standardmäßig NO FILE SYSTEM CACHING verwendet wird. Diese Änderung betrifft nur neu erstellte Tabellenbereiche. Vor Version 9.5 erstellte vorhandene Tabellenbereiche sind nicht betroffen. Diese Änderung gilt für AIX, Linux, Solaris und Windows mit den folgenden Ausnahmen, wobei das Standardverhalten FILE SYSTEM CACHING bleibt:

- AIX JFS
- Solaris (nicht VxFS)
- Linux für System z
- Alle temporären SMS-Tabellenbereichsdateien
- LF-Datendateien (LF = Long Field) und LOB-Datendateien (LOB = Large Object) in permanenten SMS-Tabellenbereichsdateien

Wenn Sie die Standardeinstellung überschreiben möchten, müssen Sie FILE SYSTEM CACHING oder NO FILE SYSTEM CACHING angeben.

Unterstützte Konfigurationen

Tabelle 46 zeigt die unterstützte Konfiguration zur Verwendung von Tabellenbereichen ohne Dateisystem-Caching. Außerdem wird noch Folgendes angegeben: (a) Ob die DIO-Funktionalität oder die erweiterte DIO-Funktionalität in allen Fällen verwendet wird; (b) es wird das Standardverhalten angegeben, wenn weder NO FILE SYSTEM CACHING noch FILE SYSTEM CACHING für einen Tabellenbereich auf der Basis der Plattform und des Dateisystemtyps angegeben ist.

Tabelle 46. Unterstützte Konfigurationen für Tabellenbereiche ohne Dateisystemcaching

Plattformen	**Dateisystemtyp und erforderliche Mindeststufe**	**Vom Datenbankmanager übergebene Anforderungen DIO oder CIO, wenn NO FILE SYSTEM CACHING angegeben ist**	**Standardverhalten, wenn weder NO FILE SYSTEM CACHING noch FILE SYSTEM CACHING angegeben ist**
AIX 5.3+	Journal File System (JFS)	DIO	FILE SYSTEM CACHING (Siehe Hinweis 1.)
AIX 5.3+	Concurrent Journal File System (JFS2)	CIO	NO FILE SYSTEM CACHING

Tabelle 46. Unterstützte Konfigurationen für Tabellenbereiche ohne Dateisystemcaching (Forts.)

Plattformen	Dateisystemtyp und erforderliche Mindeststufe	Vom Datenbankmanager übergebene Anforderungen DIO oder CIO, wenn NO FILE SYSTEM CACHING angegeben ist	Standardverhalten, wenn weder NO FILE SYSTEM CACHING noch FILE SYSTEM CACHING angegeben ist
AIX 5.3+	VERITAS Storage Foundation für DB2 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
HP-UX 11i (PA-RISC)	VERITAS Storage Foundation 4.1 (VxFS)	CIO	FILE SYSTEM CACHING
HP-UX Version 11i Version 2 (Itanium)	VERITAS Storage Foundation 4.1 (VxFS)	CIO	FILE SYSTEM CACHING
Solaris 9	UNIX File System (UFS)	DIO	FILE SYSTEM CACHING (Siehe Hinweis 2.)
Solaris 10	UNIX File System (UFS)	CIO	FILE SYSTEM CACHING (Siehe Hinweis 2.)
Solaris 9, 10	VERITAS Storage Foundation für DB2 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
Linux-Varianten SLES 9+ und RHEL 4+ (auf folgenden Architekturen: x86, x86_64, IA64, POWER)	ext2, ext3, reiserfs	DIO	NO FILE SYSTEM CACHING
Linux-Varianten SLES 9+ und RHEL 4+ (auf folgenden Architekturen: x86, x86_64, IA64, POWER)	VERITAS Storage Foundation 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
Linux-Varianten SLES 9+ und RHEL 4+ (auf folgender Architektur: zSeries)	ext2, ext3 oder reiserfs auf Small Computer System Interface-Platten (SCSI-Platten) mit Fibre Channel Protocol (FCP)	DIO	FILE SYSTEM CACHING
Windows	Keine bestimmte Anforderung; funktioniert mit allen von DB2 unterstützten Dateisystemen.	DIO	NO FILE SYSTEM CACHING

Anmerkung:

1. Unter AIX JFS ist FILE SYSTEM CACHING das Standardverhalten.
2. Unter Solaris UFS ist NO FILE SYSTEM CACHING das Standardverhalten.
3. VERITAS Storage Foundation für den Datenbankmanager hat möglicherweise andere Betriebssystemvoraussetzungen. Bei den oben aufgeführten Plattformen handelt es sich um für das aktuelle Release unterstützte Plattformen. Informationen zu den Voraussetzungen finden Sie bei der Unterstützung zu VERITAS Storage Foundation für DB2.
4. Bei der Verwendung von SFDB2 5.0 anstelle der oben stehenden Mindeststufen muss das Release MP1 RP1 von SFDB2 5.0 verwendet werden. Diese Release umfasst 5.0-spezifische Fixes.

5. Wenn der Datenbankmanager als Standardeinstellung nicht NO FILE SYSTEM CACHING auswählen soll, müssen Sie FILE SYSTEM CACHING in der entsprechenden SQL, in den entsprechenden Befehlen oder in den entsprechenden APIs angeben.

Beispiele

Beispiel 1: Standardmäßig wird dieser neue Tabellenbereich mit nicht gepufferter Ein-/Ausgabe erstellt. Die Klausel NO FILE SYSTEM CACHING wird impliziert:

```
CREATE TABLESPACE tabellenbereichsname...
```

Beispiel 2: In der folgenden Anweisung gibt die Klausel NO FILE SYSTEM CACHING an, dass die Cachingfunktion auf Dateisystemebene für diesen speziellen Tabellenbereich inaktiviert wird (OFF):

```
CREATE TABLESPACE tabellenbereichsname ... NO FILE SYSTEM CACHING
```

Beispiel 3: Die folgende Anweisung inaktiviert das Caching auf Dateisystemebene für einen vorhandenen Tabellenbereich:

```
ALTER TABLESPACE tabellenbereichsname ... NO FILE SYSTEM CACHING
```

Beispiel 4: Die folgende Anweisung aktiviert das Caching auf Dateisystemebene für einen vorhandenen Tabellenbereich:

```
ALTER TABLESPACE tabellenbereichsname ... FILE SYSTEM CACHING
```

Speicherbereichsgrößen des Tabellenbereichs

Der Wert des Parameters EXTENTSIZE für einen Tabellenbereich gibt die Anzahl der Seiten mit Tabellendaten an, die in einen Container geschrieben werden, bevor Daten in den nächsten Container geschrieben werden.

Beim Auswählen eines Werts für EXTENTSIZE ist Folgendes zu beachten:

- Größe und Typ der Tabellen im Tabellenbereich

  In DMS-Tabellenbereichen wird einer Tabelle gleichzeitig jeweils ein durch EXTENTSIZE definierter Speicherbereich zugeordnet. Wenn beim Füllen der Tabelle mit Daten ein EXTENTSIZE großer Bereich voll ist, wird ein neuer Bereich dieser Größe zugeordnet. Speicher in DMS-Tabellenbereichscontainern wird vorab reserviert. Das bedeutet, dass neue EXTENTSIZE große Speicherbereiche zugeordnet werden, bis der Container vollständig gefüllt ist.

  Speicherbereich in SMS-Tabellenbereichen wird einer Tabelle jeweils in der Größe eines EXTENTSIZE-Bereichs oder einer Seite zugeordnet. Wenn beim Füllen der Tabelle mit Daten ein EXTENTSIZE großer Bereich oder eine Seite voll ist, wird ein neuer Bereich oder eine neue Seite zugeordnet, bis alle EXTENTSIZE-Bereiche oder Seiten in dem betreffenden Dateisystem belegt sind. Bei der Verwendung von SMS-Tabellenbereichen ist die mehrseitige Dateizuordnung zulässig. Die mehrseitige Dateizuordnung ermöglicht die Zuordnung jeweils EXTENTSIZE großer Speicherbereiche anstelle einzelner Seiten.

  Die Zuordnung aus mehreren Seiten bestehender Dateien ist standardmäßig aktiviert. Der Wert des Datenbankkonfigurationsparameters *multipage_alloc* zeigt an, ob die Zuordnung aus mehreren Seiten bestehender Dateien aktiviert ist.

  Anmerkung: Die Zuordnung aus mehreren Seiten bestehender Dateien gilt nicht für Tabellenbereiche für temporäre Tabellen.

  Eine Tabelle besteht aus folgenden separaten Tabellenobjekten:

  – Ein Datenobjekt. Hier werden die regulären Spaltendaten gespeichert.

– Ein Indexobjekt. Hier werden alle für die Tabelle definierten Indizes gespeichert.
– Ein Langfelddatenobjekt (LF-Datenobjekt). Hier werden Langfelddaten gespeichert, wenn die Tabelle eine oder mehrere Spalten mit LONG-Datentypen besitzt.
– Zwei LOB-Datenobjekte. Wenn die Tabelle eine oder mehrere LOB-Spalten hat, werden diese in folgenden beiden Tabellenobjekten gespeichert:
  - Ein Tabellenobjekt für die LOB-Daten
  - Ein zweites Tabellenobjekt für Metadaten, die die LOB-Daten beschreiben
– Ein Blockzuordnungsobjekt für MDC-Tabellen (MDC = mehrdimensionales Clustering)
– Ein zusätzliches XDA-Objekt, in dem XML-Dokumente gespeichert werden.

Jedes Tabellenobjekt wird getrennt gespeichert und ordnet nach Bedarf neue (durch EXTENTSIZE definierte) Bereiche zu. Jedes DMS-Tabellenobjekt wird außerdem mit einem Metadatenobjekt verbunden, das als Speicherbereichsmaske bezeichnet wird und alle EXTENTSIZE großen Bereiche im Tabellenbereich beschreibt, die zu dem Tabellenobjekt gehören. Der Speicherbereich für Speicherbereichsmasken wird ebenfalls jeweils in der Größe von EXTENTSIZE zugeordnet. Daher beträgt die Anfangszuordnung von Speicherbereich für ein Objekt in einem DMS-Tabellenbereich zwei EXTENTSIZE-Größen. (Die Anfangszuordnung von Speicherbereich für ein Objekt in einem SMS-Tabellenbereich beträgt eine Seite).

Wenn Sie viele kleine Tabellen in einem DMS-Tabellenbereich haben, ist es möglich, dass eine relativ große Menge an Speicher für eine relativ kleine Menge von Daten zugeordnet ist. In einem solchen Fall sollten Sie einen kleinen Wert für EXTENTSIZE definieren. Wenn Sie andererseits eine sehr umfangreiche Tabelle mit einer hohen Zuwachsrate haben und einen DMS-Tabellenbereich mit einem niedrigen Wert für EXTENTSIZE verwenden, könnte durch häufiges Zuordnen zusätzlicher Speicherbereiche unnötiger Systemaufwand entstehen.

- Art des Zugriffs auf die Tabellen

  Wenn auf die Tabellen durch zahlreiche Abfragen oder Transaktionen zugegriffen wird, die große Datenmengen verarbeiten, kann das Vorablesen von Daten aus den Tabellen eine wesentliche Leistungsverbesserung bewirken.
- Minimal erforderliche Anzahl von EXTENTSIZE-Speicherbereichen

  Falls nicht genügend Platz für fünf EXTENTSIZE-Speicherbereiche des Tabellenbereichs in den Containern vorhanden ist, wird der Tabellenbereich nicht erstellt.

Seitengrößen für Tabellenbereiche

Beim Entwerfen von Tabellenbereichen müssen Sie sich Gedanken zu Seitengrößen machen.

Sie können eine Seitengrößenbegrenzung von 4, 8, 16 oder 32 KB verwenden. Für jede dieser Seitengrößen gelten zudem Maximalgrößen für die einzelnen Tabellenbereichstypen, die Sie einhalten müssen.

Tabelle 47 auf Seite 205 zeigt die Begrenzungen der Seitengrößen für die verschiedenen Typen von Tabellenbereichen:

Tabelle 47. Seitengrößenbegrenzungen für Tabellenbereiche

Tabellenbereichstyp (in GB)	Seitengrößenbegrenzung - 4 KB	Seitengrößenbegrenzung - 8 KB	Seitengrößenbegrenzung - 16 KB	Seitengrößenbegrenzung - 32 KB
SMS-Tabellenbereiche	64	128	256	512
DMS-Tabellenbereiche (REGULAR)	64	128	256	512
DMS-Tabellenbereiche (LARGE)	2048	4096	8192	16.384
Tabellenbereiche mit dynamischem Speicher (REGULAR)	64	128	256	512
Tabellenbereiche mit dynamischem Speicher (LARGE)	2048	4096	8192	16.384
Tabellenbereiche für temporäre Tabellen	64	128	256	512

Informationen zu den Begrenzungen für die Seitengrößen von Datenbank- und Indexseiten der verschiedenen Typen von Tabellenbereichen finden Sie in den Informationen zu den Seitengrößenbegrenzungen des Datenbankmanagers in SQL- und XML-Begrenzungen.

Lesen Sie die Informationen in „Sicherstellen, dass Seitengrößen von Tabellenbereichen für temporäre Systemtabellen den Anforderungen entsprechen" im Handbuch *Migration*, um sicherzustellen, dass die maximale Seitengröße Ihres Tabellenbereichs für temporäre Tabellen für Ihre Abfragen oder positionierten Aktualisierungen ausreicht.

Platten-E/A für Tabellenbereiche

Die Art und der Aufbau Ihres Tabellenbereichs bestimmen die Effizienz der Ein-/Ausgabeoperationen, die mit diesem Tabellenbereich erzielt werden kann.

Sie sollten sich mit den folgenden Konzepten vertraut machen, bevor Sie weitere Themen zum Entwerfen und Verwenden von Tabellenbereichen in Betracht ziehen:

Lesen großer Blöcke (Big Blocks)
Eine Leseoperation, bei der mehrere Seiten (in der Regel ein durch EXTENTSIZE definierter Bereich) in einer einzigen Anforderung abgerufen werden. Das Lesen mehrerer Seiten in einem Vorgang ist effizienter als das Lesen jeder Seite in getrennten Vorgängen.

Vorablesezugriff
Das Vorablesen der Seiten, auf die in einer Abfrage zugegriffen wird. Der Hauptzweck des Vorablesens ist die Verringerung von Antwortzeiten. Dies kann erreicht werden, wenn das Vorablesen von Seiten asynchron zur Ausführung der Abfrage stattfinden kann. Die besten Antwortzeiten werden erzielt, wenn entweder die CPU(s) oder das E/A-Subsystem mit maximaler Kapazität arbeiten.

Seitenlöschfunktion
Durch das Lesen und Ändern von Seiten sammeln diese sich im Pufferpool

der Datenbank an. Wenn eine Datenseite eingelesen wird, wird sie in eine Seite des Pufferpools eingelesen. Wenn der Pufferpool ganz mit geänderten Seiten gefüllt ist, muss eine dieser geänderten Seiten auf die Platte geschrieben werden, bevor die neue Seite eingelesen werden kann. Bevor nun der Pufferpool gänzlich gefüllt wird, treten Seitenlöschagenten in Aktion, die geänderte Seiten auf die Platte schreiben und im Pufferpool löschen, um die Verfügbarkeit von Pufferpoolseiten für zukünftige Leseanforderungen sicherzustellen.

Immer, wenn der Datenbankmanager das Lesen großer Blöcke (Big Blocks) als vorteilhaft erkennt, werden Lesezugriffe in großen Blöcken ausgeführt. Dies tritt in der Regel beim Abrufen von Daten, die sequenzieller oder teilweise sequenzieller Art sind. Die Menge der Daten, die in einer Leseoperation gelesen werden, hängt von der Größe des Werts für EXTENTSIZE ab. Je größer der Wert für EXTENTSIZE ist, desto mehr Seiten können in einem Vorgang gelesen werden.

Die Leistung sequenzieller Vorableseoperationen kann weiterhin verbessert werden, wenn Seiten vom Datenträger in zusammenhängende Seiten im Pufferpool gelesen werden können. Da Pufferpools standardmäßig seitenbasiert sind, gibt es keine Garantie, dass sich beim Lesen vom Datenträger in zusammenhängende Seiten eine zusammenhängende Gruppe von Seiten finden lässt. Zu diesem Zweck können blockbasierte Pufferpools verwendet werden, weil diese nicht nur einen Seitenbereich, sondern außerdem einen Blockbereich für Gruppen zusammenhängender Seiten enthalten. Jede Gruppe zusammenhängender Seiten wird als Block bezeichnet, und jeder Block enthält eine Anzahl von Seiten, die als Blockgröße bezeichnet wird. Die Größen für den Seiten- und den Blockbereich sowie die Anzahl Seiten in jedem Block sind konfigurierbar.

Die Art, wie der Bereich auf der Platte gespeichert ist, hat Einfluss auf die E/A-Effizienz. In einem DMS-Tabellenbereich mit Einheitencontainern werden die Daten eher zusammenhängend auf der Platte gespeichert und können mit minimaler Suchzeit und Plattenlatenzzeit gelesen werden. Bei der Verwendung von Dateien führt eine große Datei, die zur Verwendung durch einen DMS-Tabellenbereich vorab zugeordnet wurde, ebenfalls eher dazu, dass die Datei auf der Platte zusammenhängend gespeichert wird, insbesondere wenn die Datei in einem noch ungenutzten Speicherbereich zugeordnet wurde. Die Daten können vom Dateisystem jedoch in Teile getrennt an mehr als einer Position auf der Platte gespeichert werden. Dies geschieht häufiger bei Verwendung von SMS-Tabellenbereichen, bei denen Dateien um jeweils eine Seite erweitert werden, wodurch die Fragmentierung wahrscheinlicher wird.

Sie können den Grad des Vorablesens durch Ändern des Parameters PREFETCHSIZE in der Anweisung CREATE TABLESPACE bzw. ALTER TABLESPACE steuern, oder Sie können die Vorablesezugriffsgröße auf AUTOMATIC setzen, damit der Datenbankmanager automatisch die optimale Größe zur Verwendung auswählt. (Der Standardwert für alle Tabellenbereiche in der Datenbank wird durch den Konfigurationsparameter *dft_prefetch_sz* der Datenbank festgelegt.) Der Parameter PREFETCHSIZE gibt dem Datenbankmanager an, wie viele Seiten zu lesen sind, wenn ein Vorablesezugriff ausgelöst wird. Wenn der Wert des Parameters PREFETCHSIZE auf ein Vielfaches des Parameters EXTENTSIZE in der Anweisung CREATE TABLESPACE gesetzt wird, können mehrere EXTENTSIZE große Bereiche parallel gelesen werden. (Der Standardwert für alle Tabellenbereiche in der Datenbank wird durch den Konfigurationsparameter *dft_extent_sz* der Datenbank festgelegt.) Der Parameter EXTENTSIZE gibt die Anzahl der 4-KB-Seiten an, die in einen Container geschrieben werden, bevor zum nächsten Container übergegangen wird.

Nehmen Sie zum Beispiel an, Sie hätten einen Tabellenbereich, der drei Einheiten verwendet. Wenn Sie den Wert für PREFETCHSIZE auf das Dreifache des Werts für EXTENTSIZE setzen, kann der Datenbankmanager einen Lesezugriff in großen Blöcken von jeder Einheit parallel durchführen, wodurch der E/A-Durchsatz erheblich erhöht wird. Voraussetzungen sind, dass jede Einheit eine getrennte physische Einheit ist und dass der Controller über eine ausreichende Bandbreite verfügt, um den Datenstrom von jeder Einheit zu verarbeiten. Beachten Sie, dass der Datenbankmanager eventuell die Parameter für den Vorablesezugriff zur Laufzeit aufgrund der Abfragegeschwindigkeit, der Pufferpoolauslastung und anderer Faktoren dynamisch anpassen muss.

Einige Dateisysteme (z. B. Journaled File System unter AIX) verfügen über eine eigene Vorablesemethode. In einigen Fällen kann der Vorablesezugriff des Dateisystems auf größere Datenmengen eingestellt sein als der Vorablesezugriff des Datenbankmanagers. Dies kann dazu führen, dass der Vorablesezugriff für SMS- und DMS-Tabellenbereiche mit Dateicontainern scheinbar eine bessere Leistung zeigt als der Vorablesezugriff für DMS-Tabellenbereiche mit Einheitencontainern. Dies ist jedoch irreführend, da es sich wahrscheinlich um das Ergebnis einer zusätzlichen Ebene des Vorablesezugriffs handelt, die innerhalb des Dateisystems wirksam ist. Normalerweise sollten DMS-Tabellenbereiche jeder äquivalenten Konfiguration im Hinblick auf die Leistung überlegen sein.

Für ein effizientes Vorablesen (oder auch nur Lesen) muss eine ausreichende Anzahl verfügbarer Pufferpoolseiten vorhanden sein. Zum Beispiel könnte es eine Anforderung zum parallelen Vorablesezugriff geben, mit dem drei (durch EXTENTSIZE bestimmte) Bereiche aus einem Tabellenbereich gelesen werden und durch den für jede einzulesende Seite eine geänderte Seite aus dem Pufferpool herausgeschrieben wird. Die Anforderung zum Vorablesezugriff könnte so weit verlangsamt werden, dass sie mit der Abfrage nicht Schritt halten kann. Daher sollten Seitenlöschfunktionen in ausreichender Anzahl konfiguriert werden, um die Anforderungen von Vorablesezugriffen erfüllen zu können.

Definieren der ersten Tabellenbereiche

Bei der Erstellung einer Datenbank werden drei Tabellenbereiche definiert: (1) SYSCATSPACE für die Systemkatalogtabellen, (2) TEMPSPACE1 für temporäre Systemtabellen, die während der Datenbankverarbeitung erstellt werden, und (3) USERSPACE1 für benutzerdefinierte Tabellen und Indizes.

Anmerkung: Wenn Sie eine Datenbank zum ersten Mal erstellen, wird für diese kein Tabellenbereich für temporäre Benutzertabellen erstellt.

Sofern nicht anders angegeben, werden die drei Standardtabellenbereiche durch den dynamischen Speicher (Automatic Storage) verwaltet.

Bei Verwendung des Befehls CREATE DATABASE können Sie die Seitengröße für den Standardpufferpool und die ersten Tabellenbereiche angeben. Dieser Standardwert stellt auch die Standardseitengröße für alle zukünftigen Anweisungen CREATE BUFFERPOOL und CREATE TABLESPACE dar. Wenn Sie die Seitengröße beim Erstellen der Datenbank nicht angeben, wird die Standardseitengröße von 4 KB verwendet.

Geben Sie Folgendes in die Befehlszeile ein, um erste Tabellenbereiche zu definieren:

```
CREATE DATABASE <name>
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('<pfad>')
    EXTENTSIZE <wert> PREFETCHSIZE <wert>
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'<pfad>' 5000,
                               FILE'<pfad>' 5000)
    EXTENTSIZE <wert> PREFETCHSIZE <wert>
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('<pfad>')
  WITH "<kommentar>"
```

Wenn Sie die Standarddefinition für diese Tabellenbereiche nicht verwenden möchten, können Sie die Merkmale im Befehl CREATE DATABASE angeben. Zum Beispiel könnte der folgende Befehl zur Erstellung der Datenbank unter Windows verwendet werden:

```
CREATE DATABASE PERSONL
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
    EXTENTSIZE 16 PREFETCHSIZE 32
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                               FILE'd:\db2data\personl' 5000)
    EXTENTSIZE 32 PREFETCHSIZE 64
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('f:\db2temp\personl')
  WITH "Personnel DB for BSchiefer Co"
```

In diesem Beispiel werden die Definitionen für jeden der ersten Tabellenbereiche explizit angegeben. Definitionen müssen nur für die Tabellenbereiche angegeben werden, für die die Standarddefinition nicht verwendet werden soll.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken können Sie keine Container erstellen oder bestimmten Datenbankpartitionen zuordnen. Sie müssen die Datenbank zunächst mit den Standardtabellenbereichen für Benutzertabellen und temporäre Tabellen erstellen. Anschließend können Sie mithilfe der Anweisung CREATE TABLESPACE die erforderlichen Tabellenbereiche erstellen. Im letzten Schritt können Sie die Standardtabellenbereiche löschen.

Für die Codierung des Ausdrucks MANAGED BY im Befehl CREATE DATABASE gilt dasselbe Format wie für den Ausdruck MANAGED BY im Befehl CREATE TABLESPACE.

Sie können weitere Benutzertabellenbereiche und Tabellenbereiche für temporäre Tabellen nach Wunsch hinzufügen. Sie können den Katalogtabellenbereich SYSCATSPACE weder löschen noch einen anderen erstellen. Außerdem muss immer mindestens ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von 4 KB vorhanden sein. Sie können weitere Tabellenbereiche für temporäre Systemtabellen erstellen. Nach der Erstellung können auch die Seitengröße und der EXTENTSIZE-Wert eines Tabellenbereichs nicht mehr geändert werden.

Verbindung zu DMS-Einheiten mit direktem Zugriff herstellen

Wenn Container zum Speichern von Daten verwendet werden, unterstützt der Datenbankmanager einen direkten Plattenzugriff (unformatierte Ein-/Ausgabe).

Diese Art der Unterstützung gibt Ihnen die Möglichkeit, eine Einheit mit direktem Plattenzugriff an ein DB2-Datenbanksystem anzuschließen.

Sie müssen die Einheiten- bzw. Dateinamen der Container kennen, auf die Sie bei der Erstellung Ihrer Tabellenbereiche verweisen wollen. Sie müssen wissen, wie viel Speicherplatz auf einer Einheit bzw. in einer Datei, die dem Tabellenbereich zugeordnet werden soll, zur Verfügung steht. Sie benötigen die richtigen Berechtigungen für den Schreib- und Lesezugriff auf den Container.

Die physischen und logischen Verfahren zum Angeben eines direkten Plattenzugriffs unterscheiden sich in Abhängigkeit vom Betriebssystem:

- Unter Windows-Betriebssystemen:

  Verwenden Sie die folgende Syntax, um ein physisches Festplattenlaufwerk anzugeben:

  ```
  \\.\PhysicalDriveN
  ```

  Dabei steht N für eines der physischen Laufwerke im System. Im vorliegenden Fall kann N durch die Werte 0, 1, 2 oder eine beliebige andere positive ganze Zahl ersetzt werden:

  ```
  \\.\PhysicalDrive5
  ```

  Verwenden Sie die folgende Syntax, um ein logisches Laufwerk (d. h. eine unformatierte Datenbankpartition) anzugeben:

  ```
  \\.\N:
  ```

  Dabei ist N: ein Buchstabe eines logischen Laufwerks im System. Die Angabe N: kann z. B. durch E: oder eine beliebige andere Angabe für den Laufwerkbuchstaben ersetzt werden. Um die Einschränkung zu überwinden, die durch die Verwendung eines Buchstabens zur Angabe des Laufwerks gegeben ist, können Sie eine global eindeutige ID (GUID) für das logische Laufwerk verwenden.

  Für Windows gibt es eine neue Methode zur Angabe unformatierter DMS-Tabellenbereichscontainer. Datenträgern (d. h. Datenbankpartitionen auf Basisfestplatten oder dynamischen Datenträgern) wird bei ihrer Erstellung eine global eindeutige ID (GUID) zugeordnet. Die GUID kann als Einheiten-ID bei der Angabe der Container in einer Tabellenbereichsdefinition verwendet werden. Die GUIDs sind über Systeme hinweg eindeutig. Das heißt, dass sie in einer Mehrpartitionsdatenbank für jede Datenbankpartition unterschiedlich sind, selbst wenn die Definitionen der Plattenpartitionen übereinstimmen.

  Es steht ein Tool mit dem Namen *db2listvolumes.exe* (nur für Windows-Betriebssysteme) zur Verfügung, das die Anzeige der GUIDs für alle Plattendatenträger vereinfacht, die auf einem Windows-System definiert sind. Dieses Tool erstellt zwei Dateien in dem Verzeichnis, in dem es ausgeführt wird. Eine Datei mit dem Namen `volumes.xml` enthält in XML codierte Informationen über jeden Plattendatenträger, die einfach mit einem beliebigen XML-fähigen Browser angezeigt werden können. Die zweite Datei mit dem Namen `tablespace.ddl` enthält die erforderliche Syntax zur Angabe der Tabellenbereichscontainer. Diese Datei muss aktualisiert werden, indem die noch ausstehenden Informationen eingefügt werden, die für eine Tabellenbereichsdefinition erforderlich sind. Der Befehl db2listvolumes benötigt keine Befehlszeilenargumente.
- Auf Linux- und UNIX-Plattformen kann ein logischer Datenträger Benutzern und Anwendungen als einzelne, zusammenhängende und erweiterbarer Plattendatenträger angezeigt werden. Trotzdem kann er sich auf nicht zusammenhängenden physischen Datenbankpartitionen und sogar mehr als einem physischen Datenträger befinden. Der logische Datenträger muss außerdem in einer einzigen Datenträgergruppe enthalten sein. Es besteht eine Begrenzung auf 256 logische

Datenträger pro Datenträgergruppe. Es besteht eine Begrenzung auf 32 physische Datenträger pro Datenträgergruppe. Sie können mithilfe des Befehls mklv zusätzliche logische Datenträger erstellen. Mithilfe dieses Befehls können Sie den Namen des logischen Datenträgers angeben und seine Merkmale einschließlich der Anzahl und Position der logischen Partitionen definieren, die für ihn zugeordnet werden sollen.

Nachdem Sie einen logischen Datenträger erstellt haben, können Sie seinen Namen und seine Merkmale mithilfe des Befehls chlv ändern sowie die Anzahl der ihm zugeordneten logischen Partitionen mit dem Befehl extendlv erhöhen. Die standardmäßig vorgegebene Maximalgröße für einen logischen Datenträger bei der Erstellung sind 512 logische Partitionen, sofern sie nicht größer angegeben wurde. Mithilfe des Befehls chlv kann diese Begrenzung außer Kraft gesetzt werden.

In AIX wird die Gruppe der Betriebssystembefehle, Bibliothekssubroutinen und anderen Tools, mit denen Sie logischen Datenträgerspeicher aufbauen und steuern können, Logical Volume Manager (LVM) genannt. LVM steuert Datenträgerressourcen, indem Daten zwischen einer einfacheren und flexiblen logischen Sicht des Speicherbereichs und den tatsächlichen physischen Datenträgern zugeordnet werden.

Weitere Informationen zu mklv und anderen Befehlen für logische Datenträger sowie zu LVM finden Sie im Handbuch *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*.

Konfigurieren und Einrichten des direkten DMS-Plattenzugriffs (Linux)

Wenn Container zum Speichern von Daten verwendet werden, unterstützt der Datenbankmanager einen direkten Plattenzugriff (auf Roheinheiten) über die Blockeinheitenschnittstelle (d. h. unformatierte Ein-/Ausgabe - Raw I/O).

Bevor Sie eine unformatierte Ein-/Ausgabe unter Linux einrichten können, ist mindestens eine freie Datenbankpartition auf IDE- oder SCSI-Platten erforderlich. Um die Plattenpartition beim Erstellen des Tabellenbereichs angeben zu können, müssen Sie wissen, welchen Namen die Plattenpartition hat und wie viel Speicherplatz der Plattenpartition zugeordnet ist, die dem Tabellenbereich zugeordnet werden soll.

In einer Linux-Umgebung sind dabei die folgenden Informationen zu beachten. Unter Linux/390 unterstützt der Datenbankmanager Einheiten mit direktem Plattenzugriff nicht.

Gehen Sie wie folgt vor, um die unformatierte Ein-/Ausgabe unter Linux zu konfigurieren:

In diesem Beispiel ist '/dev/sda5' die zu verwendende unformatierte Datenbankpartition. Diese Partition sollte keine wertvollen Daten enthalten.

1. Berechnen Sie die Anzahl der 4.096 Byte großen Seiten in dieser Datenbankpartition, und runden Sie gegebenenfalls ab. Beispiel:

```
# fdisk /dev/sda
Command (m for help): p

Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes
```

Tabelle 48. Berechnungen für unformatierte E/A unter Linux

Einheiten-start	Anfang	Ende	Blöcke	ID	System
/dev/sda1	1	523	4200997	83	Linux
/dev/sda2	524	1106	4682947+	5	Extended
/dev/sda5	524	1106	4682947	83	Linux

```
Command (m for help): q
#
```

Die Anzahl der Seiten in /dev/sda5 ist:

```
num_pages = floor( (4682947 * 1024)/4096 )
num_pages = 1170736
```

2. Erstellen Sie den Tabellenbereich, indem Sie den Namen der Plattenpartition angeben. Beispiel:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/sda5' 1170736)
```

3. Wenn Sie logische Partitionen mit Junctionpunkten (oder Datenträgermountpunkten) angeben möchten, müssen Sie die Rohpartition an einen anderen NTFS-formatierten Datenträger als Junctionpunkt anhängen und anschließend den Pfad zu dem Junctionpunkt auf dem NTFS-Datenträger als Containerpfad angeben. Beispiel:

```
CREATE TABLESPACE TS4
   MANAGED BY DATABASE USING (DEVICE 'C:\JUNCTION\DISK_1' 10000,
      DEVICE 'C:\JUNCTION\DISK_2' 10000)
```

Der Datenbankmanager fragt zuerst die Partition ab, um festzustellen, ob ein Dateisystem vorhanden ist. Ist dies der Fall, wird die Partition nicht als Roheinheit (RAW) betrachtet, und der Datenbankmanager führt in der Partition normale Dateisystem-E/A-Operationen aus.

Tabellenbereiche auf Roheinheiten werden auch für alle anderen Seitengrößen unterstützt, die vom Datenbankmanager unterstützt werden.

Vor Version 9 wurde ein direkter Plattenzugriff über ein Controllerdienstprogramm für unformatierte Ein-/Ausgabe (Raw-Controller) unter Linux verwendet. Diese Methode ist jetzt veraltet, und von ihrer Verwendung wird abgeraten. Der Datenbankmanager ermöglicht die Verwendung dieser Methode, sofern das Betriebssystem Linux sie noch unterstützt. Jedoch enthält die Datei 'db2diag.log' in diesem Fall eine Nachricht, die darauf hinweist, dass diese Methode veraltet ist.

Bei der früheren Methode mussten Sie eine Plattenpartition an einen Raw-Controller "binden" und anschließend diesen Raw-Controller mithilfe des Befehls CREATE TABLESPACE für den Datenbankmanager definieren:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 1170736)
```

Erstellen von Tabellenbereichen

Bei Tabellenbereichen ohne dynamische Speicherverwaltung müssen Sie die Einheiten- bzw. Dateinamen der Container kennen, auf die Sie bei der Erstellung Ihrer Tabellenbereiche verweisen wollen.

Sie müssen wissen, wie viel Speicherplatz auf einer Einheit bzw. in einer Datei, die dem Tabellenbereich zugeordnet werden soll, zur Verfügung steht. Bei Tabellenbereichen mit dynamischem Speicher ordnet der Datenbankmanager einem Tabellenbereich Container auf der Basis der der Datenbank zugeordneten Speicherpfade zu.

Tabellenbereiche stellen eine Beziehung zwischen den physischen Speichereinheiten, die von Ihrem Datenbanksystem verwendet werden, und den logischen Containern bzw. Tabellen her, die zum Speichern von Daten verwendet werden.

Durch das Erstellen eines Tabellenbereichs innerhalb einer Datenbank werden dem Tabellenbereich Container zugeordnet und die zugehörigen Definitionen und Attribute im Datenbanksystemkatalog gespeichert. Anschließend können Sie Tabellen in diesem Tabellenbereich erstellen. Sie müssen die Einheiten- bzw. Dateinamen der Container kennen, auf die Sie bei der Erstellung Ihrer Tabellenbereiche verweisen wollen. Sie müssen wissen, wie viel Speicherplatz auf einer Einheit bzw. in einer Datei, die dem Tabellenbereich zugeordnet werden soll, zur Verfügung steht.

Beim Erstellen einer Datenbank werden drei erste Tabellenbereiche erstellt. Die Seitengröße für die drei ersten Tabellenbereiche basiert auf dem Standardwert, der bei der Verwendung des Befehls CREATE DATABASE festgelegt oder akzeptiert wird. Dieser Standardwert stellt auch die Standardseitengröße für alle zukünftigen Anweisungen CREATE BUFFERPOOL und CREATE TABLESPACE dar. Wenn Sie die Seitengröße beim Erstellen der Datenbank nicht angeben, wird die Standardseitengröße von 4 KB verwendet. Wenn Sie die Seitengröße beim Erstellen eines Tabellenbereichs nicht angeben, wird die beim Erstellen der Datenbank definierte Seitengröße als Standardseitengröße verwendet.

Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen SMS-Tabellenbereich zu erstellen:

```
CREATE TABLESPACE <NAME>
  MANAGED BY SYSTEM
  USING ('<pfad>')
```

Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen DMS-Tabellenbereich zu erstellen:

```
CREATE TABLESPACE <NAME>
  MANAGED BY DATABASE
  USING (FILE'<pfad>' <größe>)
```

Geben Sie in die Befehlszeile eine der folgenden Anweisungen ein, um einen Tabellenbereich mit dynamischem Speicher zu erstellen:

```
CREATE TABLESPACE <NAME>
```

```
CREATE TABLESPACE <NAME>
  MANAGED BY AUTOMATIC STORAGE
```

Mit der folgenden SQL-Anweisung wird ein SMS-Tabellenbereich unter Windows mit drei Verzeichnissen auf drei separaten Laufwerken erstellt:

```
CREATE TABLESPACE RESOURCE
  MANAGED BY SYSTEM
  USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
```

Mit der folgenden SQL-Anweisung wird ein DMS-Tabellenbereich mit zwei Dateicontainern von je 5000 Seiten erstellt:

```
CREATE TABLESPACE RESOURCE
  MANAGED BY DATABASE
  USING (FILE'd:\db2data\acc_tbsp' 5000,
         FILE'e:\db2data\acc_tbsp' 5000)
```

In den beiden gezeigten Beispielen werden explizit Namen für die Container angegeben. Wenn Sie jedoch relative Containernamen angeben, wird der Container in dem für die Datenbank angelegten Unterverzeichnis erstellt.

Bei der Erstellung von Tabellenbereichscontainern erstellt der Datenbankmanager alle Verzeichnisebenen, die noch nicht vorhanden sind. Wenn zum Beispiel ein Container als /project/user_data/container1 angegeben wird und das Verzeichnis /project nicht vorhanden ist, erstellt der Datenbankmanager die Verzeichnisse /project und /project/user_data.

Alle vom Datenbankmanager erstellten Verzeichnisse werden mit PERMISSION 700 erstellt. Das heißt, dass nur der Instanzeigner über Lese-, Schreib- und Ausführungszugriff verfügt. Da nur der Instanzeigner diesen Zugriff besitzt, kann das folgende Szenario auftreten, wenn mehrere Instanzen erstellt werden:

- Nehmen Sie unter Verwendung derselben Verzeichnisstruktur wie im obigen Beispiel an, dass die Verzeichnisebenen /project/user_data nicht vorhanden sind.
- Benutzer 'user1' erstellt eine Instanz, die standardmäßig den Namen 'user1' erhält. Anschließend erstellt dieser Benutzer eine Datenbank und einen Tabellenbereich mit /project/user_data/container1 als einem seiner Container.
- Benutzer 'user2' erstellt eine Instanz mit dem Standardnamen 'user2'. Anschließend versucht er, einen Tabellenbereich mit /project/user_data/container2 als einem seiner Container zu erstellen.

Da der Datenbankmanager die Verzeichnisebenen /project/user_data mit PERMISSION 700 aus der ersten Anforderung erstellt hat, besitzt Benutzer 'user2' keinen Zugriff auf diese Verzeichnisebenen und kann 'container2' in diesen Verzeichnissen nicht erstellen. In diesem Fall schlägt die Operation CREATE TABLESPACE fehl.

Dieser Konflikt lässt sich mit zwei Methoden lösen:

1. Erstellen Sie das Verzeichnis /project/user_data, bevor Sie die Tabellenbereiche erstellen, und erteilen Sie die Zugriffsberechtigungen, die für beide Benutzer ('user1' und 'user2') zur Erstellung der Tabellenbereiche erforderlich sind. Wenn alle Ebenen des Tabellenbereichsverzeichnisses vorhanden sind, modifiziert der Datenbankmanager die Zugriffsberechtigungen nicht.
2. Wenn Benutzer 'user1' den Tabellenbereichscontainer /project/user_data/container1 erstellt hat, stellen Sie die Zugriffsberechtigungen für /project/user_data ein, die Benutzer 'user2' zur Erstellung des Tabellenbereichs benötigt.

Wird ein Unterverzeichnis vom Datenbankmanager erstellt, kann es auch wieder vom Datenbankmanager gelöscht werden, wenn der Tabellenbereich gelöscht wird.

In diesem Szenario wird davon ausgegangen, dass die Tabellenbereiche nicht einer bestimmten Datenbankpartitionsgruppe zugeordnet werden. Die Standarddatenbankpartitionsgruppe IBMDEFAULTGROUP wird verwendet, wenn der folgende Parameter in der Anweisung nicht angegeben wird:

```
IN datenbankpartitionsgruppenname
```

Mit der folgenden SQL-Anweisung werden ein DMS-Tabellenbereich auf einem AIX-System mit drei logischen Datenträgern von je 10.000 Seiten erstellt und die zugehörigen E/A-Merkmale angegeben:

```
CREATE TABLESPACE RESOURCE
  MANAGED BY DATABASE
  USING (DEVICE '/dev/rdblv6' 10000,
         DEVICE '/dev/rdblv7' 10000,
         DEVICE '/dev/rdblv8' 10000)
  OVERHEAD 7.5
  TRANSFERRATE 0.06
```

Die in der SQL-Anweisung angegebenen UNIX-Einheiten müssen bereits vorhanden sein, und der Instanzeigner und die SYSADM-Gruppe müssen Schreibzugriff auf sie haben.

Im folgenden Beispiel wird ein DMS-Tabellenbereich in einer Datenbankpartitionsgruppe mit dem Namen ODDGROUP in einer UNIX-Datenbank mit mehreren Datenbankpartitionen erstellt. Die Datenbankpartitionsgruppe ODDGROUP muss zuvor mit der Anweisung CREATE DATABASE PARTITION GROUP erstellt worden sein. Im vorliegenden Beispiel wird angenommen, dass die Datenbankpartitionsgruppe ODDGROUP aus den Datenbankpartitionen 1, 3 und 5 besteht. In allen Datenbankpartitionen ist die Einheit /dev/hdisk0 für 10.000 4-KB-Seiten zu verwenden. Außerdem wird für jede Datenbankpartition eine Einheit von 40.000 4-KB-Seiten deklariert.

```
CREATE TABLESPACE PLANS IN ODDGROUP
  MANAGED BY DATABASE
  USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000)
        ON DBPARTITIONNUM 1
       (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000)
        ON DBPARTITIONNUM 3
       (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000)
        ON DBPARTITIONNUM 5
```

Der Datenbankmanager kann die Leistung sequenzieller E/A-Operationen durch die Verwendung des sequenziellen Vorablesezugriffs (Sequential Prefetching) wesentlich verbessern, da dieser mit parallelen E/A-Operationen arbeitet.

Sie können auch einen Tabellenbereich erstellen, der den Standardwert von 4 KB für die Seitengröße übersteigt. Durch die folgende SQL-Anweisung wird ein SMS-Tabellenbereich auf einem Linux- und UNIX-System mit einer Seitengröße von 8 KB erstellt:

```
CREATE TABLESPACE SMS8K
  PAGESIZE 8192
  MANAGED BY SYSTEM
  USING ('FSMS_8K_1')
  BUFFERPOOL BUFFPOOL8K
```

Beachten Sie, dass der zugeordnete Pufferpool ebenfalls die Seitengröße von 8 KB aufweisen muss.

Der erstellte Tabellenbereich kann nicht verwendet werden, bis der Pufferpool, auf den er verweist, aktiviert ist.

Die Anweisung ALTER TABLESPACE kann dazu verwendet werden, Container in einem DMS-Tabellenbereich hinzuzufügen, zu löschen oder in der Größe zu ändern sowie die Einstellungen für die Parameter PREFETCHSIZE, OVERHEAD und TRANSFERRATE für einen Tabellenbereich zu modifizieren. Die Transaktion, die die Anweisung für den Tabellenbereich absetzt, sollte möglichst bald nach der SQL-Anweisung ALTER TABLESPACE festgeschrieben werden, um Konkurrenzsituationen beim Zugriff auf den Systemkatalog zu vermeiden.

Anmerkung: Für den Wert für PREFETCHSIZE sollte ein Vielfaches des Werts für EXTENTSIZE verwendet werden. Wenn beispielsweise für EXTENTSIZE der Wert 10 angegeben wird, sollte der Wert für PREFETCHSIZE 20 oder 30 sein. Es empfiehlt sich, zur manuellen Einstellung des Wertes für PREFETCHSIZE bei der Erstellung eines Tabellenbereichs die folgende Gleichung zu verwenden:

```
PREFETCHSIZE = (Anzahl Container) X (Anzahl physischer Spindeln pro
                Container) X EXTENTSIZE
```

Sie sollten zudem in Betracht ziehen, den Wert für PREFETCHSIZE vom Datenbankmanager automatisch bestimmen zu lassen, indem Sie PREFETCHSIZE auf AUTOMATIC setzen.

Eine direkte Ein-/Ausgabe (Direct I/O, DIO) verbessert die Speicherleistung, weil das Caching auf der Dateisystemebene umgangen wird. Dieser Prozess verringert den CPU-Aufwand und stellt der Datenbankinstanz mehr Speicher zur Verfügung.

Eine gleichzeitige Ein-/Ausgabe (Concurrent I/O, CIO) bietet die Vorteile von DIO und entlastet zudem die serielle Verarbeitung von Schreibzugriffen.

DIO und CIO werden von AIX unterstützt. DIO wird von HP-UX, Solaris, Linux und Windows-Betriebssystemen unterstützt.

Die Schlüsselwörter NO FILE SYSTEM CACHING und FILE SYSTEM CACHING sind Teil der SQL-Anweisungen CREATE TABLESPACE und ALTER TABLESPACE und ermöglichen die Angabe, ob DIO oder CIO für die einzelnen Tabellenbereiche zu verwenden ist. Wenn NO FILE SYSTEM CACHING definiert ist, versucht der Datenbankmanager, nach Möglichkeit gleichzeitige Ein-/Ausgabeoperationen (Concurrent I/O, CIO) zu nutzen. In Fällen, in denen CIO nicht unterstützt wird (z. B. bei Verwendung von JFS), wird stattdessen DIO verwendet.

Wenn Sie die Anweisung CREATE TABLESPACE absetzen, wird die Funktion zur Recovery gelöschter Tabellen standardmäßig aktiviert. Diese Funktion ermöglicht es, gelöschte Tabellendaten über einen Restore und anschließende aktualisierende Recovery auf Tabellenbereichsebene wiederherzustellen. Diese Art der Recovery ist hilfreich, da sie weniger Zeit in Anspruch nimmt als eine Recovery auf Datenbankebene; zudem bleibt Ihre Datenbank für die Benutzer verfügbar.

Allerdings kann sich die Funktion zur Recovery gelöschter Tabellen auf die Leistung der aktualisierenden Recovery auswirken, wenn viele Tabellenlöschoperationen wiederhergestellt werden sollen oder wenn die Protokolldatei sehr groß ist.

Sie können diese Funktion inaktivieren, wenn Sie zahlreiche Tabellenlöschoperationen ausführen möchten und Sie entweder die Umlaufprotokollierung verwenden oder nicht beabsichtigen, eine der gelöschten Tabellen wiederherzustellen. Zur Inaktivierung dieser Funktion können Sie die Option DROPPED TABLE RECOVERY explizit auf OFF setzen, wenn Sie die Anweisung CREATE TABLESPACE absetzen. Alternativ dazu können Sie die Funktion zur Recovery gelöschter Tabellen für einen vorhandenen Tabellenbereich mit der Anweisung ALTER TABLESPACE inaktivieren.

Erstellen von Tabellenbereichen für temporäre Systemtabellen

Ein Tabellenbereich für temporäre Systemtabellen dient zum Speichern temporärer Systemtabellen. Eine Datenbank muss immer über mindestens einen Tabellenbereich für temporäre Systemtabellen verfügen, da temporäre Systemtabellen nur in einem solchen Tabellenbereich gespeichert werden können.

Beim Erstellen einer Datenbank wird einer der drei Standardtabellenbereiche als Tabellenbereich für temporäre Systemtabellen mit dem Namen "TEMPSPACE1" definiert.

Mit der Anweisung CREATE TABLESPACE können Sie einen weiteren Tabellenbereich für temporäre Systemtabellen erstellen. Beispiel:

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
  MANAGED BY SYSTEM
  USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

Sie sollten mindestens einen Tabellenbereich für jede Seitengröße haben.

Die einzige Datenbankpartitionsgruppe, die beim Erstellen eines Tabellenbereichs für temporäre Systemtabellen angegeben werden kann, ist IBMTEMPGROUP.

Erstellen von Tabellenbereichen für temporäre Benutzertabellen

Tabellenbereiche für temporäre Benutzertabellen werden nicht standardmäßig bei der Erstellung einer Datenbank erstellt. Wenn für Ihre Anwendungsprogramme temporäre Tabellen erforderlich sind, müssen Sie einen Tabellenbereich für temporäre Benutzertabellen erstellen, in dem die temporären Tabellen gespeichert werden.

Ebenso wie reguläre Tabellenbereiche können Tabellenbereiche für temporäre Benutzertabellen in allen Datenbankpartitionsgruppen mit Ausnahme von IBMTEMPGROUP erstellt werden. IBMDEFAULTGROUP ist die Standarddatenbankpartitionsgruppe, die beim Erstellen eines Tabellenbereichs für temporäre Benutzertabellen verwendet wird.

Mit der Anweisung DECLARE GLOBAL TEMPORARY TABLE werden als temporär deklarierte Tabellen für die Verwendung in einem Tabellenbereich für temporäre Benutzertabellen definiert.

Mit der Anweisung CREATE TABLESPACE können Sie einen Tabellenbereich für temporäre Benutzertabellen erstellen:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
  MANAGED BY DATABASE
  USING (FILE 'd:\db2data\user_tbsp' 5000,
  FILE 'e:\db2data\user_tbsp' 5000)
```

Ändern von Tabellenbereichen

Zur Änderung eines Tabellenbereichs über die Befehlszeile verwenden Sie die Anweisung ALTER TABLESPACE.

Sie können SMS- und DMS-Container sowie Container mit dynamischem Speicher ändern. Sie können einen Tabellenbereich darüber hinaus umbenennen und ihn vom Offlinemodus in den Onlinemodus umschalten.

Ändern von SMS-Tabellenbereichen

Bei Verwendung von SMS-Tabellenbereichen können Sie Datenbanken mit einer Einzelpartition nur einen Container bzw. partitionierten Datenbanken einen oder mehrere Container hinzufügen.

Dieser Prozess stimmt mit dem in „Hinzufügen oder Erweitern von DMS-Containern" auf Seite 217 beschriebenen Prozess überein.

Ändern von DMS-Tabellenbereichen

Bei Verwendung von DMS-Tabellenbereichen können Sie Container hinzufügen, erweitern, neu ausgleichen, in der Größe ändern, löschen oder verkleinern.

Hinzufügen oder Erweitern von DMS-Containern

Sie können einen DMS-Tabellenbereich (d. h. einen mit der Klausel MANAGED BY DATABASE erstellten Tabellenbereich) vergrößern, indem Sie dem Tabellenbereich einen oder mehrere Container hinzufügen.

Wenn einem Tabellenbereich neue Container hinzugefügt oder vorhandene Container erweitert werden, kann ein Neuausgleich des Tabellenbereichs stattfinden. Der Prozess des Neuausgleichs beinhaltet ein Versetzen von EXTENTSIZE großen Teilen von Tabellenbereichen von einer Position an eine andere. Während dieses Prozesses wird versucht, die einheitenübergreifende Speicherung (Striping) der Daten innerhalb des Tabellenbereichs beizubehalten. Der Neuausgleich findet nicht unbedingt über alle Container hinweg statt, sondern hängt von vielen Faktoren ab, zu denen die vorhandene Containerkonfiguration, die Größe der neuen Container und der Füllungsgrad des Tabellenbereichs zählen.

Wenn Container einem vorhandenen Tabellenbereich hinzugefügt werden, werden sie möglicherweise so hinzugefügt, dass sie nicht in Stripe 0 beginnen, wie in „DMS-Tabellenbereichszuordnungen" auf Seite 176 beschrieben. An welcher Stelle in der Zuordnung sie beginnen, wird durch den Datenbankmanager festgelegt und ist von der Größe der hinzugefügten Container abhängig. Wenn der Container, der hinzugefügt wird, nicht ausreichend groß ist, wird er so positioniert, dass er im letzten Stripe der Zuordnung endet. Wenn er groß genug ist, wird er so positioniert, dass er in Stripe 0 beginnt.

Wenn Sie neue Container hinzufügen und ein neues Stripe-Set erstellen, findet kein Neuausgleich statt. Ein neues Stripe-Set wird mit der Klausel BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE erstellt. Außerdem können Sie vorhandenen Stripe-Sets Container hinzufügen, indem Sie die Klausel ADD TO STRIPE SET in der Anweisung ALTER TABLESPACE verwenden.

Während des Neuausgleichs ist der Zugriff auf den Tabellenbereich nicht eingeschränkt. Wenn mehr als ein Container hinzugefügt werden muss, sollten diese Container gleichzeitig hinzugefügt werden.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um einem DMS-Tabellenbereich einen Container hinzuzufügen:

```
ALTER TABLESPACE <name>
   ADD (DEVICE '<pfad>' <größe>, FILE '<dateiname>' <größe>)
```

Das folgende Beispiel zeigt, wie einem Tabellenbereich auf einem Linux- und UNIX-System zwei neue Einheitencontainer (mit jeweils 10.000 Seiten) hinzugefügt werden.

```
ALTER TABLESPACE RESOURCE
  ADD (DEVICE '/dev/rhd9'  10000,
       DEVICE  '/dev/rhd10' 10000)
```

Beachten Sie, dass Sie mit der Anweisung ALTER TABLESPACE auch andere Merkmale des Tabellenbereichs ändern können, die sich auf die Leistung auswirken können.

Neuverteilung von DMS-Containern

Die Anweisung ALTER TABLESPACE gibt Ihnen die Möglichkeit, einem vorhandenen Tabellenbereich einen Container hinzuzufügen oder einen Containern zu vergrößern, um seine Speicherkapazität zu erhöhen.

Container können Tabellenbereichen mit dynamischem Speicher nicht manuell hinzugefügt werden. Je nach Bedarf erweitert der Datenbankmanager Container automatisch oder fügt Container automatisch hinzu.

Bei der Erstellung eines Tabellenbereichs wird die zugehörige Tabellenbereichszuordnung erstellt und alle Anfangscontainer werden so ausgerichtet, dass sie in Stripe 0 beginnen (Stripe - einheitenübergreifend gespeicherter Datenblock). Dies bedeutet, dass die Daten gleichmäßig über sämtliche Tabellenbereichscontainer verteilt gespeichert werden, bis die einzelnen Container gefüllt sind. (Siehe Beispiel 1.)

Durch das Hinzufügen eines Containers, der kleiner als vorhandene Container ist, wird eine ungleichmäßige Verteilung der Daten verursacht. Dies kann dazu führen, dass parallele E/A-Operationen wie das Vorablesen von Daten weniger effizient arbeiten, als sie es bei Containern gleicher Größe könnten.

Wenn einem Tabellenbereich neue Container hinzugefügt oder vorhandene Container erweitert werden, *kann* ein Neuausgleich der Tabellenbereichsdaten stattfinden.

Neuausgleich

Der Prozess des Neuausgleichs beim Hinzufügen oder Erweitern von Containern besteht im Versetzen von EXTENTSIZE großen Speicherbereichen des Tabellenbereichs von einer Position an eine andere. Dies geschieht, um die einheitenübergreifende Speicherung der Daten innerhalb des Tabellenbereichs beizubehalten.

Während dieser Neuverteilung der Daten wird der Zugriff auf den Tabellenbereich nicht eingeschränkt. Objekte können wie gewöhnlich gelöscht, erstellt, mit Daten gefüllt und abgefragt werden. Allerdings kann die Operation des Datenneuausgleichs erhebliche Auswirkungen auf die Leistung haben. Wenn mehr als ein Container hinzugefügt werden muss und Sie planen, die Container neu auszugleichen, sollten diese Container gleichzeitig innerhalb einer einzigen Anweisung ALTER TABLESPACE hinzugefügt werden, um zu vermeiden, dass der Datenbankmanager die Daten mehr als einmal neu verteilen muss.

Die *obere Grenze* des Tabellenbereichs spielt eine Schlüsselrolle im Neuausgleichsprozess. Die obere Grenze ist die Seitennummer der höchsten zugeordneten Seite im Tabellenbereich. Zum Beispiel hat ein Tabellenbereich 1000 Seiten und einen EXTENTSIZE-Wert von 10, was 100 EXTENTSIZE großen Speicherbereichen entspricht. Wenn der 42ste Speicherbereich der höchste zugeordnete Speicherbereich im Tabellenbereich ist, dann liegt die obere Grenze bei 42 * 10 = 420 Seiten. Dieser Wert ist nicht identisch mit dem Wert für verwendete Seiten, weil einige der Speicherbereiche unterhalb der oberen Grenze freigegeben worden sein könnten, sodass sie zur Wiederverwendung verfügbar sind.

Vor dem Start des Neuausgleichs wird eine neue Tabellenbereichszuordnung auf der Grundlage der vorgenommenen Containeränderungen erstellt. Die Neuausgleichsfunktion versetzt EXTENTSIZE große Speicherbereiche von ihrer Position, die durch die aktuelle Zuordnung festgelegt ist, an die Position, die durch die neue Zuordnung festgelegt wird. Die Neuausgleichsfunktion beginnt mit dem Speicherbereich 0 und versetzt jeweils einen Speicherbereich gleichzeitig, bis der Speicherbereich, der die obere Grenze enthält, versetzt wurde. Beim Versetzen der einzel-

nen Speicherbereiche wird die aktuelle Zuordnung stückweise in das Aussehen der neuen Zuordnung geändert. Wenn der Neuausgleich abgeschlossen ist, sollten die aktuelle Zuordnung und die neue Zuordnung bis zu dem Stripe identisch aussehen, der die obere Grenze enthält. Die aktuelle Zuordnung wird dann vollständig an das Aussehen der neuen Zuordnung angeglichen und der Neuausgleichsprozess ist abgeschlossen. Wenn die Position eines Speicherbereichs in der aktuellen Zuordnung mit seiner Position in der neuen Zuordnung übereinstimmt, wird der Speicherbereich nicht versetzt, und es finden keine E/A-Operationen statt.

Wenn ein neuer Container hinzugefügt wird, hängt die Positionierung dieses Containers innerhalb der neuen Zuordnung von seiner Größe sowie der Größe der anderen Container in seinem Stripe-Set ab. Wenn der Container groß genug ist, um im ersten Stripe des Stripe-Set zu beginnen und im letzten Stripe (oder dahinter) des Stripe-Set zu enden, wird er in dieser Weise angeordnet (siehe Beispiel 2). Wenn der Container dazu nicht groß genug ist, wird er in der Zuordnung so positioniert, dass er im letzten Stripe des Stripe-Set endet (siehe Beispiel 4.) Dies geschieht, um das Volumen der Daten zu minimieren, die neu ausgeglichen werden müssen.

Anmerkung: In den folgenden Beispielen wird bei den Containergrößen die Größe der Containerkennung (container tag) nicht berücksichtigt. Die Containergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Containergrößen dar. Die Beispiele zeigen Container unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch wird empfohlen, Container gleicher Größe zu verwenden.

Beispiel 1:

Wenn Sie einen Tabellenbereich mit drei Containern und einem EXTENTSIZE-Wert von 10 erstellen, und die Container 60, 40 bzw. 80 Seiten (3, 4 und 8 EXTENTSIZE-Werte) groß sind, wird der Tabellenbereich mit einer Zuordnung erstellt, die sich wie in Abb. 13 auf Seite 220 darstellen lässt.

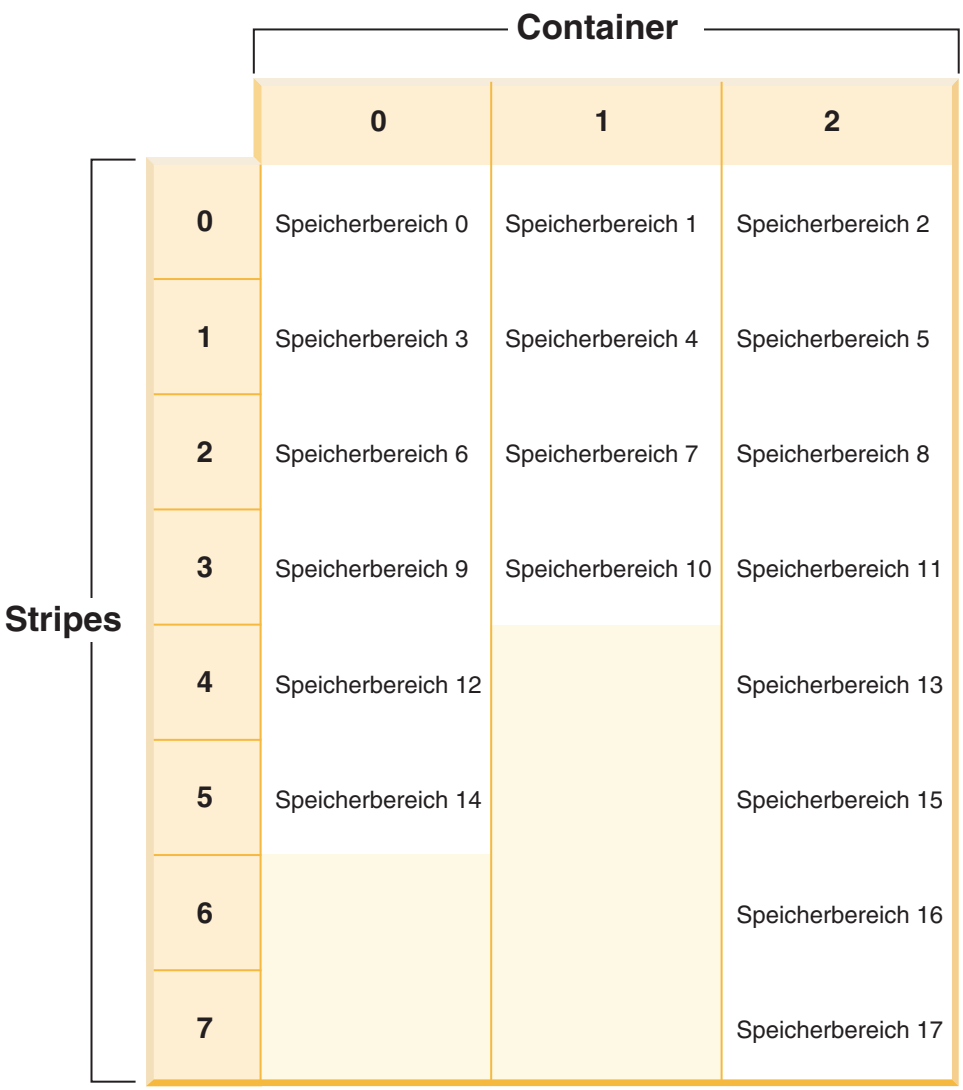


Abbildung 13. Tabellenbereich mit drei Containern und 18 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]      0      11      119       0        3    0   3 (0, 1, 2)
   [1]       [0]      0      15      159       4        5    0   2 (0, 2)
   [2]       [0]      0      17      179       6        7    0   1 (2)
```

Die Spaltenüberschriften in der Tabellenbereichszuordnung heißen 'Range Number' (Bereichsnummer), 'Stripe Set', 'Stripe Offset', 'Maximum extent number addressed by the range' (höchste Speicherbereichsnummer, die durch den Bereich adressiert wird), 'Maximum page number addressed by the range' (höchste Seitennummer, die durch den Bereich adressiert wird), 'Start Stripe' (Anfangsstripe), 'End Stripe' (Endstripe), 'Range adjustment' (Bereichsanpassung) und 'Container list' (Containerliste).

Beispiel 2:

Wenn dem Tabellenbereich in Beispiel 1 ein 80 Seiten großer Container hinzugefügt wird, ist der Container groß genug, um im ersten Stripe (Stripe 0) zu beginnen und

im letzten Stripe (Stripe 7) zu enden. Er wird so positioniert, dass er im ersten Stripe beginnt. Der resultierende Tabellenbereich lässt sich wie in Abb. 14 darstellen.

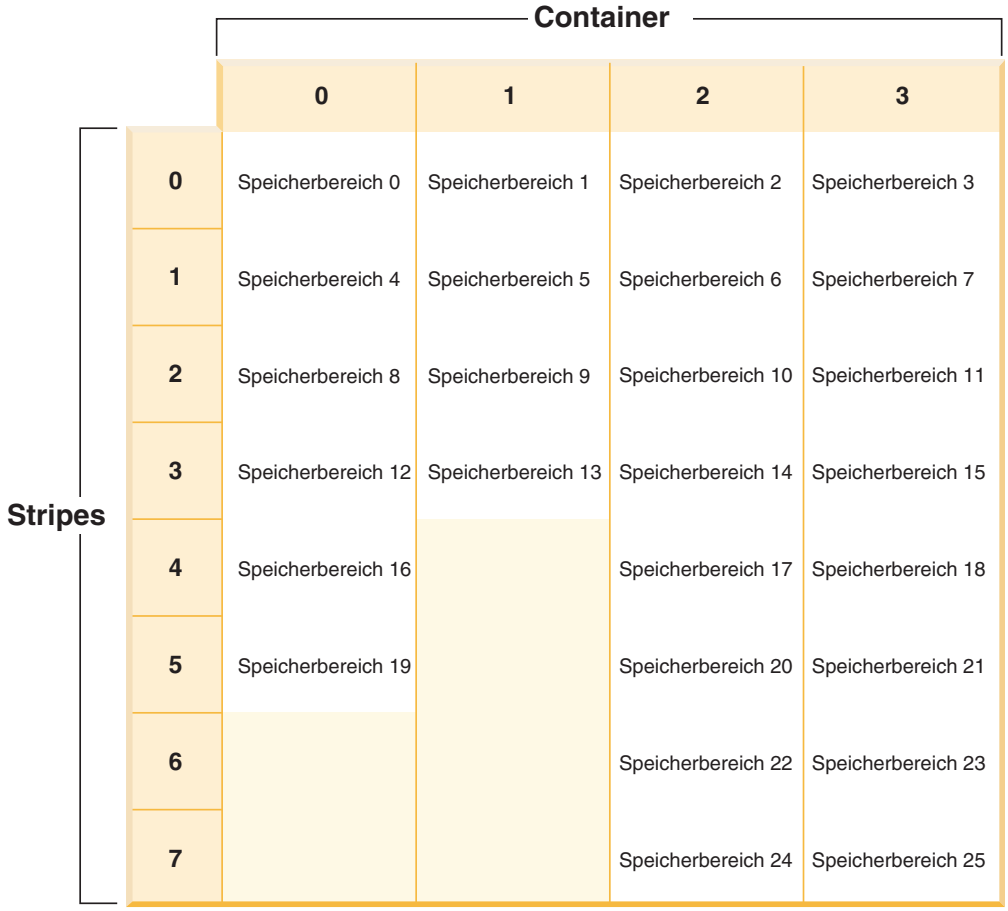


Abbildung 14. Tabellenbereich mit vier Containern und 26 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]      0      15      159       0        3    0   4 (0, 1, 2, 3)
   [1]       [0]      0      21      219       4        5    0   3 (0, 2, 3)
   [2]       [0]      0      25      259       6        7    0   2 (2, 3)
```

Wenn sich die obere Grenze im Speicherbereich 14 befindet, startet die Neuausgleichsfunktion bei Speicherbereich 0 und versetzt alle Speicherbereiche bis einschließlich Speicherbereich 14. Die Position von Speicherbereich 0 innerhalb der beiden Zuordnungen ist identisch, sodass dieser Speicherbereich nicht versetzt werden muss. Das Gleiche gilt für die Speicherbereiche 1 und 2. Speicherbereich 3 muss versetzt werden, sodass der Speicherbereich von der alten Position (zweiter Speicherbereich in Container 0) gelesen und an die neue Position (erster Speicherbereich in Container 3) geschrieben wird. Jeder Speicherbereich nach diesem bis einschließlich Speicherbereich 14 wird versetzt. Wenn Speicherbereich 14 versetzt wurde, sieht die aktuelle Zuordnung wie die neue Zuordnung aus, und die Neuausgleichsfunktion wird beendet.

Wenn die Zuordnung so geändert wird, dass sämtlicher neu hinzugefügter Speicherplatz über der oberen Grenze liegt, ist kein Neuausgleich erforderlich und der gesamte Speicherplatz ist sofort zur Verwendung verfügbar. Wenn die Zuordnung so geändert wird, dass einiger Speicherplatz oberhalb der oberen Grenze liegt, ist der Speicherplatz in den Stripes oberhalb der oberen Grenze verfügbar. Der übrige Speicherplatz ist erst verfügbar, wenn die Neuausgleichsfunktion abgeschlossen ist.

Wenn Sie einen Container erweitern, arbeitet die Funktion des Neuausgleichs ähnlich. Wenn ein Container so erweitert wird, dass er über den letzten Stripe in seinem Stripe-Set hinausreicht, wird das Stripe-Set entsprechend vergrößert, und die folgenden Stripe-Sets werden entsprechend nach hinten verschoben. Infolgedessen reicht der Container nicht in eines der nachfolgenden Stripe-Sets hinein.

Beispiel 3:

Betrachten Sie den Tabellenbereich aus Beispiel 1. Wenn Sie den Container 1 von 40 Seiten auf 80 Seiten erweitern, sieht der neue Tabellenbereich wie in Abb. 15 aus.

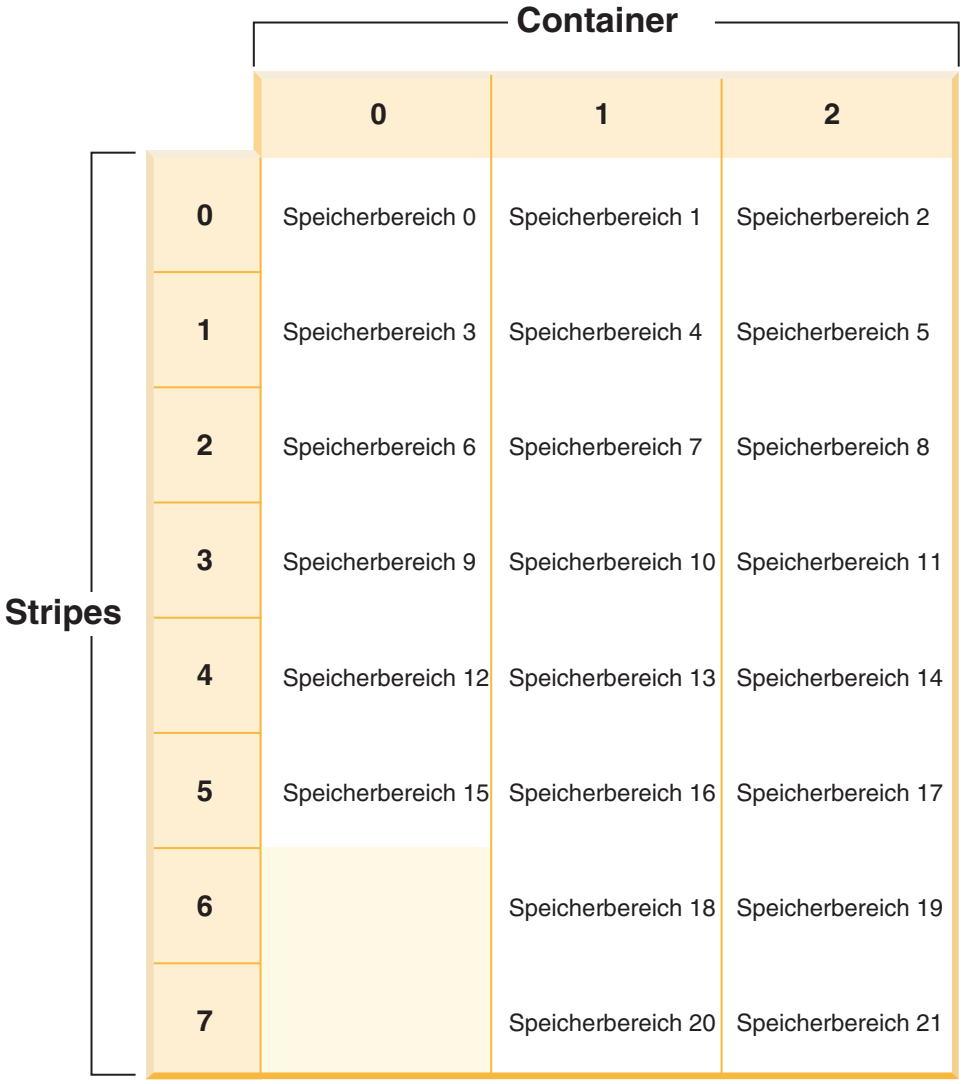


Abbildung 15. Tabellenbereich mit drei Containern und 22 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End      Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0      17      179        0       5      0   3 (0, 1, 2)
   [1]      [0]       0      21      219        6       7      0   2 (1, 2)
```

Beispiel 4:

Betrachten Sie den Tabellenbereich aus Beispiel 1. Wenn ein 50 Seiten großer Container (fünf EXTENTSIZE-Größen) hinzugefügt wird, wird der Container der neuen Zuordnung wie folgt hinzugefügt. Der Container ist nicht groß genug, um im ersten Stripe (Stripe 0) zu beginnen und im letzten Stripe (Stripe 7) oder dahinter zu enden. Daher wird er in der Weise angeordnet, dass er im letzten Stripe endet. (Siehe Abb. 16.)

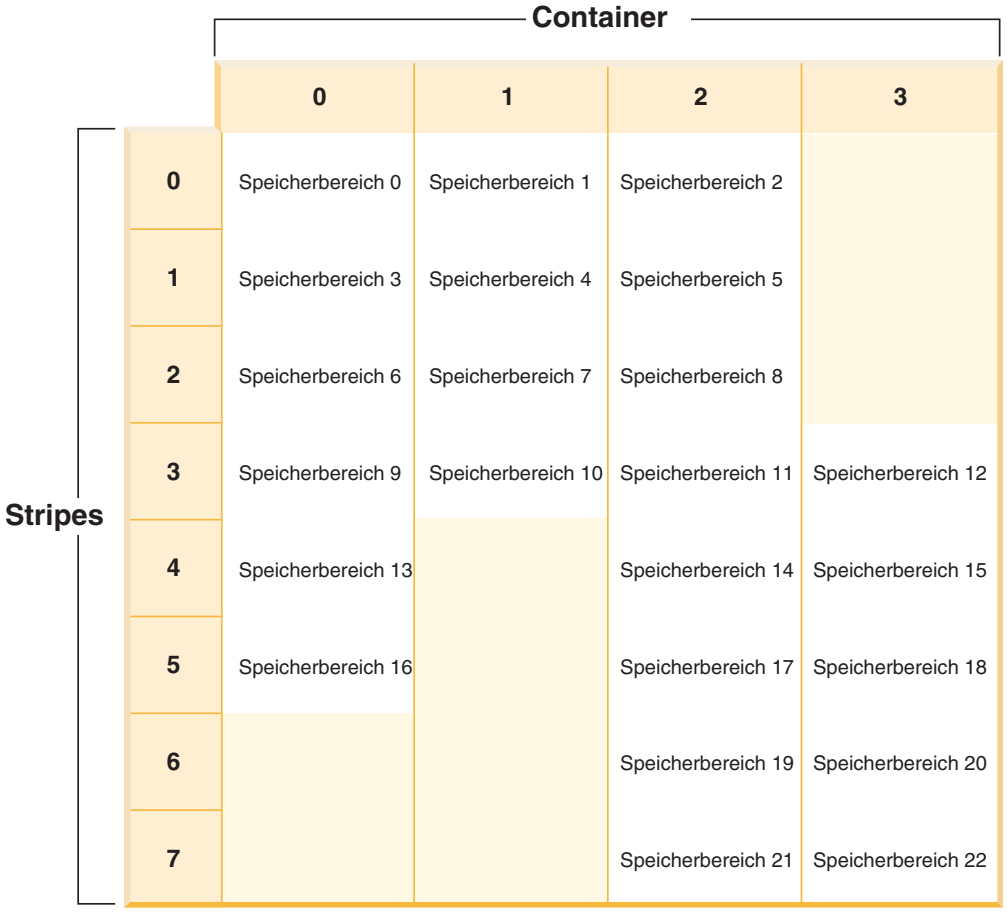


Abbildung 16. Tabellenbereich mit vier Containern und 23 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End      Adj.     Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0       8       89        0       2      0     3 (0, 1, 2)
   [1]      [0]       0      12      129        3       3      0     4 (0, 1, 2, 3)
   [2]      [0]       0      18      189        4       5      0     3 (0, 2, 3)
   [3]      [0]       0      22      229        6       7      0     2 (2, 3)
```

Verwenden Sie zur Erweiterung eines Containers die Klausel EXTEND oder RESIZE in der Anweisung ALTER TABLESPACE. Zum Hinzufügen eines Containers und Neuausgleichen der Daten dient die Klausel ADD in der Anweisung ALTER TABLESPACE. Wenn Sie einem Tabellenbereich, der bereits mehr als ein Stripe-Set hat, einen Container hinzufügen, können Sie angeben, welchem Stripe-Set der Container hinzugefügt werden soll. Dazu verwenden Sie die Klausel ADD TO STRIPE SET in der Anweisung ALTER TABLESPACE. Wenn Sie kein Stripe-Set angeben, wird der Container standardmäßig dem aktuellen Stripe-Set hinzugefügt. Das aktuelle Stripe-Set ist das zuletzt erstellte Stripe-Set, nicht das, dem zuletzt Speicherplatz hinzugefügt wurde.

Jede Änderung an einem Stripe-Set kann einen Neuausgleich in diesem Stripe-Set und den anderen nachfolgenden Stripe-Sets zur Folge haben.

Sie können den Fortschritt eines Neuausgleichs mithilfe von Momentaufnahmen der Tabellenbereiche überwachen. Eine Momentaufnahme eines Tabellenbereichs kann Informationen über einen Neuausgleich liefern, wie zum Beispiel den Startzeitpunkt des Neuausgleichs, die Anzahl der versetzten Speicherbereiche und die Anzahl der noch zu versetzenden Speicherbereiche.

Ohne Neuausgleich (bei Verwendung von Stripe-Sets)

Wenn Sie einen Container hinzufügen oder erweitern und der Speicherplatz oberhalb der oberen Grenze des Tabellenbereichs hinzugefügt wird, findet kein Neuausgleich statt.

Durch Hinzufügen eines Containers wird beinahe immer Speicherplatz unterhalb der oberen Grenze hinzugefügt. Mit anderen Worten, wenn Sie einen Container hinzufügen, ist häufig ein Neuausgleich erforderlich. Es ist eine Option verfügbar, mit der verlasst wird, dass neue Container oberhalb der oberen Grenze hinzugefügt werden. Dadurch ergibt sich die Möglichkeit, keinen Neuausgleich des Inhalts des Tabellenbereichs durchzuführen. Ein Vorteil dieser Methode ist, dass der neue Container sofort zur Verwendung verfügbar ist. Die Option, keinen Neuausgleich durchzuführen, gilt nur für das Hinzufügen von Containern, nicht für die Erweiterung vorhandener Container. Bei der Erweiterung von Containern lässt sich ein Neuausgleich nur vermeiden, wenn der Speicherplatz, den Sie hinzufügen, oberhalb der oberen Grenze liegt. Wenn Sie zum Beispiel eine Anzahl von Containern haben, die die gleiche Größe besitzen, und Sie jeden von ihnen um den gleichen Betrag erweitern, ändern sich die relativen Positionen der Speicherbereiche nicht, und es findet kein Neuausgleich statt.

Das Hinzufügen von Containern zu einem Tabellenbereich ohne Neuausgleich geschieht durch Hinzufügen eines neuen *Stripe-Set*. Ein Stripe-Set ist eine Gruppe von Containern in einem Tabellenbereich, die Datenblöcke enthält, die einheitenübergreifend in ihr gespeichert sind, und die von den anderen Containern getrennt ist, die zu diesem Tabellenbereich gehören. Die vorhanden Container in den vorhandenen Stripe-Sets bleiben unberührt, und die Container, die Sie hinzufügen, werden Teil eines neuen Stripe-Set.

Verwenden Sie die Klausel BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE, wenn Sie Container ohne Neuausgleich hinzufügen wollen.

Beispiel 5:

Wenn Sie einen Tabellenbereich mit drei Containern und einem EXTENTSIZE-Wert von 10 haben, und die Container 30, 40 und 40 Seiten (3, 4 und 4 EXTENTSIZE-Werte) groß sind, kann der Tabellenbereich wie in Abb. 17 dargestellt werden.

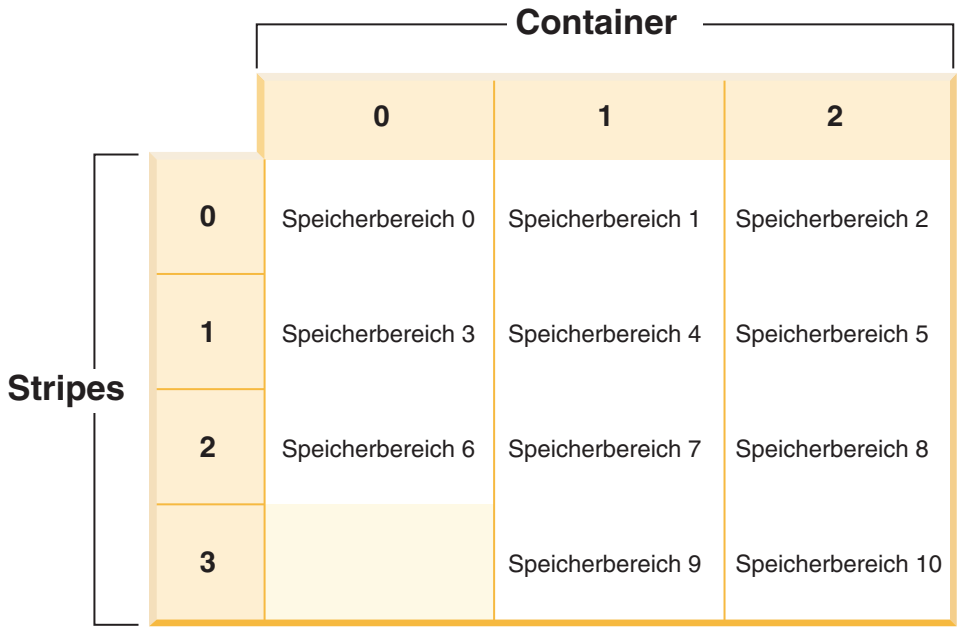


Abbildung 17. Tabellenbereich mit drei Containern und 11 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End      Adj.    Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]      [0]       0       8      89        0        2      0    3 (0, 1, 2)
   [1]      [0]       0      10     109        3        3      0    2 (1, 2)
```

Beispiel 6:

Wenn Sie zwei neue Container mit der Klausel BEGIN NEW STRIPE SET hinzufügen, die 30 und 40 Seiten (3 und 4 EXTENTSIZE-Größen) groß sind, werden die vorhandenen Bereiche (Ranges) davon nicht berührt. Stattdessen wird eine neue Gruppe von Bereichen (Ranges) erstellt. Diese neue Gruppe von Bereichen ist ein Stripe-Set und das zuletzt erstellte Stripe-Set wird als aktuelles Stripe-Set bezeichnet. Nach dem Hinzufügen der beiden neuen Container sieht der Tabellenbereich wie in Abb. 18 auf Seite 226 aus.

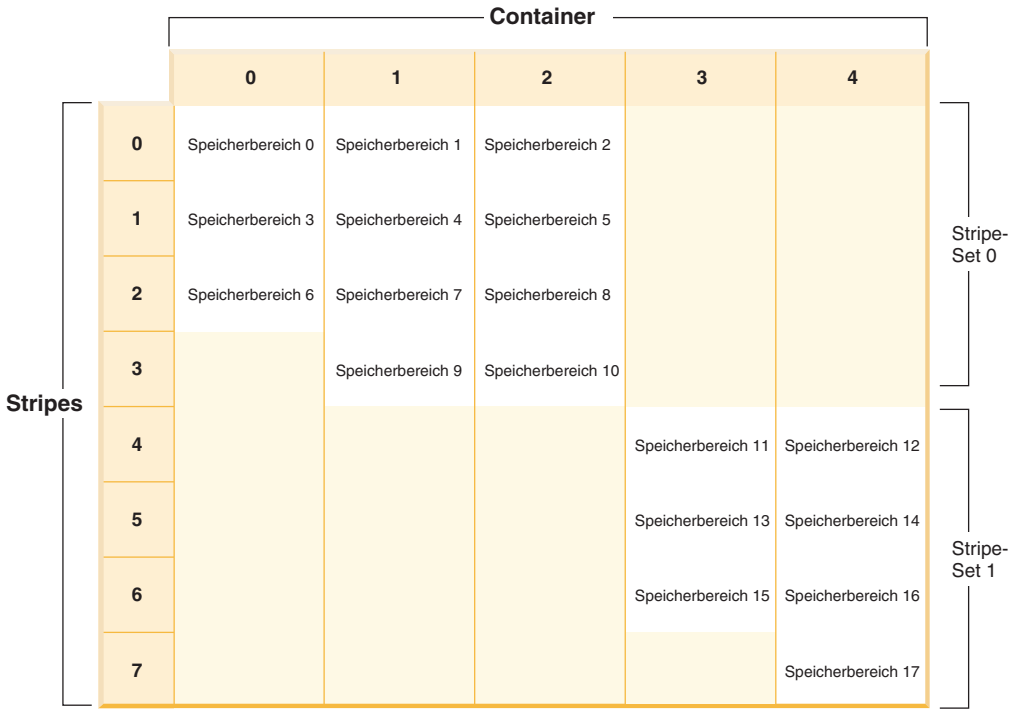


Abbildung 18. Tabellenbereich mit zwei Stripe-Sets

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]      [0]      0       8       89       0        2    0   3 (0, 1, 2)
   [1]      [0]      0      10      109       3        3    0   2 (1, 2)
   [2]      [1]      4      16      169       4        6    0   2 (3, 4)
   [3]      [1]      4      17      179       7        7    0   1 (4)
```

Wenn Sie einem Tabellenbereich neue Container hinzufügen und Sie die Klausel TO STRIPE SET mit der Klausel ADD *nicht* verwenden, werden die Container dem aktuellen Stripe-Set (d. h. dem Stripe-Set mit der höchsten Nummer) hinzugefügt. Mit der Klausel ADD TO STRIPE SET können Sie jedem Stripe-Set im Tabellenbereich Container hinzufügen. Sie müssen ein gültiges Stripe-Set angeben.

Der Datenbankmanager verwaltet die Stripe-Sets mithilfe der Tabellenbereichszuordnung. Durch das Hinzufügen von Containern ohne Neuausgleich wächst die Zuordnung in der Regel schneller als bei Durchführung eines Neuausgleichs der Container. Wenn die Tabellenbereichszuordnung zu groß wird, empfangen Sie den Fehler SQL0259N, wenn Sie versuchen, weitere Container hinzuzufügen.

Ändern der Größe von DMS-Containern

Die Größe von Containern lässt sich in Tabellenbereichen mit dynamischem Speicher nicht manuell ändern.

Der Datenbankmanager sorgt automatisch dafür, dass Container nach Bedarf erweitert oder hinzugefügt werden. Sie können jedoch die Größe eines DMS-Tabellenbereichs (d. h. eines mit der Klausel MANAGED BY DATABASE erstellten Tabellenbereichs) ändern.

Jede Roheinheit kann nur als ein Container verwendet werden. Die Roheinheit ist nach ihrer Erstellung festgelegt. Wenn Sie beabsichtigen, einen Container für eine

Roheinheit mithilfe der Option RESIZE bzw. EXTEND zu vergrößern, sollten Sie die Größe der Roheinheit zunächst überprüfen, um sicherzugehen, dass Sie nicht versuchen, den Einheitencontainer über die Größe der Roheinheit hinaus zu vergrößern.

Sie haben auch die Möglichkeit, vorhandene Container aus einem DMS-Tabellenbereich zu löschen, die Größe vorhandener Container in einem DMS-Tabellenbereich zu verringern und einem DMS-Tabellenbereich neue Container hinzuzufügen, ohne dass ein Neuausgleich der Daten über alle Container durchgeführt wird.

Das Löschen vorhandener Tabellenbereichscontainer sowie das Verkleinern vorhandener Container ist nur zulässig, wenn die Anzahl von EXTENTSIZE großen Speicherbereichen, die gelöscht oder verkleinert werden sollen, kleiner als oder gleich der Anzahl der freien EXTENTSIZE großen Speicherbereiche oberhalb der „oberen Grenze" im Tabellenbereich ist. Die obere Grenze ist die Seitennummer der höchsten zugeordneten Seite im Tabellenbereich. Diese Grenze ist nicht identisch mit dem Wert für verwendete Seiten im Tabellenbereich, weil einige der Speicherbereiche unterhalb der oberen Grenze möglicherweise zur Wiederverwendung verfügbar gemacht wurden.

Die Anzahl der freien Speicherbereiche oberhalb der oberen Grenze im Tabellenbereich spielt eine wichtige Rolle, da sich alle Speicherbereiche bis zur oberen Grenze und einschließlich der oberen Grenze an der gleichen logischen Position innerhalb des Tabellenbereichs befinden müssen. Der sich ergebende Tabellenbereich muss ausreichend Platz haben, um alle Daten aufnehmen zu können. Wenn nicht genügend freier Speicherplatz verfügbar ist, wird eine Fehlernachricht (SQL20170N, SQLSTATE 57059) zurückgegeben.

Zum Löschen von Containern wird die Option DROP in der Anweisung ALTER TABLESPACE verwendet. Beispiel:

```
ALTER TABLESPACE TS1 DROP (FILE 'datei1', DEVICE '/dev/rdisk1')
```

Zum Verkleinern vorhandener Container können Sie entweder die Option RESIZE oder die Option REDUCE verwenden. Bei Verwendung der Option RESIZE müssen alle Container, die in der Anweisung aufgelistet werden, entweder vergrößert oder verkleinert werden. Sie können nicht innerhalb derselben Anweisung einige Container vergrößern und andere Container verkleinern. Sie sollten die Methode der Größenänderung (RESIZE) in Betracht ziehen, wenn Sie die neue untere Begrenzung für die Größe des Containers kennen. Sie sollten die Methode der Verringerung (REDUCE) in Betracht ziehen, wenn Sie die aktuelle Größe des Containers nicht kennen (oder diese für Sie keine Rolle spielt).

Geben Sie die folgende Anweisung in die Befehlszeile ein, um einen oder mehrere Container in einem DMS-Tabellenbereich zu verkleinern:

```
ALTER TABLESPACE <name>
   REDUCE (FILE '<dateiname>' <größe>)
```

Das folgende Beispiel zeigt, wie ein Dateicontainer (der bereits mit einer Größe von 1.000 Seiten vorhanden ist) in einem Tabellenbereich auf einem Windows-System verkleinert wird:

```
ALTER TABLESPACE PAYROLL
  REDUCE (FILE 'd:\hldr\finance'  200)
```

Durch diese Aktion wird die Datei von 1.000 Seiten auf die Größe von 800 Seiten verkleinert.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um einen oder mehrere Container in einem DMS-Tabellenbereich zu vergrößern:

```
ALTER TABLESPACE <name>
   RESIZE (DEVICE '<pfad>' <größe>)
```

Das folgende Beispiel zeigt, wie zwei Einheitencontainer (die bereits mit einer Größe von 1.000 Seiten vorhanden sind) in einem Tabellenbereich auf einem Linux- und UNIX-System vergrößert werden:

```
 ALTER TABLESPACE HISTORY
   RESIZE (DEVICE '/dev/rhd7'  2000,
           DEVICE  '/dev/rhd8' 2000)
```

Durch diese Aktion werden die beiden Einheitencontainer von 1.000 Seiten auf 2.000 Seiten vergrößert. Der Inhalt des Tabellenbereichs kann in den Containern neu ausgeglichen werden. Während des Neuausgleichs ist der Zugriff auf den Tabellenbereich nicht eingeschränkt.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um einen oder mehrere Container in einem DMS-Tabellenbereich zu erweitern:

```
ALTER TABLESPACE <name>
   EXTEND (FILE '<dateiname>' <größe>)
```

Das folgende Beispiel zeigt, wie Dateicontainer (die jeweils in einer Größe von 1.000 Seiten bereits vorhanden sind) in einem Tabellenbereich auf einem Windows-System vergrößert werden:

```
ALTER TABLESPACE PERSNEL
   EXTEND (FILE 'e:\wrkhist1'  200
           FILE  'f:\wrkhist2'  200)
```

Durch diese Aktion werden die beiden Dateien von 1.000 Seiten auf 1.200 Seiten vergrößert. Der Inhalt des Tabellenbereichs kann in den Containern neu ausgeglichen werden. Während dieses Neuausgleichs wird der Zugriff auf den Tabellenbereich nicht eingeschränkt.

Das Hinzufügen bzw. das Ändern von DMS-Containern (sowohl von Datei- als auch von Roheinheitencontainern) wird über Vorablesefunktionen parallel ausgeführt. Um die parallele Verarbeitung der Operationen zur Erstellung oder Größenänderung von Containern zu verbessern, können Sie die Anzahl der im System ausgeführten Vorablesefunktionen erhöhen. Der einzige Prozess, der nicht parallel ausgeführt werden kann, ist das Protokollieren dieser Aktionen und im Falle des Erstellens von Containern, das Kennzeichnen der Container.

Anmerkung: Um die parallele Verarbeitung der Anweisungen CREATE TABLESPACE oder ALTER TABLESPACE zu maximieren (im Hinblick auf das Hinzufügen neuer Container für einen vorhandenen Tabellenbereich), stellen Sie sicher, dass die Anzahl der Vorablesefunktionen größer oder gleich der Anzahl der hinzugefügten Container ist. Die Anzahl der Vorablesefunktionen wird über den Datenbankkonfigurationsparameter *num_ioservers* gesteuert. Die Datenbank muss gestoppt werden, damit der neue Parameterwert in Kraft treten kann. Das heißt, dass alle Anwendungen und Benutzer die Verbindung zur Datenbank trennen müssen, damit die Änderung wirksam werden kann.

Beachten Sie, dass Sie mit der Anweisung ALTER TABLESPACE auch andere Merkmale des Tabellenbereichs ändern können, die sich auf die Leistung auswirken können.

Löschen oder Verkleinern von DMS-Containern

Bei einem DMS-Tabellenbereich ist es möglich, mit der Anweisung ALTER TABLESPACE einen Container aus dem Tabellenbereich zu löschen oder einen Container zu verkleinern.

Das Löschen oder Verkleinern eines Containers ist nur zulässig, wenn die Anzahl von EXTENTSIZE großen Speicherbereichen, die durch die Operation gelöscht werden sollen, kleiner als oder gleich der Anzahl der freien EXTENTSIZE großen Speicherbereiche oberhalb der oberen Grenze im Tabellenbereich ist. Dies ist notwendig, weil durch die Operation keine Seitennummern geändert werden können und daher alle Speicherbereiche bis zur oberen Grenze (einschließlich) an der gleichen logischen Position im Tabellenbereich verbleiben müssen. Das heißt, der resultierende Tabellenbereich muss ausreichend Platz haben, um alle Daten bis zur oberen Grenze einschließlich enthalten zu können. In dem Fall, dass nicht genügend freier Speicher verbleibt, empfangen Sie sofort nach Ausführung der Anweisung eine Fehlernachricht.

Die obere Grenze ist die Seitennummer der höchsten zugeordneten Seite im Tabellenbereich. Zum Beispiel hat ein Tabellenbereich 1000 Seiten und einen EXTENTSIZE-Wert von 10, was 100 EXTENTSIZE großen Speicherbereichen entspricht. Wenn der 42ste Speicherbereich der höchste zugeordnete Speicherbereich im Tabellenbereich ist, bedeutet dies, dass die obere Grenze bei 42 * 10 = 420 Seiten liegt. Dieser Wert ist nicht identisch mit dem Wert für verwendete Seiten, weil einige der Speicherbereiche unterhalb der oberen Grenze freigegeben worden sein könnten, sodass sie zur Wiederverwendung verfügbar sind.

Wenn Container gelöscht oder verkleinert werden, findet ein Neuausgleich statt, wenn sich Daten in dem Speicherbereich befinden, der aus dem Tabellenbereich gelöscht wird. Vor dem Start des Neuausgleichs wird eine neue Tabellenbereichszuordnung auf der Grundlage der vorgenommenen Containeränderungen generiert. Die Neuausgleichsfunktion versetzt EXTENTSIZE große Speicherbereiche von ihrer Position, die durch die aktuelle Zuordnung festgelegt ist, an die Position, die durch die neue Zuordnung festgelegt wird. Die Neuausgleichsfunktion beginnt mit dem Speicherbereich, der die obere Grenze enthält, und versetzt jeweils einen Speicherbereich gleichzeitig, bis Speicherbereich 0 versetzt wurde. Beim Versetzen der einzelnen Speicherbereiche wird die aktuelle Zuordnung stückweise in das Aussehen der neuen Zuordnung geändert. Wenn die Position eines Speicherbereichs in der aktuellen Zuordnung mit seiner Position in der neuen Zuordnung übereinstimmt, wird der Speicherbereich nicht versetzt, und es finden keine E/A-Operationen statt. Da beim Neuausgleich Speicherbereiche vom höchsten zugeordneten Speicherbereich angefangen und mit dem ersten Speicherbereich im Tabellenbereich zum Schluss versetzt werden, wird er als *umgekehrter Neuausgleich* (im Gegensatz zum *Vorwärtsneuausgleich*, der stattfindet, wenn der Tabellenbereich nach dem Hinzufügen oder Erweitern von Containern vergrößert wird) bezeichnet.

Wenn Container gelöscht werden, werden die verbleibenden Container neu durchnummeriert, sodass ihre Container-IDs bei 0 anfangen und sich jeweils um 1 erhöhen. Wenn alle Container in einem Stripe-Set gelöscht werden, wird das Stripe-Set aus der Zuordnung entfernt und alle nachfolgenden Stripe-Sets werden nach unten verschoben und neu nummeriert, sodass keine Lücken in den Nummern der Stripe-Sets auftreten.

Anmerkung: In den folgenden Beispielen wird bei den Containergrößen die Größe der Containerkennung (container tag) nicht berücksichtigt. Die Containergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Containergrößen dar. Die Beispiele zeigen Container unter-

schiedlicher Größen innerhalb eines Tabellenbereichs, jedoch dient dies nur der Veranschaulichung. Zu empfehlen ist die Verwendung von Containern gleicher Größe.

Betrachten Sie zum Beispiel einen Tabellenbereich mit drei Containern und einem EXTENTSIZE-Wert 10. Die Container sind 20, 50 und 50 Seiten (d. h. 2, 5 und 5 EXTENTSIZE-Größen groß). Der Tabellenbereich lässt sich wie in Abb. 19 darstellen.

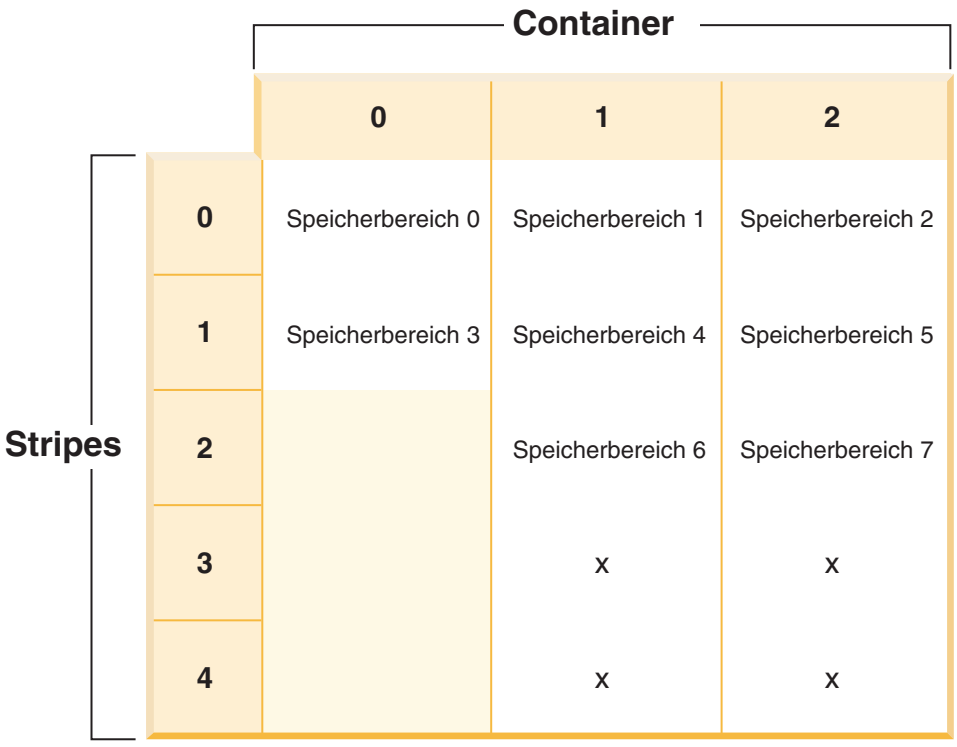


Abbildung 19. Tabellenbereich mit zwölf Speicherbereichen und vier Speicherbereichen ohne Daten

Ein X zeigt an, dass an der Stelle ein Speicherbereich vorhanden ist, der jedoch keine Daten enthält.

Wenn Sie den Container 0 löschen, der zwei Speicherbereiche enthält, müssen über der oberen Grenze mindestens zwei freie Speicherbereiche vorhanden sein. Die obere Grenze ist der Speicherbereich 7, sodass vier freie Speicherbereiche verbleiben. In diesem Fall können Sie also den Container 0 löschen.

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0       5       59        0        1    0   3 (0, 1, 2)
   [1]      [0]       0      11      119        2        4    0   2 (1, 2)
```

Nach dem Löschen enthält der Tabellenbereich nur Container 0 und Container 1. Der neue Tabellenbereich lässt sich wie in Abb. 20 darstellen.

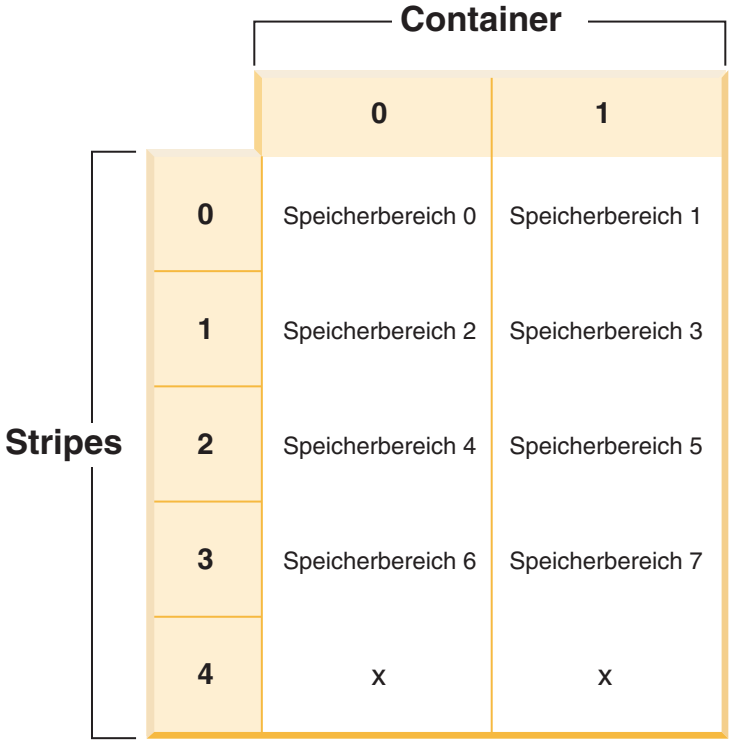


Abbildung 20. Tabellenbereich nach dem Löschen eines Containers

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       9       99        0       4    0   2 (0, 1)
```

Wenn Sie die Größe eines Containers verringern wollen, arbeitet die Neuausgleichsfunktion in ähnlicher Weise.

Verwenden Sie zur Verkleinerung eines Containers die Option REDUCE oder RESIZE in der Anweisung ALTER TABLESPACE. Zum Löschen eines Containers dient die Option DROP in der Anweisung ALTER TABLESPACE.

Ändern von Tabellenbereichen mit dynamischem Speicher

Bei Verwendung von Tabellenbereichen mit dynamischem Speicher können Sie nur die Größe der Container verringern.

Dieser Prozess stimmt mit dem in „Ändern der Größe von DMS-Containern" auf Seite 226 beschriebenen Prozess überein.

Umbenennen eines Tabellenbereichs

Zum Umbenennen eines Tabellenbereichs verwenden Sie die Anweisung RENAME TABLESPACE.

Der Tabellenbereich SYSCATSPACE kann nicht umbenannt werden. Tabellenbereiche, die sich im Status 'Aktualisierende Recovery anstehend' oder 'Aktualisierende Recovery wird ausgeführt' befinden, können nicht umbenannt werden.

Beim Restore eines Tabellenbereichs, der seit seinem Backup umbenannt wurde, müssen Sie im Befehl RESTORE DATABASE den neuen Tabellenbereichsnamen verwenden. Wenn Sie den vorherigen Tabellenbereichsnamen benutzen, kann der Tabellenbereich nicht lokalisiert werden. Wenn Sie für den Tabellenbereich mit dem Befehl ROLLFORWARD DATABASE eine aktualisierende Recovery durchführen, müssen Sie ebenfalls den neuen Namen verwenden. Wenn Sie den vorherigen Tabellenbereichsnamen benutzen, kann der Tabellenbereich nicht lokalisiert werden.

Sie können einem vorhandenen Tabellenbereich einen neuen Namen zuordnen, ohne dass sich dies auf die einzelnen Objekte innerhalb des Tabellenbereichs auswirkt. Beim Umbenennen eines Tabellenbereichs werden alle Katalogsätze geändert, die auf diesen Tabellenbereich verweisen.

Umschalten von Tabellenbereichen vom Offlinestatus in den Onlinestatus

Die Klausel SWITCH ONLINE der Anweisung ALTER TABLESPACE kann verwendet werden, um den Status OFFLINE von einem Tabellenbereich zu entfernen, wenn auf die Container, die diesem Tabellenbereich zugeordnet sind, zugegriffen werden kann.

Der Status OFFLINE wird vom Tabellenbereich entfernt, während der Rest der Datenbank weiterhin aktiv ist und verwendet wird.

Eine Alternative zur Verwendung dieser Klausel ist das Trennen aller Anwendungen von der Datenbank und das anschließende erneute Verbinden der Anwendungen mit der Datenbank. Dadurch wird der Status OFFLINE vom Tabellenbereich entfernt.

Geben Sie in der Befehlszeile Folgendes ein, um den Status OFFLINE vom Tabellenbereich zu entfernen.

```
db2 ALTER TABLESPACE <name>
   SWITCH ONLINE
```

Optimieren der Leistung von Tabellenbereichen bei Datenspeicherung auf RAID-Einheiten

Befolgen Sie diese Richtlinien, um die Leistung zu optimieren, wenn Daten auf RAID-Einheiten (Redundant Array of Independent Disks) gespeichert werden.

1. Wenn Sie einen Tabellenbereich auf einer Gruppe von RAID-Einheiten erstellen, erstellen Sie die Container für einen vorgegebenen Tabellenbereich (SMS oder DMS) auf separaten Einheiten.

   Angenommen Sie verfügen über fünfzehn 146-GB-Platten, die als drei RAID-5-Arrays mit fünf Platten in jedem Array konfiguriert sind. Nach der Formatie-

rung kann jede Platte ungefähr 136 GB an Daten enthalten. Jeder Array kann daher ungefähr 544 GB (4 aktive Platten x 136 GB) speichern. Wenn Sie über einen Tabellenbereich verfügen, der einen Speicher von 300 GB benötigt, erstellen Sie drei Container, und stellen Sie jeden Container auf eine separate Einheit. Jeder Container belegt 100 GB (300 GB/3) auf einer Einheit, und es verbleiben 444 GB (544 GB - 100 GB) auf jeder Einheit für weitere Tabellenbereiche.

2. Wählen Sie eine angemessene Speicherbereichsgröße für die Tabellenbereiche aus. Die Speicherbereichsgröße für einen Tabellenbereich ist das Datenvolumen, das der Datenbankmanager in einen Container schreibt, bevor er Daten in den nächsten Container schreibt. Idealerweise sollte die Speicherbereichsgröße ein Vielfaches von der zugrunde liegenden Segmentgröße der Platten sein, wobei die Segmentgröße das Datenvolumen ist, das der Plattencontroller auf eine physische Platte schreibt, bevor er Daten auf die nächste physische Platte schreibt. Die Auswahl einer Speicherbereichsgröße, die ein Vielfaches der Segmentgröße bildet, stellt sicher, dass die auf dem Speicherbereich basierenden Operationen, wie z. B. paralleles sequenzielles Lesen beim Vorablesezugriff, nicht um dieselben physischen Platten konkurrieren. Wählen Sie außerdem eine Speicherbereichsgröße aus, die ein Vielfaches der Seitengröße ist.

   Wenn die Segmentgröße im Beispiel 64 KB beträgt und die Seitengröße 16 KB, könnten 256 KB eine angemessene Speicherbereichsgröße ausmachen.

3. Verwenden Sie die Registrierdatenbankvariable DB2_PARALLEL_IO, um die parallele Ein-/Ausgabe für alle Tabellenbereiche zu aktivieren und um die Anzahl physischer Platten pro Container anzugeben.

   Legen Sie für die Situation im Beispiel Folgendes fest: DB2_PARALLEL_IO = *:4.

   Wenn Sie die Vorablesezugriffsgröße eines Tabellenbereichs auf AUTOMATIC setzen, verwendet der Datenbankmanager den Wert von der Anzahl physischer Platten, die Sie für DB2_PARALLEL_IO angegeben haben, um den Wert für die Vorablesezugriffsgröße festzulegen. Wenn die Vorablesezugriffsgröße nicht auf AUTOMATIC gesetzt ist, können Sie sie manuell festlegen, berücksichtigen Sie dabei die RAID-Stripegröße, die der Wert der Segmentgröße multipliziert mit der Anzahl aktiver Platten ist. Wählen Sie einen Wert für die Vorablesezugriffsgröße aus, der die folgenden Bedingungen erfüllt:

   - Er entspricht dem Produkt aus der RAID-Stripegröße multipliziert mit der Anzahl paralleler RAID-Einheiten (oder einer ganzzahligen Darstellung dieses Produkts).
   - Er ist eine ganzzahlige Darstellung der Speicherbereichsgröße.

   Im Beispiel setzen Sie die Vorablesezugriffsgröße möglicherweise auf 768 KB. Dieser Wert entspricht dem Produkt aus der RAID-Stripegröße (256 KB) multipliziert mit der Anzahl paralleler RAID-Einheiten (3). Er ist auch ein Vielfaches der Speicherbereichsgröße (256 KB). Die Auswahl dieser Vorablesezugriffsgröße bedeutet, dass ein einzelner Vorablesezugriff alle Platten in allen Arrays betrifft. Wenn Sie wollen, dass die Vorablesefunktionen intensiver ausgeführt werden, weil Ihre Auslastung hauptsächlich sequenzielle Suchen umfasst, können Sie stattdessen ein Vielfaches dieses Werts verwenden, wie z. B. 1536 KB (768 KB x 2).

4. Setzen Sie nicht die Registrierdatenbankvariable DB2_USE_PAGE_CONTAINER_TAG. Wie weiter oben beschrieben, sollten Sie einen Tabellenbereich mit einem Wert für EXTENTSIZE erstellen, der der RAID-Stripegröße oder einem Vielfachen dieser Größe entspricht. Wenn DB2_USE_PAGE_CONTAINER_TAG jedoch auf den Wert ON gesetzt wird, wird eine Containerkennung in der Größe einer Seite verwendet und die EXTENTSIZE-Bereiche richten sich nicht nach den einheitenübergreifend

gespeicherten RAID-Datenblöcken (Stripes) aus. Infolgedessen kann bei einer E/A-Anforderung ein Zugriff auf mehr als die optimale Anzahl physischer Platten erforderlich werden.

Löschen von Tabellenbereichen

Beim Löschen eines Tabellenbereichs werden alle Daten in diesem Tabellenbereich gelöscht, die Container freigegeben, die Katalogeinträge entfernt und alle Objekte, die in dem Tabellenbereich definiert sind, entweder gelöscht oder als ungültig markiert.

Die Container in einem leeren Tabellenbereich können erneut verwendet werden, indem der Tabellenbereich gelöscht wird. Allerdings muss die Anweisung DROP TABLESPACE mit COMMIT festgeschrieben werden, bevor Sie versuchen, die Container erneut zu verwenden.

Anmerkung: Sie können einen Tabellenbereich nicht löschen, ohne nicht auch alle Tabellenbereiche zu löschen, die diesem zugeordnet sind. Wenn Sie zum Beispiel eine Tabelle in einem Tabellenbereich haben und der zugehörige Index in einem anderen Tabellenbereich erstellt wurde, müssen Sie sowohl den Indextabellenbereich als auch den Datentabellenbereich in einem Befehl DROP TABLESPACE löschen.

Löschen von Benutzertabellenbereichen

Sie können einen Benutzertabellenbereich löschen, der sämtliche Tabellendaten einschließlich Index- und LOB-Daten innerhalb dieses einen Benutzertabellenbereichs enthält. Sie können auch einen Benutzertabellenbereich löschen, der möglicherweise Tabellen enthält, die sich über mehrere Tabellenbereiche erstrecken. Dabei können sich die Tabellendaten in einem Tabellenbereich, die Indizes in einem anderen Tabellenbereich und die LOB-Daten in einem dritten Tabellenbereich befinden. Sie müssen alle drei Tabellenbereiche gleichzeitig mit einer einzigen Anweisung löschen. Wenn sich Tabellen über mehrere Tabellenbereiche erstrecken, muss die Löschanweisung alle Tabellenbereiche enthalten, auf die sich diese Tabellen erstrecken, sonst kann sie nicht erfolgreich ausgeführt werden.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um einen Benutzertabellenbereich zu löschen:

```
DROP TABLESPACE <name>
```

Mit der folgenden SQL-Anweisung wird der Tabellenbereich ACCOUNTING gelöscht:

```
DROP TABLESPACE ACCOUNTING
```

Löschen von Tabellenbereichen für temporäre Benutzertabellen

Ein Tabellenbereich für temporäre Benutzertabellen kann nur gelöscht werden, wenn in diesem Tabellenbereich momentan keine deklarierten temporären Tabellen definiert sind. Wenn Sie den Tabellenbereich löschen, wird nicht versucht, alle deklarierten temporären Tabellen dieses Tabellenbereichs zu löschen.

Anmerkung: Eine deklarierte temporäre Tabelle wird implizit gelöscht, wenn die Anwendung, die zur Deklaration verwendet wurde, die Verbindung zur Datenbank trennt.

Löschen von Tabellenbereichen für temporäre Systemtabellen

Sie können einen Tabellenbereich für temporäre Systemtabellen, der eine Seitengröße von 4 KB hat, nur löschen, wenn Sie zuvor einen anderen Tabellenbereich für temporäre Systemtabellen erstellt haben. Der neue Tabellenbereich für temporäre Systemtabellen muss eine Seitengröße von 4 KB haben, weil die Datenbank immer über mindestens einen Tabellenbereich für temporäre Systemtabellen verfügen muss, der eine Seitengröße von 4 KB hat. Wenn Sie zum Beispiel über einen einzigen Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von 4 KB verfügen und diesem einen Container hinzufügen möchten und es sich um einen SMS-Tabellenbereich handelt, müssen Sie zunächst einen neuen Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von 4 KB mit der entsprechenden Anzahl von Containern hinzufügen, und anschließend den alten Tabellenbereich für temporäre Systemtabellen löschen. (Wenn Sie mit DMS-Tabellenbereichen arbeiten, können Sie einen Container hinzufügen, ohne den Tabellenbereich löschen und erneut erstellen zu müssen.)

Die Standardseitengröße für Tabellenbereiche ist die Seitengröße, mit der die Datenbank erstellt wurde (standardmäßig 4 KB, kann jedoch auch 8 KB, 16 KB oder 32 KB sein).

Die folgende Anweisung dient zur Erstellung eines Tabellenbereichs für temporäre Systemtabellen:

```
CREATE SYSTEM TEMPORARY TABLESPACE <name>
  MANAGED BY SYSTEM USING ('<verzeichnisse>')
```

Geben Sie anschließend zum Löschen eines Systemtabellenbereichs in die Befehlszeile die folgende Anweisung ein:

```
DROP TABLESPACE <name>
```

Mit der folgenden SQL-Anweisung können Sie einen neuen Tabellenbereich für temporäre Systemtabellen mit dem Namen TEMPSPACE2 erstellen:

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
  MANAGED BY SYSTEM USING ('d:\systemp2')
```

Nach der Erstellung von TEMPSPACE2 können Sie den ursprünglichen Tabellenbereich für temporäre Systemtabellen TEMPSPACE1 mit dem folgenden Befehl löschen:

```
DROP TABLESPACE TEMPSPACE1
```

Kapitel 10. Schemata

Ein *Schema* ist eine Sammlung von benannten Objekten. Es stellt eine Möglichkeit zur logischen Gruppierung dieser Objekte bereit. Ein Schema ist außerdem ein Qualifikationsmerkmal für Namen. Es stellt eine Möglichkeit zur Verwendung desselben natürlichen Namens für mehrere Objekte sowie zur Vermeidung mehrdeutiger Verweise auf diese Objekte bereit.

Beispiel: Mithilfe der Schemanamen 'INTERNAL' und 'EXTERNAL' ist die Unterscheidung zweier unterschiedlicher SALES-Tabellen (INTERNAL.SALES, EXTERNAL.SALES) einfach.

Mit Schemata können auch mehrere Anwendungen Daten in einer einzigen Datenbank speichern, ohne dass es zu Namensbereichskollisionen kommt.

Ein Schema unterscheidet sich von einem *XML-Schema* und darf mit einem solchen nicht verwechselt werden. Ein XML-Schema ist ein Standard, der die Struktur beschreibt und den Inhalt von XML-Dokumenten prüft.

Ein Schema kann Tabellen, Sichten, Kurznamen, Trigger, Funktionen, Pakete und weitere Objekte enthalten. Ein Schema selbst ist eine Datenbankobjekt. Es wird explizit mit der Anweisung CREATE SCHEMA erstellt, wobei der aktuelle Benutzer oder eine angegebene Berechtigungs-ID als Schemaeigner erfasst wird. Es kann auch implizit beim Erstellen eines anderen Objekts erstellt werden, wenn der Benutzer über die Berechtigung IMPLICIT_SCHEMA verfügt.

Ein *Schemaname* wird als höherwertige Komponente eines zweiteiligen Objektnamens verwendet. Wenn das Objekt bei der Erstellung speziell mit einem Schemanamen qualifiziert wird, wird das Objekt dem Schema zugeordnet. Wenn bei der Erstellung des Objekts kein Schemaname angegeben wird, wird der Standardschemaname verwendet (angegeben im Sonderregister CURRENT SCHEMA).

Beispiel: Ein Benutzer mit der Berechtigung DBADM erstellt ein Schema mit dem Namen C für den Benutzer A:

```
CREATE SCHEMA C AUTHORIZATION A
```

Der Benutzer A kann dann die folgende Anweisung zum Erstellen einer Tabelle mit dem Namen X im Schema C absetzen (vorausgesetzt, dass der Benutzer A über die Datenbankberechtigung CREATETAB verfügt):

```
CREATE TABLE C.X (COL1 INT)
```

Einige Schemanamen sind reserviert. Integrierte Funktionen z. B. gehören zum Schema SYSIBM, und die vorinstallierten benutzerdefinierten Funktionen gehören zum Schema SYSFUN.

Wenn eine Datenbank erstellt wird, verfügen alle Benutzer über die Berechtigung IMPLICIT_SCHEMA, sofern sie nicht mit der Option RESTRICTIVE erstellt wird. Mit dieser Berechtigung erstellen Benutzer implizit ein Schema, wenn sie ein Objekt mit einem Schemanamen erstellen, der nicht bereits existiert. Bei der impliziten Erstellung von Schemata werden CREATEIN-Zugriffsrechte erteilt, mit denen alle Benutzer weitere Objekte in diesem Schema erstellen können. Die Funktionalität, Objekte wie z. B. Aliasnamen, einzigartige Datentypen, Funktionen und Trigger

zu erstellen, wurde auf implizit erstellte Schemata ausgeweitet. Die Standardzugriffsrechte für ein implizit erstelltes Schema stellen die Abwärtskompatibilität mit älteren Versionen bereit.

Wenn PUBLIC die Berechtigung IMPLICIT_SCHEMA entzogen wird, können Schemata mithilfe der Anweisung CREATE SCHEMA explizit bzw. durch Benutzer implizit erstellt werden (z. B. durch die Benutzer mit der Berechtigung DBADM), denen die Berechtigung IMPLICIT_SCHEMA erteilt wurde. Zwar wird durch das Entziehen der Berechtigung IMPLICIT_SCHEMA von PUBLIC die Kontrolle über die Verwendung von Schemanamen verschärft, aber es kann zu Berechtigungsfehlern kommen, wenn vorhandene Anwendungen Objekte erstellen.

Schemata verfügen ebenfalls über Zugriffsrechte, mit denen Schemaeigner steuern können, welche Benutzer das Zugriffsrecht zum Erstellen, Ändern, Kopieren und Löschen von Objekten in dem Schema haben. Dadurch kann die Bearbeitung einer Untergruppe von Objekten in der Datenbank gesteuert werden. Ein Schemaeigner erhält anfangs alle diese Zugriffsrechte für das Schema zusammen mit der Möglichkeit, diese Zugriffsrechte anderen Personen zu erteilen. Ein implizit erstelltes Schema gehört dem System, und alle Benutzer erhalten anfangs das Zugriffsrecht zum Erstellen von Objekten in einem solchen Schema. Ein Benutzer mit der Berechtigung SYSADM oder DBADM kann die Zugriffsrechte ändern, über die Benutzer eines beliebigen Schemas verfügen. Somit kann der Zugriff zum Erstellen, Ändern, Kopieren und Löschen von Objekten in einem beliebigen Schema (auch in einem implizit erstellten Schema) gesteuert werden.

Entwerfen von Schemata

Wenn Sie Ihre Daten in Tabellen organisieren, kann es vorteilhaft sein, die Tabellen und andere andere zugehörige Objekte in Gruppen zusammenzufassen. Dies geschieht durch Definieren eines Schemas mithilfe der Anweisung CREATE SCHEMA.

Informationen zu dem Schema werden in den Systemkatalogtabellen der Datenbank gespeichert, mit der Sie verbunden sind. Wenn weitere Objekte erstellt werden, können diese in den Schemata, die Sie erstellen, angelegt werden. Es ist jedoch zu beachten, dass sich ein Objekt jeweils nur in einem Schema befinden kann.

Schemata lassen sich mit Verzeichnissen vergleichen, wobei das aktuelle Schema dem aktuellen Verzeichnis entspricht. Nach dieser Analogie entspricht die Anweisung SET SCHEMA dem Befehl 'CD' (Change Directory - Verzeichnis wechseln).

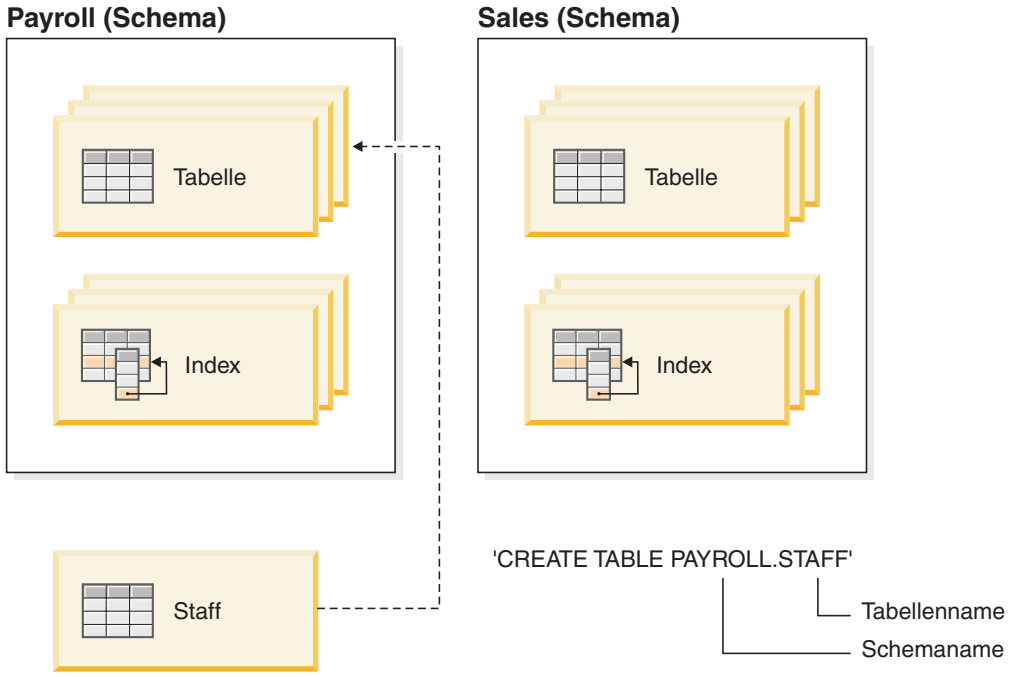
Wichtig: Es ist wichtig zu wissen, dass es keine Beziehung zwischen Berechtigungs-IDs und Schemata gibt mit Ausnahme der Standardeinstellung CURRENT SCHEMA (siehe Beschreibung unten).

Beim Entwerfen Ihrer Datenbanken und Tabellen sollten Sie sich außerdem Gedanken zu den Schemata in Ihrem System machen. Dazu gehört zum Beispiel die Auswahl von Namen für die Schemata sowie der Objekte, die den einzelnen Schemata zugeordnet werden sollen.

Den meisten Objekten in einer Datenbank wird ein eindeutiger Name zugeordnet, der aus zwei Teilen besteht. Der erste (linke) Teil wird als Qualifikationsmerkmal (Qualifier) oder Schema bezeichnet. Der zweite (rechte) Teil ist der einfache (bzw. unqualifizierte) Name. Syntaktisch werden diese beiden Teile zu einer, durch einen Punkt getrennten Zeichenfolge verkettet. Wenn ein Objekt, das durch einen

Schemanamen qualifiziert werden kann (z. B. Tabelle, Index, Sicht, benutzerdefinierter Datentyp, benutzerdefinierte Funktion, Kurzname, Paket oder Trigger) zu Anfang erstellt wird, wird ihm ein bestimmtes Schema auf der Grundlage des Qualifikationsmerkmals in seinem Namen zugeordnet.

Das folgende Diagramm veranschaulicht zum Beispiel, wie eine Tabelle während des Prozesses der Tabellenerstellung einem bestimmten Schema zugeordnet wird:



Sie sollten außerdem wissen, wie der Schemazugriff erteilt wird, um Ihren Benutzern die entsprechende Berechtigung und zutreffende Anweisungen geben zu können:

Schemanamen

Wenn ein neues Schema erstellt wird, darf der Name weder mit einem bereits im Katalog beschriebenen Schemanamen identisch sein noch mit der Buchstabenfolge ″SYS″ beginnen. Weitere Einschränkungen und Empfehlungen finden Sie in „Einschränkungen und Empfehlungen zu Schemanamen” auf Seite 242.

Zugriff auf Schemata

Ein Zugriff ohne Angabe eines Schemas als Qualifikationsmerkmal auf Objekte in einem Schema ist nicht zulässig, da durch das Schema die Eindeutigkeit in der Datenbank sichergestellt wird. Angesichts der Möglichkeit, dass zwei Benutzer zwei Tabellen (oder andere Objekte) mit demselben Namen erstellen könnten, leuchtet dies ein. Ohne Schema, das für Eindeutigkeit sorgt, gäbe es Mehrdeutigkeiten, wenn ein dritter Benutzer versuchte, die Tabelle abzufragen. Ohne weitere Qualifikationsmerkmale ist es unmöglich, die zu verwendende Tabelle zu ermitteln.

Der Benutzer, der durch die Anweisung CREATE SCHEMA erstellte Objekte definiert hat, ist der Eigner des Schemas. Dieser Eigner kann Zugriffsrechte auf Schemata anderen Benutzern erteilen (GRANT) und entziehen (REVOKE).

Wenn ein Benutzer die Berechtigung SYSADM oder DBADM hat, kann er ein Schema mit einem beliebigen gültigen Namen erstellen. Bei der Erstellung einer Datenbank wird die Berechtigung IMPLICIT_SCHEMA der Gruppe PUBLIC (d. h. allen Benutzern) erteilt.

Wenn Benutzer nicht über die Berechtigung IMPLICIT_SCHEMA oder DBADM verfügen, können sie nur ein Schema erstellen, das den gleichen Namen wie ihre eigene Berechtigungs-ID hat.

Standardschema

Wenn kein Schema oder Qualifikationsmerkmal als Teil des Namens des zu erstellenden Objekts angegeben wird, wird dieses Objekt dem Standardschema zugeordnet, das im Sonderregister CURRENT SCHEMA angegeben ist. Der Standardwert dieses Sonderregisters ist der Wert der Berechtigungs-ID der Sitzung.

Ein Standardschema wird von nicht qualifizierten Objektverweisen in dynamischen Anweisungen benötigt. Sie können ein Standardschema für eine bestimmte DB2-Verbindung festlegen, indem Sie das Sonderregister CURRENT SCHEMA auf das Schema setzen, das als Standardschema verwendet werden soll. Zum Setzen dieses Sonderregisters ist keine bestimmte Berechtigung erforderlich, sodass jeder beliebige Benutzer das Sonderregister CURRENT SCHEMA festlegen kann.

Die Syntax der Anweisung SET SCHEMA sieht wie folgt aus:

```
SET SCHEMA = <schemaname>
```

Sie können diese Anweisung interaktiv oder aus einer Anwendung heraus absetzen. Der Anfangswert des Sonderregisters CURRENT SCHEMA entspricht der Berechtigungs-ID des Benutzers der aktuellen Sitzung. Weitere Informationen zu diesem Thema finden Sie in den Informationen zur Anweisung SET SCHEMA.

Anmerkung:

- Es gibt darüber hinaus noch weitere Methoden, um das Standardschema beim Verbindungsaufbau festzulegen. Dazu können zum Beispiel die Datei cli.ini für CLI/ODBC-Anwendungen oder die Verbindungseigenschaften für die JDBC-Programmierschnittstelle (Java Database Connectivity) verwendet werden.
- Der Standardschemasatz wird nicht in den Systemkatalogen erstellt, sondern ist nur als Wert vorhanden, den der Datenbankmanager jederzeit (aus dem Sonderregister CURRENT SCHEMA) abrufen kann, wenn kein Schema oder Qualifikationsmerkmal als Teil des Namens eines zu erstellenden Objekts angegeben wird.

Implizite Erstellung

Sie können Schemata implizit erstellen, wenn Sie über die Berechtigung IMPLICIT_SCHEMA verfügen. Mit dieser Berechtigung können Sie ein Schema implizit erstellen, wenn Sie ein Objekt mit einem Schemanamen erstellen, der noch nicht vorhanden ist. Häufig werden Schemata implizit erstellt, wenn zum ersten Mal ein Datenobjekt in einem Schema erstellt wird, sofern der Benutzer, der das Objekt erstellt, die Berechtigung IMPLICIT_SCHEMA besitzt.

Explizite Erstellung

Schemata können darüber hinaus auch explizit erstellt und gelöscht werden. Dazu muss die Anweisung CREATE SCHEMA bzw. DROP SCHEMA

über die Befehlszeile oder aus einem Anwendungsprogramm heraus ausgeführt werden. Weitere Informationen zu diesem Thema finden Sie in den Informationen zu den Anweisungen CREATE SCHEMA und DROP SCHEMA.

Nach Schema definierte Aliasnamen für Tabellen und Sichten

Wenn ein anderer Benutzer in der Lage sein soll, auf eine Tabelle oder Sicht zuzugreifen, ohne den entsprechenden Schemanamen als Teil der Qualifikation des Tabellen- oder Sichtnamens einzugeben, muss ein Aliasname für diesen Benutzer erstellt werden. Die Definition des Aliasnamens muss den vollständig qualifizierten Tabellen- oder Sichtnamen einschließlich des Schemas des Benutzers definieren. Anschließend führt der Benutzer seine Abfrage einfach unter Verwendung des Aliasnamens aus. Der Aliasname muss durch das Schema des Benutzers als Teil der Aliasnamensdefinition vollständig qualifiziert werden.

Gruppieren von Objekten nach Schema

Datenbankobjektnamen können aus einer einzigen Kennung bestehen; sie können aber auch *über ein Schema qualifizierte Objekte* mit zwei Kennungen sein. Das Schema, d. h. die höherwertige Komponente, eines derart qualifizierten Objekts stellt eine Methode bereit, Objekte in der Datenbank zu klassifizieren oder zu gruppieren. Wenn ein Objekt wie eine Tabelle, eine Sicht, ein Aliasname, ein benutzerdefinierter Datentyp, eine Funktion, ein Index, ein Paket oder ein Trigger erstellt wird, wird es einem Schema zugeordnet. Diese Zuordnung erfolgt entweder explizit oder implizit.

Eine explizite Verwendung des Schemas liegt vor, wenn Sie die höherwertige Komponente eines zweiteiligen Objektnamens beim Verweisen auf das Objekt in einer Anweisung verwenden. Der Benutzer A führt zum Beispiel eine Anweisung CREATE TABLE in Schema C wie folgt aus:

```
CREATE TABLE C.X (COL1 INT)
```

Eine implizite Verwendung des Schemas liegt vor, wenn Sie die höherwertige Komponente eines zweiteiligen Objektnamens nicht verwenden. Wenn dies der Fall ist, wird anhand des Sonderregisters CURRENT SCHEMA der Schemaname ermittelt, der als höherwertige Komponente des Objektnamens ergänzt wird. Der Anfangswert von CURRENT SCHEMA ist die Berechtigungs-ID des Benutzers der aktuellen Sitzung. Wenn Sie diesen Wert während der aktuellen Sitzung ändern möchten, können Sie für das Sonderregister mit der Anweisung SET SCHEMA einen anderen Schemanamen definieren.

Bei der Erstellung der Datenbank werden einige Objekte innerhalb bestimmter Schemata erstellt und in den Systemkatalogtabellen gespeichert.

Sie müssen nicht explizit angeben, in welchem Schema ein Objekt erstellt werden soll; wenn es nicht angegeben ist, wird die Berechtigungs-ID der Anweisung verwendet. Beispiel: Für die folgende CREATE TABLE-Anweisung nimmt der Schemaname standardmäßig den Wert der Berechtigungs-IDs an, die momentan angemeldet ist (d. h., der Wert des Sonderregisters CURRENT SCHEMA):

```
CREATE TABLE X (COL1 INT)
```

Bei dynamischen SQL- und XQuery-Anweisungen wird in der Regel der Wert des Sonderregisters CURRENT SCHEMA verwendet, um implizit sämtliche Objektnamenverweise ohne Qualifikationsmerkmal zu qualifizieren.

Bevor Sie eigene Objekte erstellen, müssen Sie entscheiden, ob Sie sie im eigenen Schema erstellen wollen oder ob Sie ein anderes Schema verwenden wollen, das die Objekte logisch gruppiert. Wenn Objekte erstellt werden, die gemeinsam benutzt werden sollen, kann die Verwendung eines anderen Schemanamens sehr vorteilhaft sein.

Einschränkungen und Empfehlungen zu Schemanamen

Bei der Benennung von Schemata müssen Sie einige Einschränkungen und Empfehlungen berücksichtigen.

- Benutzerdefinierte Typen (UDTs) dürfen keine Schemanamen verwenden, deren Längen die in SQL- und XML-Begrenzungen aufgeführte Schemalänge überschreiten.
- Die folgenden Schemanamen sind reservierte Wörter und dürfen nicht verwendet werden: SYSCAT, SYSFUN, SYSIBM, SYSSTAT, SYSPROC.
- Um mögliche Probleme bei einer zukünftigen Migration auszuschließen, sollten Sie keine Schemanamen verwenden, die mit der Zeichenfolge SYS beginnen. Der Datenbankmanager lässt die Erstellung von Triggern, benutzerdefinierten Typen oder benutzerdefinierten Funktionen, die einen mit SYS beginnenden Schemanamen verwenden, nicht zu.
- Es wird empfohlen, das Wort SESSION nicht als Schemanamen zu verwenden. Deklarierte temporäre Tabellen müssen durch SESSION qualifiziert werden. Daher kann es vorkommen, dass eine Anwendung eine temporäre Tabelle mit einem Namen deklariert, der mit dem einer persistenten Tabelle identisch ist. In diesem Fall kann die Anwendungslogik zu komplex werden. Vermeiden Sie die Verwendung des Schemas SESSION, außer wenn Sie mit deklarierten temporären Tabellen arbeiten.

Erstellen von Schemata

Mithilfe von Schemata können Sie Objekte bei ihrer Erstellung gruppieren. Ein Objekt kann nur zu einem Schema gehören. Zur Erstellung von Schemata verwenden Sie die Anweisung CREATE SCHEMA. Informationen zu den Schemata werden in den Systemkatalogtabellen der Datenbank gespeichert, zu der die Verbindung hergestellt wurde.

Wenn Sie ein Schema erstellen und einen anderen Benutzer optional zum Eigner dieses Schemas machen möchten, benötigen Sie dazu die Berechtigung SYSADM oder DBADM. Wenn Sie keine dieser beiden Berechtigungen besitzen, können Sie dennoch ein Schema mit Ihrer eigenen Berechtigungs-ID erstellen. Der Benutzer, der beliebige Objekte definiert, die als Teil der Anweisung CREATE SCHEMA erstellt werden, ist der Schemaeigner. Dieser Eigner kann Schemazugriffsrechte anderen Benutzern erteilen (GRANT) und entziehen (REVOKE).

Geben Sie die folgende Anweisung in die Befehlszeile ein, um ein Schema zu erstellen:

```
CREATE SCHEMA <schemaname> [ AUTHORIZATION <name_des_schemaeigners> ]
```

Dabei ist <schemaname> der Name des Schemas. Dieser Name muss unter den bereits im Katalog erfassten Schemata eindeutig sein und darf nicht mit der Zeichenfolge SYS beginnen. Wenn die optionale Klausel AUTHORIZATION angegeben wird, wird die in <name_des_schemaeigners> angegebene Berechtigungs-ID Eigner des Schemas. Fehlt diese Klausel, wird die Berechtigungs-ID, unter der diese Anweisung abgesetzt wurde, zum Eigner des Schemas.

Weitere Informationen finden Sie in der Beschreibung der Anweisung CREATE SCHEMA. Siehe auch „Einschränkungen und Empfehlungen zu Schemanamen" auf Seite 242.

Kopieren von Schemata

Das Dienstprogramm 'db2move' und die Prozedur ADMIN_COPY_SCHEMA ermöglichen Ihnen eine schnelle Erstellung von Kopien Ihres Datenbankschemas. Nach der Einrichtung eines Modellschemas können Sie dieses als Vorlage für die Erstellung neuer Versionen verwenden.

Zum Kopieren eines einzelnen Schemas innerhalb derselben Datenbank können Sie die Prozedur ADMIN_COPY_SCHEMA verwenden. Zum Kopieren eines oder mehrerer Schemata von einer Quellen- in eine Zieldatenbank können Sie das Dienstprogramm db2move mit der Aktion `COPY -co` verwenden. Die meisten Datenbankobjekte aus dem Quellenschema werden unter das neue Schema in der Zieldatenbank kopiert.

Tipps zur Fehlerbehebung

Sowohl die Prozedur ADMIN_COPY_SCHEMA als auch das Dienstprogramm db2move rufen den Befehl LOAD auf. Während der Verarbeitung des Ladevorgangs werden die Tabellenbereiche, in denen sich die Zieldatenbankobjekte befinden, in den Status 'Backup anstehend' versetzt.

Prozedur ADMIN_COPY_SCHEMA

Durch die Verwendung dieser Prozedur mit der Option COPYNO werden die Tabellenbereiche, in denen sich das Zielobjekt befindet, in den Status 'Backup anstehend' versetzt, wie in der obigen Anmerkungen beschrieben. Um den Tabellenbereich aus dem Status 'Festlegen der Integrität anstehend' herauszunehmen, setzt diese Prozedur eine Anweisung SET INTEGRITY ab. In Situationen, in denen ein Zieltabellenobjekt definierte referenzielle Integritätsbedingungen besitzt, wird die Zieltabelle ebenfalls in den Status 'Festlegen der Integrität anstehend' versetzt. Da sich die Tabellenbereiche bereits im Status 'Backup anstehend' befinden, schlägt die Ausführung der Anweisung SET INTEGRITY der Prozedur ADMIN_COPY_SCHEMA fehl.

Zur Behebung dieses Problems müssen Sie einen Befehl BACKUP DATABASE absetzen, um den Status 'Backup anstehend' für die betroffenen Tabellenbereiche aufzuheben. Als Nächstes müssen Sie die Spalte **Statement_text** der Fehlertabelle prüfen, die von dieser Prozedur generiert wurde, um eine Liste der Tabellen im Status 'Festlegen der Integrität anstehend' zu finden. Führen Sie anschließend die Anweisung SET INTEGRITY für jede in der Liste aufgeführte Tabelle aus, um jede einzelne Tabelle aus dem Status 'Festlegen der Integrität anstehend' herauszunehmen.

Dienstprogramm 'db2move'

Dieses Dienstprogramm versucht, alle zulässigen Schemaobjekte mit Ausnahme der folgenden Typen zu kopieren:

- Tabellenhierarchie
- Zwischenspeichertabellen (vom Dienstprogramm LOAD in Umgebungen mit mehreren Datenbankpartitionen nicht unterstützt)
- JAR-Dateien (Java Routine Archives)
- Kurznamen
- Pakete

- Sichthierarchien
- Objektzugriffsrechte (Alle neuen Objekte werden mit Standardberechtigungen erstellt.)
- Statistiken (Neue Objekte enthalten keine statistischen Informationen.)
- Indexerweiterungen (in Verbindung mit benutzerdefinierten strukturierten Typen)
- Benutzerdefinierte strukturierte Typen und ihre Umsetzungsfunktionen

Fehler aufgrund nicht unterstützter Typen
Wenn ein Objekt eines der nicht unterstützten Typen im Quellenschema angetroffen wird, wird ein Eintrag in einer Fehlerdatei protokolliert, der darauf hinweist, dass ein nicht unterstützter Objekttyp gefunden wurde. Die COPY-Operation wird trotzdem erfolgreich ausgeführt. Der protokollierte Eintrag soll Sie über Objekte informieren, die durch diese Operation nicht kopiert wurden.

Nicht mit Schemata gekoppelte Objekte
Objekte, die nicht mit einem Schema gekoppelt sind, wie zum Beispiel Tabellenbereiche und Ereignismonitore, werden bei einer Schemakopieroperation nicht berücksichtigt. Sie müssen sie in der Zieldatenbank vor dem Aufrufen der Schemakopieroperation erstellen.

Replizierte Tabellen
Wenn eine replizierte Tabelle kopiert wird, ist die neue Kopie der Tabelle nicht für die Replikation eingerichtet. Die Tabelle wird als reguläre Tabelle erneut erstellt.

Unterschiedliche Instanzen
Die Quellendatenbank muss katalogisiert werden, wenn sie sich nicht in derselben Instanz wie die Zieldatenbank befindet.

Option SCHEMA_MAP
Wenn die Option SCHEMA_MAP zur Angabe eines anderen Schemanamens in der Zieldatenbank verwendet wird, führt die Schemakopieroperation nur eine minimale Analyse der Objektdefinitionsanweisungen durch, um den ursprünglichen Schemanamen durch den neuen Schemanamen zu ersetzen. Zum Beispiel werden alle Vorkommen des ursprünglichen Schemanamens, die im Inhalt einer SQL-Prozedur auftreten, nicht durch den neuen Schemanamen ersetzt. Aus diesem Grund kann die Erstellung dieser Objekte durch die Schemakopieroperation fehlschlagen. Sie können die Anweisungen der Datendefinitionssprache (DDL) in der Fehlerdatei verwenden, um diese nicht erstellten Objekte nach Abschluss der Kopieroperation erneut zu erstellen.

Gegenseitige Abhängigkeiten zwischen Objekten
Die Schemakopieroperation versucht, Objekte in einer Reihenfolge erneut zu erstellen, die die gegenseitigen Abhängigkeiten zwischen diesen Objekten berücksichtigt. Wenn zum Beispiel eine Tabelle T1 eine Spalte enthält, die eine benutzerdefinierte Funktion U1 referenziert, erstellt die Operation die Funktion U1 erneut, bevor sie die Tabelle T1 erneut erstellt. Informationen zu Abhängigkeiten von Prozeduren sind jedoch in den Katalogen nicht einfach verfügbar. Bei der erneuten Erstellung von Prozeduren versucht die Schemakopieroperation daher zunächst, alle Prozeduren erneut zu stellen. Anschließend wiederholt sie den Versuch für die Prozeduren, deren erste Neuerstellung fehlgeschlagen ist. (Dies geschieht in der Annahme, dass sich Prozeduren, die von einer Prozedur abhängig sind, die beim vorherigen Versuch erfolgreich erstellt wurde, bei einem nachfolgenden Versuch erfolgreich neu erstellen lassen.) Die Operation setzt die Versuche, diese

Prozeduren, deren Neuerstellung fehlgeschlagen ist, erneut zu erstellen, so lange fort, wie sie in der Lage ist, mindestens eine weitere Prozedur bei einem nachfolgenden Versuch erneut zu erstellen. Bei jedem Versuch, eine Prozedur erneut zu erstellen, wird ein Fehler (mit zugehörigen DDL-Anweisungen) in der Fehlerdatei protokolliert. Möglicherweise sehen Sie in der Fehlerdatei viele Einträge für dieselben Prozeduren. Diese Prozeduren können dennoch bei einem nachfolgenden Versuch erfolgreich erneut erstellt worden sein. Sie sollten nach Abschluss der Schemakopieroperation die Tabelle SYSCAT.PROCEDURES abfragen, um festzustellen, ob diese in der Fehlerdatei aufgeführten Prozeduren erfolgreich erneut erstellt wurden.

Weitere Informationen finden Sie in den Beschreibungen zur Prozedur ADMIN_COPY_SCHEMA und zum Dienstprogramm db2move.

Beispiel für das Kopieren eines Schemas mit der Prozedur ADMIN_COPY_SCHEMA

Zum Kopieren eines einzelnen Schemas innerhalb derselben Datenbank können Sie die Prozedur ADMIN_COPY_SCHEMA wie nachfolgend gezeigt verwenden.

```
DB2 "SELECT SUBSTR(OBJECT_SCHEMA,1, 8)
  AS OBJECT_SCHEMA, SUBSTR(OBJECT_NAME,1, 15)
  AS OBJECT_NAME, SQLCODE, SQLSTATE, ERROR_TIMESTAMP, SUBSTR(DIAGTEXT,1, 80)
  AS DIAGTEXT, SUBSTR(STATEMENT,1, 80)
  AS STATEMENT FROM COPYERRSCH.COPYERRTAB"

CALL SYSPROC.ADMIN_COPY_SCHEMA('SOURCE_SCHEMA', 'TARGET_SCHEMA',
  'COPY', NULL, 'SOURCETS1 , SOURCETS2', 'TARGETTS1, TARGETTS2,
  SYS_ANY', 'ERRORSCHEMA', 'ERRORNAME')
```

Die Ausgabe aus dieser Anweisung SELECT sieht wie folgt aus:

```
OBJECT_SCHEMA OBJECT_NAME     SQLCODE     SQLSTATE ERROR_TIMESTAMP
------------- --------------- ----------- -------- --------------------------
SALES         EXPLAIN_STREAM         -290 55039    2006-03-18-03.22.34.810346

DIAGTEXT
--------------------------------------------------------------------------------
[IBM][CLI Driver][DB2/LINUXX8664] SQL0290N  Der Zugriff auf einen Tabellenbereich
ist nicht zulässig.


STATEMENT
--------------------------------------------------------------------------------
set integrity for "SALES  "."ADVISE_INDEX" , "SALES"."ADVISE_MQT" , "SALES"."
1 Satz/Sätze ausgewählt.
```

Beispiele für das Kopieren eines Schemas mit dem Dienstprogramm 'db2move'

Zum Kopieren eines oder mehrerer Schemata aus einer Quellendatenbank in eine Zieldatenbank können Sie das Dienstprogramm db2move mit der Aktion 'COPY -co' verwenden. Nach der Einrichtung eines Modellschemas können Sie dieses als Vorlage für die Erstellung neuer Versionen verwenden.

Beispiel 1: Verwenden der Optionen 'COPY -c'

Im folgenden Beispiel für das Dienstprogramm db2move mit der Option COPY -co wird das Schema BAR aus der Datenbank 'sample' in die Zieldatenbank ('target') kopiert und in FOO umbenannt:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" -u benutzer-id -p kennwort
```

Die neuen Zielschemaobjekte werden mit den gleichen Objektnamen wie die Objekte im Quellenschema, jedoch mit dem Qualifikationsmerkmal 'target' erstellt. Es ist möglich, Kopien von Tabellen mit oder ohne Daten aus der Quellentabelle zu erstellen. Die Quellendatenbank und die Zieldatenbank können sich auf verschiedenen Systemen befinden.

Beispiel 2: Angeben der Namenszuordnungen für Tabellenbereiche bei der COPY-Operation

Das folgende Beispiel zeigt, wie bestimmte Namenszuordnungen für Tabellenbereiche angegeben werden, die bei einer COPY-Operation mit dem Dienstprogramm db2move anstelle der Tabellenbereiche aus dem Quellensystem verwendet werden sollen. Sie können das Schlüsselwort SYS_ANY angeben, um festzulegen, dass der Zieltabellenbereich mithilfe des Standardauswahlalgorithmus für Tabellenbereiche ausgewählt werden soll. In diesem Fall wählt das Dienstprogramm db2move alle verfügbaren Tabellenbereiche zur Verwendung als Zielobjekte aus:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" -u benutzer-id -p kennwort
```

Das Schlüsselwort SYS_ANY kann für alle Tabellenbereiche verwendet werden. Alternativ können Sie auch bestimmte Zuordnungen für einige der Tabellenbereiche und den Standardauswahlalgorithmus für die restlichen Tabellenbereiche angeben:

```
db2move sample COPY -sn BAR -co target_db  target schema_map "
  ((BAR,FOO))" tablespace_map "((TS1, TS2),(TS3, TS4), SYS_ANY)"
  -u benutzer-id -p kennwort
```

In diesem Beispiel wird angegeben, dass Tabellenbereich TS1 dem Tabellenbereich TS2 und Tabellenbereich TS3 dem Tabellenbereich TS4 zugeordnet werden, die restlichen Tabellenbereiche jedoch den Standardauswahlalgorithmus für Tabellenbereiche verwenden.

Beispiel 3: Ändern der Objekteigner nach der COPY-Operation

Nach einer erfolgreichen COPY-Operation können Sie den Eigner der neuen Objekte ändern, die im Zielschema erstellt wurden. Der Standardeigner der Zielobjekte ist der Benutzer, der die Verbindung hergestellt hat. Wenn diese Option angegeben wird, wird das Eigentumsrecht an einen neuen Eigner wie folgt übertragen:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" owner jrichards
  -u benutzer-id -p kennwort
```

Der neue Eigner der Zielobjekte ist 'jrichards'.

Das Dienstprogramm db2move muss auf dem Zielsystem aufgerufen werden, wenn sich das Quellenschema und das Zielschema auf verschiedenen Systemen befinden. Um Schemata aus einer Datenbank in eine andere kopieren zu können, muss für diese Aktion eine Liste mit durch Kommata getrennten Schemanamen, die aus einer Quellendatenbank zu kopieren sind, sowie ein Zieldatenbankname eingegeben werden.

Zum Kopieren eines Schemas geben Sie db2move in eine Eingabeaufforderung des Betriebssystems wie folgt ein:

```
db2move <dbname> COPY -co <copy-optionen>
 -u <benutzer-id> -p <kennwort>
```

Erneutes Starten einer fehlgeschlagenen Operation zum Kopieren eines Schemas

Fehler, die bei einer Kopieroperation (COPY) mit dem Dienstprogramm db2move auftreten, können abhängig vom Typ des kopierten Objekts oder von der Phase der Kopieroperation, in der die Kopieroperation fehlgeschlagen ist (d. h., entweder die Phase, in der Objekte erneut erstellt werden, oder die Phase, in der die Daten geladen werden), auf unterschiedliche Weise behandelt werden.

Das Dienstprogramm db2move meldet Nachrichten und Fehler an den Benutzer in Nachrichten- und Fehlerdateien zurück. Schemakopieroperationen verwenden die Nachrichtendatei COPYSCHEMA_<zeitmarke>.MSG und die Fehlerdatei COPYSCHEMA_<zeitmarke>.err. Diese Dateien werden im aktuellen Arbeitsverzeichnis erstellt. Die aktuelle Uhrzeit wird an den Dateinamen angefügt, um die Eindeutigkeit der Dateien sicherzustellen. Dem Benutzer obliegt es, diese Nachrichten- und Fehlerdateien wieder zu löschen, wenn sie nicht mehr benötigt werden.

Anmerkung: Es ist möglich, mehrere Instanzen von db2move gleichzeitig auszuführen. Die Option COPY gibt keine SQLCODE-Werte zurück. Dies ist mit dem Verhalten des Dienstprogramms db2move konsistent.

Objekttypen

Der Typ des Objekts, das gerade kopiert wird, lässt sich in eine von zwei Kategorien einordnen: physisches Objekt oder Geschäftsobjekt.

Die Bezeichnung physisches Objekt bezieht sich auf Objekte, die sich physisch in einem Container befinden, wie zum Beispiel Tabellen, Indizes und benutzerdefinierte strukturierte Typen. Die Bezeichnung Geschäftsobjekt bezieht sich auf katalogisierte Objekte, die sich nicht in Containern befinden, wie zum Beispiel Sichten, benutzerdefinierte strukturierte Typen (UDTs) und Aliasnamen.

Fehler, die während der erneuten Erstellung eines physischen Objekts auftreten, bewirken, dass das Dienstprogramm ein Rollback durchführt, wohingegen Fehler, die während der erneuten Erstellung eines logischen Objekts auftreten, kein Rollback auslösen.

Erneutes Starten der Operation zum Kopieren eines Schemas

Nach der Behebung der Probleme, die das Fehlschlagen der Ladeoperationen (wie in der Fehlerdatei beschrieben) verursacht haben, können Sie den Befehl db2move -COPY mit der Option -tf zur Angabe der zu kopierenden Tabellen und zum Füllen der Tabellen mit Daten (unter Angabe des Dateinamens LOADTABLE.err) wie im folgenden Syntaxbeispiel gezeigt erneut ausführen:

```
db2move quellen_db COPY -tf LOADTABLE.err -co TARGET_DB ziel_db
  -mode load_only
```

Sie können die Tabellennamen auch manuell mit der Option -tn angeben, wie im folgenden Syntaxbeispiel gezeigt:

```
db2move quellen_db COPY -tn "FOO"."TABELLE1","FOO 1"."TAB 444",
  -co TARGET_DB ziel_db -mode load_only
```

Anmerkung: Der Modus load_only erfordert die Eingabe mindestens einer Tabelle mit der Option -tn oder -tf.

Beispiele

Fehler, die bei einer Schemaoperation COPY mit dem Dienstprogramm db2move auftreten, können abhängig vom Typ des kopierten Objekts oder von der Phase, in der die Operation COPY fehlgeschlagen ist, auf unterschiedliche Weise behandelt werden.

Das Dienstprogramm db2move meldet Nachrichten und Fehler bei Schemakopieroperationen in den folgenden Nachrichten- und Fehlerdateien zurück:

- Nachrichtendatei COPYSCHEMA <zeitmarke>.MSG
- Fehlerdatei COPYSCHEMA_<zeitmarke>.err

Diese Dateien werden im aktuellen Arbeitsverzeichnis erstellt. Die aktuelle Uhrzeit wird an den Dateinamen angefügt, um die Eindeutigkeit der Dateien sicherzustellen. Diese Nachrichten- und Fehlerdateien sollten gelöscht werden, wenn sie nicht mehr benötigt werden.

Anmerkung: Es ist möglich, mehrere Instanzen von db2move gleichzeitig auszuführen. Die Option COPY gibt keine SQLCODE-Werte zurück. Dies ist mit dem Verhalten des Dienstprogramms db2move konsistent.

Beispiel 1: Fehler beim Kopieren von Schemata im Zusammenhang mit physischen Objekten

Fehler, die bei der der Neuerstellung physischer Objekte in der Zieldatenbank auftreten, werden in der Fehlerdatei COPYSCHEMA_<zeitmarke>.err protokolliert. Für jedes fehlerhafte Objekt enthält die Fehlerdatei Informationen wie den Objektnamen, den Objekttyp, den DDL-Text, die Zeitmarke und einen als Zeichenfolge formatierten SQL-Kommunikationsbereich (SQLCA-Feldnamen gefolgt von den zugehörigen Datenwerten).

Beispielausgabe für die Fehlerdatei COPYSCHEMA_<zeitmarke>.err:

```
1. schema: FOO.T1
Type:      TABLE
Error Msg: SQL0104N Auf ... folgte das unerwartete Token 'FOO.T1'.
Timestamp: 2005-05-18-14.08.35.65
DDL:       create view FOO.v1

2. schema:  FOO.T3
Type:      TABLE
Error Msg: SQL0204N FOO.V1 ist ein nicht definierter Name.
Timestamp: 2005-05-18-14.08.35.68
DDL:       create table FOO.T3
```

Wenn Fehler bei der Erstellung physischer Objekte am Ende der Phase der Neuerstellung und vor der Ladephase protokolliert werden, schlägt die Ausführung des Dienstprogramms db2move unter Rückgabe eines Fehlers fehl. Alle bereits ausgeführten Schritte der Objekterstellung in der Zieldatenbank werden rückgängig gemacht (Rollback), während in der Quellendatenbank alle intern erstellten Tabellen bereinigt werden. Die Rollback-Operation erfolgt am Ende der Phase der Neuerstellung, nachdem versucht wurde, alle Objekte zu erstellen, und nicht bereits nach dem ersten Fehler. Auf diese Weise können alle möglichen Fehler in der Fehlerdatei erfasst werden. Dies bietet Ihnen die Möglichkeit, alle Probleme zu beheben, bevor Sie die db2move-Operation erneut starten. Wenn keine Fehler festgestellt werden, wird die Fehlerdatei gelöscht.

Beispiel 2: Fehler beim Kopieren von Schemata im Zusammenhang mit Geschäftsobjekten

Fehler, die bei der Neuerstellung von Geschäftsobjekten in der Zieldatenbank auftreten, führen nicht dazu, dass die Ausführung des Dienstprogramms db2move fehlschlägt. Solche Fehler werden stattdessen in der Fehlerdatei COPYSCHEMA_<zeitmarke>.err protokolliert. Nach der Beendigung des Dienstprogramms db2move können Sie die Fehler untersuchen, entsprechende Maßnahmen ergreifen und jedes Objekt, dessen Erstellung fehlgeschlagen ist, manuell erneut erstellen. (Hilfreiche DDL-Anweisungen zu diesem Zweck werden in der Fehlerdatei bereitgestellt).

Ein Fehler, der auftritt, während das Dienstprogramm db2move versucht, Tabellendaten mithilfe des Dienstprogramms LOAD in Tabellen zu laden, führt nicht dazu, dass die Ausführung des Dienstprogramms db2move fehlschlägt. Vielmehr werden generische Fehlerinformationen in der Datei COPYSCHEMA_<zeitmarke>.err (Objektname, Objekttyp, DDL-Text, Zeitmarke, SQL-Kommunikationsbereich usw.) protokolliert. Der vollständig qualifizierte Name der Tabelle wird in einer anderen Datei mit dem Namen LOADTABLE_<zeitmarke>.err protokolliert. In dieser Datei wird eine Tabelle pro Zeile aufgeführt, um die Formatanforderung der Option db2move -tf zu erfüllen. Das Format sieht in etwa wie folgt aus:

```
"FOO"."TABLE1"
"FOO 1"."TAB 444"
```

Beispiel 3: Andere Typen von db2move-Fehlern

Fehler bei internen Operationen, wie zum Beispiel Speicherfehler oder Dateisystemfehler, können dazu führen, dass die Ausführung des Dienstprogramms db2move fehlschlägt.

Wenn ein interner Fehler während der Operation in der Phase der Neuerstellung durch DDL-Anweisungen auftritt, werden alle erfolgreich erstellten Objekte aus dem Zielschema rückgängig gemacht und alle intern erstellten Tabellen, zum Beispiel die DMT-Tabelle und die db2look-Tabelle, werden in der Quellendatenbank bereinigt.

Wenn während der Ladephase der Operation ein interner Fehler auftritt, verbleiben alle erfolgreich erstellten Objekte im Zielschema. Alle Tabellen, bei denen während der Ladeoperation ein Fehler auftritt, und alle Tabellen, in die noch keine Daten geladen wurden, werden in der Fehlerdatei LOADTABLE.err protokolliert. Sie können anschließend den Befehl db2move COPY mit der Datei LOADTABLE.err wie in Beispiel 2 gezeigt ausführen. Wenn das Dienstprogramm db2move abnormal beendet wird (z. B. durch einen Systemabsturz, einen Trap oder einen manuellen Abbruch des Dienstprogramms usw.), gehen die Informationen zu den noch zu ladenden Tabellen verloren. In diesem Fall können Sie das Zielschema mithilfe der Prozedur ADMIN_DROP_SCHEMA löschen und den Befehl db2move COPY erneut absetzen.

Unabhängig von dem Fehler, der bei dem Versuch auftritt, eine Operation zum Kopieren eines Schemas auszuführen, haben Sie immer die Möglichkeit, das Zielschema mit der Prozedur ADMIN_DROP_SCHEMA zu löschen und den Befehl db2move COPY erneut abzusetzen.

Löschen von Schemata

Bevor Sie ein Schema löschen, müssen alle Objekte, die sich in diesem Schema befinden, gelöscht oder in ein anderes Schema versetzt werden. Der Schemaname muss im Katalog vorhanden sein, wenn die Anweisung DROP ausgeführt wird. Ansonsten wird ein Fehler zurückgegeben.

Geben Sie in der Befehlszeile Folgendes ein, um ein Schema zu löschen:

```
DROP SCHEMA <name> RESTRICT
```

Im folgenden Beispiel wird das Schema „joeschma" gelöscht:

```
DROP SCHEMA joeschma RESTRICT
```

Das Schlüsselwort `RESTRICT` erzwingt die Regel, dass im angegebenen Schema keine Objekte für das Schema definiert werden können, das aus der Datenbank gelöscht werden soll. Es muss außerdem angegeben werden.

Teil 3. Datenbankobjekte

Die Gestaltung logischer Datenbanken besteht aus der Definition von Datenbankobjekten.

In einer DB2-Datenbank können die folgenden Datenbankobjekte erstellt werden:

- Tabellen
- Integritätsbedingungen
- Indizes
- Trigger
- Sequenzen
- Sichten

Diese Datenbankobjekte können mithilfe von grafischen Benutzerschnittstellen oder durch das explizite Ausführen von Anweisungen erstellt werden. Die zum Erstellen dieser Datenbankobjekte verwendeten Anweisungen werden DLL-Anweisungen (DLL = Data Definition Language, Datendefinitionssprache) genannt und verfügen in der Regel über die Schlüsselwörter CREATE oder ALTER als Präfix.

Für die Implementierung einer guten Datenbankgestaltung, die den derzeitigen Bedürfnissen der Datenspeicherung Ihres Geschäfts entspricht und gleichzeitig flexibel genug ist, um mit der Zeit Erweiterung und Wachstum Rechnung zu tragen, ist es wichtig, sich über die Funktionen und die Funktionalität im Klaren zu sein, die jedes einzelne Datenbankobjekt bereitstellt.

Kapitel 11. Tabellen

Bei Tabellen handelt es sich um logische Strukturen, die vom Datenbankmanager verwaltet werden. Tabellen bestehen aus Spalten und Zeilen.

Am Schnittpunkt jeder Spalte und jeder Zeile gibt es ein bestimmtes Datenelement, das *Wert* genannt wird. Eine *Spalte* ist eine Gruppe von Werten desselben Typs oder eines der Untertypen dieses Typs. Eine *Zeile* ist eine Folge von Werten, die so angeordnet sind, dass der *n*te Wert ein Wert der *n*ten Spalte der Tabelle ist.

Ein Anwendungsprogramm kann die Reihenfolge bestimmen, in der die Zeilen in die Tabelle eingefügt werden. Die tatsächliche Reihenfolge der Zeilen wird jedoch durch den Datenbankmanager bestimmt und kann in der Regel nicht gesteuert werden. Das mehrdimensionale Clustering (MDC) stellt eine gewisse Clustering-methode bereit, bestimmt jedoch nicht die tatsächliche Anordnung unter den Zeilen.

Typen von Tabellen

In Abhängigkeit von Ihrer Umgebung müssen Sie zum Speichern Ihrer Daten eine oder mehrere Tabellen in Ihren DB2-Datenbanken erstellen. Bei der Erstellung von Tabellen geben Sie den Typ des Inhalts in den einzelnen Spalten der Tabellen an, und Sie definieren weitere Merkmale wie z. B. die Integritätsbedingungen über Primärschlüssel und Prüfungen auf Integritätsbedingungen für die Umsetzung von Geschäftsregeln. Wenn Sie Tabellen erstellen, müssen Sie auch den Typ berücksichtigen, der am besten für Ihre Bedürfnisse geeignet ist.

Außer (deklarierten) globalen temporären Tabellen können Sie alle nachfolgend aufgeführten Typen von Tabellen mit der Anweisung CREATE TABLE erstellen:

Basistabellen
Diese Tabellentypen enthalten persistente Daten.

Reguläre Tabellen
Diese Tabellentypen werden als Zwischenspeicher implementiert. Reguläre Tabellen mit Indizes sind die ″allgemein einsetzbaren″ Tabellen.

Tabellen im Anfügemodus
Diese Tabellentypen sind reguläre Tabellen, die in erster Linie für INSERT-Operationen optimiert werden. Reguläre Tabellen werden durch eine Anweisung ALTER TABLE in den Anfügemodus versetzt. Der Anfügemodus eignet sich am besten für Tabellen, bei denen das Clustering nach einem bestimmten Index nicht von Bedeutung ist, die Einfügerate hoch ist und keine oder keine häufigen Löschoperationen an den Tabellen stattfinden.

Ergebnistabellen
Diese Tabellentypen bestehen aus Zeilengruppen, die der Datenbankmanager aus mindestens einer Tabelle auswählt oder auf der Basis mindestens einer Tabelle generiert, um eine Abfrage zu erfüllen.

Übersichtstabellen
Diese Tabellentypen werden durch eine Abfrage definiert, mit der auch die Daten für die Tabelle ermittelt werden. Durch Übersichtstabellen kann die Leistungsfähigkeit von Abfragen erhöht werden. Wenn der Datenbankmanager erkennt, dass ein Teil einer Abfrage durch eine Übersichtstabelle

aufgelöst werden kann, kann der Datenbankmanager die Abfrage so abwandeln, dass die entsprechende Übersichtstabelle verwendet wird. Diese Entscheidung basiert auf Datenbankkonfigurationseinstellungen wie z. B. den Sonderregistern CURRENT REFRESH AGE und CURRENT QUERY OPTIMIZATION.

Typisierte Tabellen
Eine Tabelle kann den Datentyp aller Spalten separat definieren oder die Typzuordnung auf der Basis der Attribute vornehmen, die für einen benutzerdefinierten strukturierten Typen gelten. Derartige Tabellen werden als *typisierte Tabellen* bezeichnet. Ein benutzerdefinierter strukturierter Typ kann Teil einer Typhierarchie sein. Ein *untergeordneter Typ* übernimmt Attribute des zugehörigen *übergeordneten Typen*. In ähnlicher Weise kann eine typisierte Tabelle Teil einer Tabellenhierarchie sein. Eine *untergeordnete Tabelle* übernimmt Spalten der zugehörigen *übergeordneten Tabelle*. Beachten Sie, dass der Term *untergeordneter Typ* für einen benutzerdefinierten strukturierten Typen und alle benutzerdefinierten strukturierten Typen gilt, die sich innerhalb der Typhierarchie unter diesem befinden. Ein *echter untergeordneter Typ* eines strukturierten Typs T ist ein strukturierter Typ in der Typhierarchie unter T. In ähnlicher Weise gilt der Term *untergeordnete Tabelle* für eine typisierte Tabelle und alle typisierten Tabellen, die sich innerhalb der Tabellenhierarchie unter dieser befinden. Eine *echte untergeordnete Tabelle* einer Tabelle T ist eine Tabelle innerhalb der Tabellenhierarchie unter T.

(Deklarierte) globale temporäre Tabellen
Diese Tabellentypen, die auch als benutzerdefinierte temporäre Tabellen bezeichnet werden, werden von Anwendungen verwendet, die mit Daten in der Datenbank arbeiten. Ergebnisse aus der Bearbeitung der Daten müssen temporär in einer Tabelle gespeichert werden. Ein Tabellenbereich für temporäre Benutzertabellen muss vorhanden sein, bevor globale temporäre Tabellen erstellt werden.

Anmerkung: Die Beschreibung globaler temporärer Tabellen wird im Systemkatalog nicht aufgeführt, sodass sie von anderen Anwendungen nicht identifiziert und gemeinsam verwendet werden kann. Wenn die Anwendung, die diese Tabelle verwendet, beendet wird oder die Verbindung zur Datenbank trennt, werden alle Daten in dieser Tabelle gelöscht und auch die Tabelle wird implizit gelöscht.

Folgende Spalten werden von globalen temporären Tabellen nicht unterstützt:

- LOB-Spalten (oder Spalten eines einzigartiger Datentyps, die ebenfalls auf LOB basieren)
- Spalten mit benutzerdefinierten Datentypen
- LONG VARCHAR-Spalten
- XML-Spalten

Diese Tabellentypen werden mit der Anweisung DECLARE GLOBAL TEMPORARY TABLE erstellt und zum Speichern temporärer Daten für eine einzelne Anwendung verwendet. Diese Tabelle wird implizit gelöscht, wenn die Anwendung die Verbindung zur Datenbank trennt.

MDC-Tabellen (MDC = mehrdimensionales Clustering)
Diese Tabellentypen werden als Tabellen implementiert, die physisch auf der Grundlage mehrerer Schlüssel bzw. Dimensionen gleichzeitig in Clustern angeordnet werden. MDC-Tabellen werden beim Data-Warehousing

und in umfangreichen Datenbankumgebungen verwendet. Clusterindizes in regulären Tabellen unterstützen das eindimensionale Clustering von Daten. MDC-Tabellen bieten die Vorteile des Datenclusterings in mehr als einer Dimension.

Anmerkung: Eine MDC-Tabelle ist ein Typ von Tabelle, der jedoch zusammen mit einer partitionierten Tabelle vorhanden sein kann oder selbst eine partitionierte Tabelle sein kein. Der MDC-Tabellentyp schließt sich daher nicht mit anderen Typen von Tabellen gegenseitig aus. Eine partitionierte Tabelle kann auch eine Tabelle im Anfügemodus sein. (Einige andere Kombinationen sind jedoch nicht zulässig, wie zum Beispiel der MDC-Typ mit dem Anfügemodus, der RCT-Typ (Bereichsclustertabelle) mit einer anderen Tabelle oder andere Kombinationen). Außerdem sorgt der MDC-Typ für ein *garantiertes Clustering* in der zusammengesetzten Dimension, während bei regulären Tabellen mit Clusterindex das Clustering vom Datenbankmanager angestrebt wird, jedoch nicht garantiert werden kann und in der Regel im Verlauf der Zeit abnimmt.

Bereichsclustertabelle (RCT-Tabelle)
Diese Tabellentypen werden als sequenzielle Datencluster implementiert und bieten schnellen, direkten Zugriff. Jeder Datensatz in der Tabelle verfügt über eine vorbestimmte Satz-ID, die eine interne Kennung zur Lokalisierung eines Datensatzes in einer Tabelle darstellt. RCT-Tabellen werden verwendet, wenn die Daten eine dichte Clusterbildung über eine oder mehrere Tabellenspalten aufweisen. Die höchsten und niedrigsten Werte in den Spalten definieren den Bereich der möglichen Werte. Sie verwenden diese Spalten für den Zugriff auf Datensätze in der Tabelle. Dies ist die optimale Methode zur Nutzung des Merkmals der vorbestimmten Satz-IDs (RIDs) von RCT-Tabellen.

Partitionierte Tabellen
Diese Tabellentypen werden als Tabellen implementiert, die Werten in den Tabellenpartitionierungsschlüsselspalten für die Tabelle entsprechend ihre Daten auf mehrere Datenpartitionen verteilt haben. Partitionierte Tabellen ermöglichen eine einfachere Ein- und Auslagerung (Rollin und Rollout) von Tabellendaten, eine einfachere Verwaltung, eine flexible Positionierung von Indizes sowie eine bessere Abfrageverarbeitung als reguläre Tabellen.

Bei jeder Tabelle, die Ihre Daten enthält, sollten Sie prüfen, welcher Tabellentyp für Ihre Anforderungen am besten geeignet ist. Wenn Sie z. B. mit Datensätzen arbeiten, die über eine lockere Clusterbildung verfügen (ohne monotones Wachstum), sollten Sie prüfen, ob reguläre Tabellen mit Indizes eingesetzt werden können. Wenn Sie über Datensätze verfügen, die doppelte (nicht eindeutige) Werte im Schlüssel aufweisen, sollten Sie keine Bereichsclustertabellen verwenden. Wenn Sie vorab keine festgelegte Menge an Plattenspeicherplatz für Bereichsclustertabellen zuordnen können, sollten Sie diesen Tabellentyp auch nicht verwenden. Diese Faktoren erleichtern Ihnen die Feststellung, ob Sie über Daten verfügen, die sich für eine Bereichsclustertabelle eignen.

Entwerfen von Tabellen

Beim Entwerfen von Tabellen müssen Sie mit bestimmten Konzepten vertraut sein, den Platzbedarf für Tabellen- und Benutzerdaten bestimmen sowie festlegen, ob bestimmte Funktionen, wie zum Beispiel die Speicherbereichskomprimierung und das optimistische Sperren, genutzt werden sollen.

Beim Entwerfen von partitionierten Tabellen müssen Sie sich mit den Partitionierungskonzepten vertraut machen, wie zum Beispiel:

- Datenorganisationsschemata
- Tabellenpartitionierungsschlüssel
- Für die Verteilung von Daten auf Datenpartitionen verwendete Schlüssel
- Für MDC-Dimensionen verwendete Schlüssel

Informationen zu diesen und anderen Partitionierungskonzepten finden Sie in „Tabellenpartitionierung und Datenorganisationsschemata" auf Seite 287.

Entwurfskonzepte für Tabellen

Beim Entwerfen von Tabellen müssen Sie sich mit einigen zugehörigen Konzepten vertraut machen.

Angeben von Spaltendatentypen

Bei der Definition von Spalten müssen Sie die Spalten benennen, die Typen der Daten definieren, die in den Spalten gespeichert werden sollen (*Datentypen*), und die Längen der Daten für jede Spalte in der Tabelle definieren, die Sie erstellen wollen.

Als Binärdaten gespeicherte Zeichendaten

SMALL INTEGER (ganze Zahl ohne erweiterte Genauigkeit)
Dieser Datentyp dient zum Speichern binärer Ganzzahlwerte, die eine Genauigkeit von 15 Bit haben. Der Bereich für einen Wert des Typs SMALL INTEGER reicht von `-32.768` bis `+32.767`. Der Datentyp SMALL INTEGER verwendet den kleinstmöglichen Speicherplatz zum Speichern numerischer Werte (2 Byte Speicher sind für jeden gespeicherten Wert erforderlich). In einer Tabellendefinition wird eine Spalte des Typs SMALL INTEGER durch die Angabe SMALLINT deklariert.

INTEGER (ganze Zahl)
Dieser Datentyp dient zum Speichern binärer Ganzzahlwerte, die eine Genauigkeit von 31 Byte haben. Der Datentyp INTEGER benötigt zwar das Doppelte an Speicherplatz wie der Datentyp SMALL INTEGER (4 Byte Speicher sind für jeden gespeicherten Wert erforderlich), jedoch ist der Wertebereich wesentlich größer. Der Bereich für einen INTEGER-Wert reicht von `-2.147.483.648` bis `+2.147.483.647`. In einer Tabellendefinitionen können die Angaben INTEGER und INT zur Deklaration einer Spalte des Typs INTEGER verwendet werden.

BIG INTEGER (große ganze Zahl)
Dieser Datentyp dient zum Speichern binärer Ganzzahlwerte, die eine Genauigkeit von 63 Bit auf Plattformen besitzen, die 64-Bit-INTEGER-Werte unterstützen. Die Verarbeitung großer Zahlen, die als BIG INTEGER-Werte gespeichert sind, ist effizienter als die Verarbeitung ähnlicher Zahlen, die als DECIMAL-Werte gespeichert wurden. Darüber hinaus sind Berechnungen, die mit BIG INTEGER-Werten ausgeführt werden, präziser als Berechnungen, die mir REAL- oder DOUBLE-Werten ausgeführt werden.

Dieser Datentyp erfordert das Vierfache des Speicherplatzes des Datentyps SMALL INTEGER (8 Byte Speicher sind für jeden gespeicherten Wert erforderlich). Der Wertebereich des Datentyps BIG INTEGER reicht von `-9.223.372.036.854.775.808` bis

+9.223.372.036.854.775.807. In einer Tabellendefinition wird eine Spalte des Typs BIG INTEGER durch die Angabe BIGINT deklariert.

DECIMAL (Dezimalwert)
Dieser Datentyp dient zum Speichern numerischer Werte, die sowohl ganzzahlige Teile als auch Bruchteile enthalten. Die Teile werden kombiniert und in einem gepackten Dezimalformat gespeichert. Die Genauigkeit (die Gesamtanzahl der Stellen) und die Anzahl der Kommastellen (die Anzahl der Stellen, die für den Bruchteil des Wertes zu verwenden sind) müssen bei jeder Deklaration eines Datentyps DECIMAL angegeben werden. Der Bereich für die Genauigkeit eines DECIMAL-Werts liegt zwischen 1 und 31. Der Speicherbedarf zum Speichern eines DECIMAL-Werts lässt sich anhand der folgenden Gleichung berechnen: *Genauigkeit* dividiert durch 2 (abgeschnitten) + 1 = *Benötigte Byte zum Speichern*.

Ein Wert mit dem definierten Datentyp DECIMAL(8,2) benötigt zum Beispiel 5 Byte Speicher (8 dividiert durch 2 = 4 und 4 + 1 = 5). Ein Wert mit dem definierten Datentyp DECIMAL(7,2) benötigt dagegen 4 Byte Speicher (7 dividiert durch 2 = 3,5 (abgeschnitten auf 3) und 3 + 1 = 4).

In Tabellendefinitionen kann eine Spalte des Datentyps DECIMAL mit den Angaben DECIMAL, DEC, NUMERIC und NUM deklariert werden.

Anmerkung: Wenn die Werte für Genauigkeit und Anzahl der Kommastellen in einer DECIMAL-Spaltendefinition nicht angegeben werden, werden standardmäßig der Wert 5 für die Genauigkeit und der Wert 0 für die Anzahl der Kommastellen verwendet. (Daher werden 3 Byte Speicher benötigt.)

Gleitkommadatentyp mit einfacher Genauigkeit
Dieser Datentyp dient zum Speichern einer 32-Bit-Approximation einer reellen Zahl. Der Gleitkommadatentyp mit einer Genauigkeit (single-precision floating-point) und der Datentyp INTEGER erfordern den gleichen Speicherplatz (4 Byte Speicher sind für jeden gespeicherten Wert erforderlich). Der Wertebereich des Gleitkommadatentyps mit einfacher Genauigkeit ist jedoch wesentlich größer: $10E^{-38}$ bis $10E^{+38}$.

In einer Tabellendefinitionen können die Angaben REAL und FLOAT zur Deklaration einer Spalte des Gleitkommadatentyps mit einfacher Genauigkeit verwendet werden. Wenn jedoch FLOAT verwendet wird, muss die für die Spalte angegebene Länge zwischen 1 und 24 liegen, da FLOAT zur Darstellung beider Gleitkommadatentypen (mit einfacher und doppelter Genauigkeit) verwendet werden kann. Die angegebene Länge bestimmt, welcher Datentyp tatsächlich zu verwenden ist.

Gleitkommadatentyp mit doppelter Genauigkeit
Der Gleitkommadatentyp mit doppelter Genauigkeit (double-precision floating-point) dient zum Speichern einer 64-Bit-Approximation einer reellen Zahl. Der Gleitkommadatentyp mit doppelter Genauigkeit erfordert den gleichen Speicherplatz wie der Datentyp BIG INTEGER (8 Byte Speicher sind für jeden gespeicherten Wert

erforderlich). Der Gleitkommadatentyp mit doppelter Genauigkeit besitzt den größtmöglichen Wertebereich: $-1{,}79760^{+308}$ bis $-2{,}225E^{-307}$, 0 und $2{,}225E^{-307}$ bis $-1{,}79769E^{+308}$.

Zeichenfolge mit fester Länge

Dieser Datentyp dient zum Speichern von Zeichendaten und Zeichenfolgedaten, die eine bestimmte Länge haben, die 254 Zeichen nicht überschreitet. In einer Tabellendefinitionen kann eine Spalte mit dem Zeichenfolgedatentyp mit fester Länge durch die Angabe CHARACTER und CHAR deklariert werden. Die Länge der zu speichernden Zeichenfolgedaten muss bei jeder Deklaration eine Zeichenfolgedatentyps mit fester Länge angegeben werden. Der Speicherplatzbedarf zum Speichern eines Zeichenfolgewerts mit fester Länge lässt sich anhand der folgenden Gleichung bestimmen: *Feste Länge* x 1 = *Benötige Byte zum Speichern*. Ein Wert des Datentyps CHAR(8) erfordert zum Beispiel einen Speicherplatz von 8 Byte.

Anmerkung: Bei Verwendung von Zeichendatentypen mit fester Länge kann Speicherplatz verschwendet werden, wenn die tatsächliche Länge der Daten erheblich kleiner als die bei der Definition der Spalte angegebene Länge ist. Dies ist zum Beispiel der Fall, wenn die Werte 'JA' und 'NEIN' in einer Spalte gespeichert werden sollen, die mit dem Datentyp CHAR(20) wurde. Daher sollte die feste Länge, die für eine Spalte für Zeichenfolgen mit fester Länge angegeben wird, der tatsächlichen Länge der Daten, die in der Spalte gespeichert werden, möglichst nahe kommen.

Zeichendaten mit variabler Länge

Dieser Datentyp dient zum Speichern von Zeichenfolgedaten, die in der Länge variieren. Zeichenfolgedaten mit variabler Länge können bis zu 32.672 Zeichen lang sein. Die tatsächlich zulässige Länge unterliegt jedoch einer Einschränkung: Die Daten müssen auf eine Tabellenbereichsseite passen. Dies bedeutet, dass für eine Tabelle, die sich in einem Tabellenbereich mit 4-KB-Seiten befindet, Zeichenfolgedaten mit variabler Länge nicht länger als 4.092 Zeichen sein können. Für eine Tabelle, die sich in einem Tabellenbereich mit der Seitengröße 8 KB befinden, können die Zeichenfolgedaten mit variabler Länge nicht länger als 8.188 Zeichen sein (dies gilt analog für die weiteren Seitengrößen bis 32 KB). Da Tabellenbereiche standardmäßig mit einer Seitengröße von 4 KB erstellt werden, müssen Sie einen Tabellenbereich explizit mit einer größeren Seitengröße erstellen, wenn Sie Zeichenfolgen, die mehr als 4.092 Zeichen enthalten, mit dem Zeichenfolgedatentyp mit variabler Länge speichern wollen.

Anmerkung:

- Sie müssen darüber hinaus auch genügend Speicherplatz in der Tabellenzeile haben, um die Zeichenfolgedaten unterzubringen. Mit anderen Worten: Der Speicherbedarf für andere Spalten in der Tabelle muss zu dem Speicherbedarf der Zeichenfolgedaten hinzuaddiert werden, und der Gesamtspeicherbedarf darf die Seitengröße des Tabellenbereichs nicht überschreiten.
- Wenn ein Zeichenfolgedatenwert mit variabler Länge aktualisiert wird und der neue Wert größer ist als der ursprüngliche Wert, wird der Datensatz, der den Wert enthält, auf eine andere Seite in der Tabelle versetzt. Solche Datensätze werden als Zeigerdatensätze bezeichnet. Zu

viele Zeigerdatensätze können erhebliche Leistungseinbußen zur Folge haben, da mehrere Seiten abgerufen werden müssen, um einen einzelnen Datensatz zu verarbeiten.

In einer Tabellendefinitionen kann eine Spalte mit dem Zeichenfolgedatentyp mit variabler Länge durch die Angaben CHARACTER VARYING, CHAR VARYING und VARCHAR deklariert werden. Wenn eine Spalte mit dem Zeichenfolgedatentyp mit variabler Länge definiert wird, muss die erwartete maximale Anzahl Zeichen, die in dieser Spalte gespeichert werden, als Teil der Deklaration angegeben werden. Nachfolgende Zeichenfolgedatenwerte, die in der Spalte gespeichert werden, können kürzer oder gleich der angegebenen Maximallänge sein. Wenn sie länger sind, werden sie nicht gespeichert und ein Fehler wird zurückgegeben.

Der Speicherplatzbedarf zum Speichern eines Zeichenfolgewerts mit variabler Länge lässt sich anhand der folgenden Gleichung bestimmen: (*Zeichenfolgelänge* x 1) + 4 = *Benötigte Byte zum Speichern*. Das heißt, wenn eine Zeichenfolge aus 30 Zeichen mit dem Datentyp VARCHAR(30) gespeichert wird, benötigt dieser Wert 34 Byte Speicherplatz. (Alle Zeichenfolgen, die diesen Datentyp verwenden, dürfen maximal 30 Zeichen lang sein.)

Lange Zeichendaten mit variabler Länge

Der Datentyp für lange Zeichendaten mit variabler Länge dient ebenfalls zum Speichern von Zeichenfolgedaten, deren Länge variiert. Dieser Datentyp dient zum Speichern von Zeichenfolgedaten, deren Länge kleiner oder gleich `32.700` Zeichen ist, in einer Tabelle, die sich in einem Tabellenbereich mit einer Seitengröße von 4 KB befindet. Das heißt, bei Verwendung des langen Zeichenfolgedatentyps mit variabler Länge treffen die Einschränkungen in Bezug auf die Seitengröße/Zeichenfolgedatenlänge nicht zu, die für Zeichenfolgedaten mit variabler Länge gelten.

In einer Tabellendefinition wird eine Spalte mit dem Datentyp für lange Zeichenfolgen mit variabler Länge durch die Angabe LONG VARCHAR deklariert. Der Speicherplatzbedarf zum Speichern eines langen Zeichenfolgewerts mit variabler Länge lässt sich anhand der folgenden Gleichung bestimmen: (*Zeichenfolgelänge* x 1) + 24 = *Benötigte Byte zum Speichern*. Die Datentypen LONG VARCHAR und LONG VARGRAPHIC sind veraltet und werden in einem zukünftigen Release möglicherweise entfernt. Verwenden Sie als Spaltendatentyp Datentypen wie VARCHAR, VARGRAPHIC, CLOB oder DBCLOB, da diese in zukünftigen Releases weiterhin unterstützt und für portierbare Anwendungen empfohlen werden.

Anmerkung: Bei der Deklaration in einer Tabellendefinition kann die Klausel FOR BIT DATA mit einem beliebigen Zeichenfolgedatentyp verwendet werden. Wenn diese Klausel verwendet wird, werden bei Datenaustauschoperationen keine Codepagekonvertierungen ausgeführt und die Daten selbst werden als Binärdaten (Bitdaten) behandelt und verglichen.

Große Zeichenobjekte (CLOBs)

Ein Wert des Typs CLOB (Character Large Object, großes Zeichenobjekt) kann bis zu 2 Gigabyte (`2.147.483.647` Byte) lang sein. Ein CLOB-Datentyp dient zum Speichern großer zeichenbasierter Daten in einem Einzelbytezeichensatz (SBCS) oder in gemischten Einzel- und Mehrbytezeichensätzen (SBCS und MBCS), wie zum Beispiel Dokumente, die in einem Einzelbytezeichensatz geschrieben sind, und besitzt daher eine zugeordnete SBCS-Codepage bzw. eine gemischte Codepage.

Als Binärdaten gespeicherte Zeichendaten mit variabler Länge (LOBs und BLOBs)

Der Begriff *großes Objekt* und das generische Akronym LOB beziehen sich auf die Datentypen BLOB, CLOB oder DBCLOB. LOB-Werte unterliegen Einschränkungen, die auch für LONG VARCHAR-Werte gelten, wie im Abschnitt „Zeichendaten mit variabler Länge" beschrieben. Diese Einschränkungen gelten auch, wenn das Längenattribut der LOB-Zeichenfolge kleiner oder gleich 254 Byte ist. Dieser Datentyp dient zum Speichern binärer Zeichenfolgedaten, deren Länge variiert. Es wird häufig zum Speichern nicht konventioneller Daten wie Dokumente, Abbildungen und Bilder sowie Audio- und Videodaten verwendet.

Anmerkung: Große Binärobjekte (BLOBs) können mit SQL nicht in derselben Weise wie andere Daten bearbeitet werden. Zum Beispiel können BLOB-Werte nicht sortiert werden.

Unicode-Daten

Alle unterstützten Datentypen werden auch in einer Unicode-Datenbank unterstützt. Insbesondere werden Grafikzeichenfolgedaten für eine Unicode-Datenbank unterstützt und in UCS-2-Codierung gespeichert. Alle Clients, einschließlich Clients mit Einzelbytezeichensätzen (SBCS), können mit Grafikzeichenfolgedatentypen in UCS-2-Codierung arbeiten, wenn sie eine Verbindung zu einer Unicode-Datenbank herstellen.

Datums- und Zeitdaten (Zeitmarken)

Der Datumsdatentyp (*date*) dient zum Speichern eines dreiteiligen Werts (Jahr, Monat, Tag), der ein gültiges Kalenderdatum bezeichnet. Der Wertebereich für den Jahresteil umfasst die Werte 0001 bis 9999, der Wertebereich für den Monatsteil die Werte 1 bis 12 und der Wertebereich für den Tagesteil die Werte 1 bis n (28, 29, 30 oder 31), wobei n vom Monatsteil abhängig ist sowie davon, ob der Jahresteil einem Schaltjahr entspricht. Extern erscheint der Datumsdatentyp als Zeichenfolgedatentyp mit fester Länge, der 10 Zeichen lang ist. Intern erfordern die Datumsdaten jedoch wesentlich weniger Speicherplatz: Jeder gespeicherte Wert belegt 4 Byte Speicher, da Datumswerte als gepackte Zeichenfolgen gespeichert werden. In einer Tabellendefinition wird eine Datumsspalte durch die Angabe DATE deklariert.

Der Zeitdatentyp dient zum Speichern eines dreiteiligen Werts (Stunden, Minuten, Sekunden), der eine gültige Tageszeit im 24-Stunden-Format bezeichnet. Der Wertebereich für den Stundenteil umfasst die Werte 0 bis 24, der Wertebereich für den Minutenteil die Werte 0 bis 59 und der Wertebereich für den Sekundenteil ebenfalls die Werte 0 bis 59. (Wenn der Stundenteil auf 24 gesetzt wird, müssen der Minuten- und der Stundenteil auf 0 gesetzt werden.) Extern erscheint der Zeitdatentyp als Zeichenfolgedatentyp mit fester Länge, der 8 Zeichen lang ist. Ebenso wie Datumswerte werden Zeitwerte als gepackte Zeichenfolgen gespeichert, sodass in diesem Fall 3 Byte Speicherplatz für jeden gespeicherten Zeitwert erforderlich sind. In einer Tabellendefinition wird eine Zeitspalte durch die Angabe TIME deklariert.

Wie bei Datumswerten variiert die Darstellung der Uhrzeit in verschiedenen Teilen der Welt. Daher wird das Format eines Zeitwerts auch durch den Gebietscode bestimmt, der einer Datenbank zugeordnet ist. In Tabelle 49 auf Seite 261 sind die verfügbaren Zeitformate mit Beispielen für ihre Zeichenfolgedarstellung aufgeführt:

Tabelle 49. Datumsformate (JJJJ = Jahr, MM = Monat, TT = Tag)

Formatname	Abkürzung	Format der Datumszeichenfolge
International Standards Organization	ISO	JJJJ-MM-TT
IBM USA Standard	USA	MM/TT/JJJJ
IBM European Standard	EUR	MM/TT/JJJJ
Japanese Industrial Standard	JIS	JJJJ-MM-TT
Standortspezifisch	LOC	Basiert auf dem Gebietscode der Datenbank.

Numerische Daten

Alle numerischen Werte haben ein Vorzeichen und eine Genauigkeit. Das Vorzeichen wird als positiv betrachtet, wenn der numerische Wert null ist. Die Genauigkeit ist die Anzahl der Bit oder Stellen ohne Vorzeichen. Informationen dazu finden Sie im Abschnitt zu Datentypen in der Beschreibung der Anweisung CREATE TABLE.

Währungsdaten

Mit Version 9.5 wird der dezimale Gleitkommadatentyp DECFLOAT eingeführt, der in Geschäftsanwendungen (z. B. Finanzanwendungen) von Nutzen ist, die mit exakten Dezimalwerten arbeiten. Binäre Gleitkommadatentypen (REAL und DOUBLE), die binäre Approximationen für Dezimaldaten darstellen, sind in solchen Anwendungen nicht geeignet. Der Datentyp DECFLOAT kombiniert die Präzision des Datentyps DECIMAL mit den Leistungsvorteilen des Datentyps FLOAT, was für Anwendungen vorteilhaft ist, durch die Währungsdatenwerte verarbeitet werden.

XML-Daten

Der Datentyp XML dient zum Definieren von Spalten einer Tabelle, in denen XML-Werte gespeichert werden, wobei alle gespeicherten XML-Werte korrekt formatierte XML-Dokumente sein müssen. Die Einführung dieses nativen XML-Datentyps bietet die Möglichkeit, korrekt formatierte XML-Dokumente in ihrem nativen hierarchischen Format neben anderen relationalen Daten in der Datenbank zu speichern.

Generierte Spalten

Eine generierte Spalte wird in einer Tabelle definiert, bei der der gespeicherte Wert mithilfe eines Ausdrucks ermittelt wird, anstatt mit einer Einfüge- oder Aktualisierungsoperation angegeben zu werden.

Beim Erstellen einer Tabelle, bei der bestimmte Ausdrücke und Vergleichselemente ständig verwendet werden, können eine oder mehrere generierte Tabellen zu dieser Tabelle hinzugefügt werden. Durch die Verwendung einer generierten Spalte ergibt sich die Möglichkeit, bei der Durchführung von Abfragen in den Tabellendaten eine verbesserte Leistung zu erzielen.

Es gibt zum Beispiel zwei Fälle, in denen die Auswertung von Ausdrücken aufwendig sein kann, wenn die Leistung wichtig ist:

1. Die Auswertung des Ausdrucks muss während einer Abfrage häufig durchgeführt werden.
2. Die Berechnung ist komplex.

Um die Leistung einer Abfrage zu verbessern, können Sie eine zusätzliche Spalte definieren, die die Resultate des Ausdrucks enthält. Beim Absetzen einer Abfrage mit demselben Ausdruck kann die generierte Spalte dann direkt verwendet werden. Andernfalls kann die Komponente zum Umschreiben von Abfragen, die im Optimierungsprogramm implementiert ist, den Ausdruck gegen die generierte Spalte austauschen.

Wenn bei Abfragen Daten von zwei oder mehr Tabellen verknüpft werden, ermöglicht das Hinzufügen einer generierten Spalte dem Optimierungsprogramm die Auswahl potenziell besserer Joinstrategien.

Generierte Spalten können zur Verbesserung der Leistung von Abfragen eingesetzt werden. Aus diesem Grund werden generierte Spalten häufig nach dem Erstellen und Füllen einer Tabelle mit Werten hinzugefügt.

Beispiele

Das folgende Beispiel zeigt die Definition einer generierten Spalte in der Anweisung CREATE TABLE:

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

Nach dem Erstellen dieser Tabelle können mithilfe der generierten Spalten Indizes erstellt werden. Beispiel:

```
CREATE INDEX i1 ON t1(c4)
```

Abfragen können die Vorteile generierter Spalten nutzen. Beispiel:

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

Diese Abfrage könnte folgendermaßen geschrieben werden:

```
SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

Weiteres Beispiel:

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

Diese Abfrage könnte folgendermaßen geschrieben werden:

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

Automatische Nummerierung und ID-Spalten

Eine Identitätsspalte ermöglicht DB2 das automatische Generieren eines eindeutigen numerischen Wertes für jede Zeile, die der Tabelle hinzugefügt wird.

Beim Erstellen einer Tabelle, wobei Sie jede der Tabelle hinzugefügte Zeile eindeutig identifizieren müssen, können Sie der Tabelle eine Identitätsspalte hinzufügen. Um einen eindeutigen numerischen Wert für jede Zeile sicherzustellen, die der Tabelle hinzugefügt wird, sollten Sie einen eindeutigen Index für die Identitätsspalte definieren oder die Spalte als Primärschlüssel deklarieren.

Außerdem können Identitätsspalten für die Zuordnung von Bestellnummern, Personalnummern, Produktnummern oder Ereignisnummern verwendet werden. Die Werte für eine Identitätsspalte können mit dem DB2-Datenbankmanager unter Verwendung von ALWAYS oder BY DEFAULT generiert werden.

Eine mit GENERATED ALWAYS definierte Identitätsspalte erhält immer vom DB2-Datenbankmanager generierte Werte. Für Anwendungen ist die Angabe eines expliziten Wertes nicht zulässig. Eine mit GENERATED BY DEFAULT definierte Identitätsspalte ermöglicht Anwendungen die explizite Angabe eines Wertes für die Identitätsspalte. Wenn die Anwendung keinen Wert bereitstellt, wird ein entsprechender Wert von DB2 generiert. Da der Wert von der Anwendung gesteuert wird, kann seine Eindeutigkeit durch DB2 nicht garantiert werden. Die Klausel GENERATED BY DEFAULT sollte für die Datenweitergabe eingesetzt werden, wenn der Inhalt einer vorhandenen Tabelle kopiert werden soll. Andere Einsatzbereiche sind das Entladen und das erneute Laden einer Tabelle.

Nach der Erstellung können Sie die Tabellenbeschreibung nicht mehr ändern, um eine Identitätsspalte einzufügen.

Wenn Zeilen in eine Tabelle mit der Angabe expliziter Werte für Identitätsspalten eingefügt werden, wird der nächste intern generierte Wert nicht aktualisiert und kann Konflikte mit anderen in der Tabelle vorhandenen Werten verursachen. Doppelte Werte führen zu einer Fehlernachricht, wenn die Eindeutigkeit der Werte in der Identitätsspalte durch einen Primärschlüssel oder einen für die Identitätsspalte definierten eindeutigen Index überprüft wird.

Wenn Sie eine Identitätsspalte in einer neuen Tabelle definieren möchten, verwenden Sie in der Anweisung CREATE TABLE die Klausel AS IDENTITY.

Beispiel

Das folgende Beispiel zeigt die Definition einer Identitätsspalte in der Anweisung CREATE TABLE:

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

In diesem Beispiel wurde die dritte Spalte als Identitätsspalte definiert. Sie können darüber hinaus den in der Spalte verwendeten Wert zum eindeutigen Identifizieren aller hinzugefügten Zeilen verwenden. Im vorliegenden Beispiel wird für die erste eingegebene Zeile der Wert „100" in der Spalte platziert. Für jede nachfolgende, der Tabelle hinzugefügte Zeile wird der zugehörige Wert um 5 erhöht.

Einschränken von Spaltendaten durch Integritätsbedingungen, Standardwerte und Nulleinstellungen

Daten müssen häufig bestimmten Einschränkungen oder Regeln genügen. Solche Einschränkungen können für Einzelinformationen, wie zum Beispiel das Format und die Reihenfolge von Nummern, oder für verschiedene Informationen gelten.

Optionalität der Spaltendateneingabe

Nullwerte stellen unbekannte Status dar. Standardmäßig unterstützen alle integrierten Datentypen das Vorhandensein von Nullwerten. Es ist jedoch möglich, dass Geschäftsregeln vorgeben, dass für einige Spalten in jedem Fall ein Wert angegeben werden muss, wie zum Beispiel Notfallinformationen. Zur Definition einer solchen Bedingung können Sie die Integritätsbedingung NOT NULL angeben, die sicherstellt, dass eine bestimmte Spalte einer Tabelle nie einen Nullwert aufnimmt. Wenn die Integritätsbedingung NOT NULL für eine bestimmte Spalte definiert wurde, schlägt jede Einfüge- oder Aktualisierungsoperation fehl, die versucht, in dieser Spalte einen Nullwert zu speichern.

Standardspaltendatenwerte

Ebenso wie Geschäftsregeln vorschreiben können, dass immer ein Wert anzugeben ist, können andere Geschäftsregeln vorschreiben, was dieser Wert zu sein hat, zum Beispiel dass der Wert für das Geschlecht eines Mitarbeiters entweder M oder F sein muss. Die Standardwertintegritätsbedingung für eine Spalte dient zur Sicherstellung, dass eine bestimmte Spalte einer Tabelle immer einen vordefinierten Wert erhält, wenn eine Zeile, die keinen bestimmten Wert für diese Spalte enthält, in die Tabelle eingefügt wird. Der für eine Spalte vorgesehene Standardwert kann null sein, kann ein Integritätswert sein, der mit dem Datentyp der Spalte kompatibel ist, oder kann ein Wert sein, der vom Datenbankmanager bereitgestellt wird. Weitere Informationen finden Sie in „Standardspalten- und Datentypdefinitionen".

Schlüssel

Ein Schlüssel ist eine einzelne Spalte oder eine Gruppe von Spalten in einer Tabelle oder einem Index, die zur Ermittlung einer bestimmten Zeile bzw. für den Zugriff auf diese Zeile verwendet werden kann. Jede Spalte kann Teil eines Schlüssels sein, und dieselbe Spalte kann Teil mehrerer Schlüssel sein. Ein Schlüssel, der aus einer einzelnen Spalte besteht, wird als Atomarschlüssel (engl. atomic key) bezeichnet. Ein Schlüssel, der sich aus mehreren Spalten zusammensetzt, wird als zusammengesetzter Schlüssel bezeichnet. Neben ihrer Eigenschaft als atomare oder zusammengesetzte Schlüssel werden Schlüssel auch danach klassifiziert, wie sie zur Implementierung von Integritätsbedingungen eingesetzt werden:

- Ein eindeutiger Schlüssel wird zur Implementierung eindeutiger Integritätsbedingungen verwendet.
- Ein Primärschlüssel wird zur Implementierung von Entitätenintegritätsbedingungen verwendet. (Ein Primärschlüssel ist ein spezieller Typ von eindeutigem Schlüssel, der keine Nullwerte unterstützt.)
- Ein Fremdschlüssel wird zur Implementierung referenzieller Integritätsbedingungen verwendet. (Fremdschlüssel müssen auf Primärschlüssel oder eindeutige Schlüssel verweisen. Fremdschlüssel haben keine zugehörigen Indizes.)

Schlüssel werden in der Regel bei der Deklaration einer Tabelle, eines Index oder einer Definition einer referenziellen Integritätsbedingung angegeben.

Integritätsbedingungen

Integritätsbedingungen sind Regeln, die die Werte in einer Tabelle einschränken, die eingefügt, gelöscht oder aktualisiert werden können. Es gibt Prüfungen auf Integritätsbedingungen, Integritätsbedingungen über Primärschlüssel, referenzielle Integritätsbedingungen, eindeutige Integritätsbedingungen, Integritätsbedingungen über eindeutige Schlüssel, Integritätsbedingungen über Fremdschlüssel und informative Integrationsbedingungen. Detaillierte Informationen zu den einzelnen Typen von Integritätsbedingungen finden Sie in Kapitel 12, „Integritätsbedingungen", auf Seite 305 bzw. „Typen von Integritätsbedingungen" auf Seite 305.

Standardspalten- und Datentypdefinitionen:

Bestimmte Spalten- und Datentypen verfügen über vordefinierte oder zugeordnete Standardwerte.

Die folgenden Standardspaltenwerte für die verschiedenen Datentypen sind möglich:

- *NULL*
- *0:* Wird für ganze Zahlen ohne erweiterte Genauigkeit, ganze Zahlen, Dezimalzahlen, Gleitkommazahlen mit einfacher Genauigkeit und Gleitkommazahlen mit doppelter Genauigkeit verwendet.
- *Leerzeichen:* Wird für Zeichenfolgen mit fester Länge und für Doppelbytezeichenfolgen mit fester Länge verwendet.
- *Zeichenfolgewert der Länge null:* Wird für Zeichenfolgen mit variabler Länge, große Binärobjekte, große Zeichenobjekte und große Doppelbytezeichenobjekte verwendet.
- *Datum:* Dabei handelt es sich um das Systemdatum, an dem die Zeile eingefügt wurde (abgerufen aus dem Sonderregister CURRENT_DATE). Wenn einer vorhandenen Tabelle eine Datumsspalte hinzugefügt wird, wird vorhandenen Zeilen das Datum January, 01, 0001 zugeordnet.
- *Zeit oder Zeitmarke:* Dabei handelt es sich um die Systemzeit bzw. um das Systemdatum/die Systemzeit, an dem/zu der die Anweisung eingefügt wurde (abgerufen aus dem Sonderregister CURRENT_TIME). Wenn einer vorhandenen Tabelle eine Zeitspalte hinzugefügt wird, wird vorhandenen Zeilen die Uhrzeit 00:00:00 bzw. eine Zeitmarke zugeordnet, die das Datum January, 01, 0001 und die Uhrzeit 00:00:00 enthält.

  Anmerkung: Alle Zeilen erhalten denselben Standardwert für die Zeit/Zeitmarke für eine angegebene Anweisung.
- *Einzigartiger benutzerdefinierter Datentyp:* Dabei handelt es sich um den systemdefinierten Standardwert für den Basisdatentyp des einzigartigen benutzerdefinierten Datentyps (Umsetzung in den einzigartigen benutzerdefinierten Datentyp).

Integritätsbedingungen über Primärschlüssel, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und eindeutige Integritätsbedingungen

Integritätsbedingungen sind Regeln, die die Werte begrenzen, die in eine Tabelle eingefügt, aus einer Tabelle gelöscht oder in einer Tabelle aktualisiert werden können.

Integritätsbedingungen über Primärschlüssel

Eine Integritätsbedingung über Primärschlüssel ist eine Spalte oder eine Kombination von Spalten, die die gleichen Merkmale wie eine eindeutige Integritätsbedingung besitzt. Ein Primärschlüssel und Integritätsbedingungen über Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

Referenzielle Integritätsbedingungen (oder Integritätsbedingungen über Fremdschlüssel)

Eine Integritätsbedingung über Fremdschlüssel (auch als referenzielle Integritätsbedingung bezeichnet) ist eine logische Regel über Werte in einer oder mehreren Spalten in mindestens einer Tabelle. Zum Beispiel enthält eine Gruppe von Tabellen gemeinsame Informationen über die Lieferanten einer Firma. Gelegentlich ändert sich der Name eines Lieferanten. Sie können eine referenzielle Integritätsbedingung definieren, die besagt, dass die Kennung (ID) des Lieferanten in einer Tabelle mit einer Lieferanten-ID in den Lieferanteninformationen übereinstimmen muss. Diese Integritätsbedingung verhindert Einfüge-, Aktualisierungs- oder Löschoperationen, die ansonsten zu fehlenden Lieferanteninformationen führen würden.

Prüfungen auf Integritätsbedingungen
Eine Prüfung auf Integritätsbedingung (in Tabellen) definiert Einschränkungen für Daten, die einer bestimmten Tabelle hinzugefügt werden.

Eindeutige Integritätsbedingungen
Eine eindeutige Integritätsbedingung (auch als Integritätsbedingung über eindeutige Schlüssel bezeichnet) ist eine Regel, die untersagt, dass in einer oder mehreren Spalten innerhalb einer Tabelle doppelte Werte auftreten. Eindeutige Integritätsbedingungen werden in Form von eindeutigen Schlüsseln und Primärschlüsseln unterstützt.

Unicode-Tabellen und -Daten - Hinweise

Der Unicode-Zeichencodierungsstandard ist ein Codierungsschema für Zeichen mit fester Länge, das Zeichen fast aller lebenden Sprachen der Welt umfasst.

Weitere Informationen zu den Aspekten bezüglich Unicode-Tabellen und -Daten finden Sie in den folgenden Abschnitten:

- „Unicode-Zeichencodierung" in *Internationalisierung*
- „Zeichenvergleiche auf der Basis von Sortierfolgen" in *Internationalisierung*
- „Datums- und Zeitformate nach Gebietscodes" in *Internationalisierung*
- „Konvertierungstabellendateien für Euro-fähige Codepages" in *Internationalisierung*

Informationen zu Unicode stehen in der aktuellen Ausgabe von *The Unicode Standard* sowie auf der Website von The Unicode Consortium unter www.unicode.org zur Verfügung.

Speicherbedarf für Tabellen

Beim Entwerfen von Tabellen müssen Sie den Bedarf an Speicherplatz mit einkalkulieren.

Langfelddaten (LF-Daten)
Langfelddaten (LF) werden in einem separaten Tabellenobjekt gespeichert, das eine andere Struktur als der Speicherbereich für andere Datentypen aufweist.

Die Daten werden in Bereichen von 32 KB gespeichert, die sich aus Segmenten zusammensetzen, deren Größen sich aus dem Produkt der Zweierpotenzen und 512 Byte errechnen. (Diese Segmente können also aus 512 Byte, 1024 Byte, 2048 Byte usw. bis 32.768 Byte bestehen.)

Langfelddatentypen (LONG VARCHAR oder LONG VARGRAPHIC) werden auf eine Weise gespeichert, die eine problemlose Neuverwendung von freigegebenem Speicherbereich ermöglicht. Die Informationen zur Zuordnung und zu freien Speicherbereichen werden in Zuordnungsseiten von jeweils 4 KB gespeichert, die gelegentlich im Objekt erscheinen.

Der Bereich des freien Speicherplatzes in den Objekten ist von der Größe der Langfelddaten sowie davon abhängig, ob diese Größe bei jedem Vorkommen der Daten relativ konstant ist. Bei Dateneinträgen von mehr als 255 Byte kann dieser nicht genutzte Speicherplatz bis zu 50 Prozent der Größe der Langfelddaten ausmachen.

Wenn Zeichendaten kleiner als die Seitengröße sind und sie zusammen mit dem Rest der Daten in den Datensatz passen, sollten anstelle der Datentypen LONG VARCHAR oder LONG VARGRAPHIC die Datentypen CHAR, GRAPHIC, VARCHAR oder VARGRAPHIC verwendet werden.

LOB-Daten (LOB = Large Object, großes Objekt)

Daten großer Objekte (LOB-Daten) werden in zwei separaten Tabellenobjekten gespeichert, die eine andere Struktur als der Speicherbereich für andere Datentypen aufweisen. Bei der Abschätzung des für LOB-Daten erforderlichen Speicherbereichs müssen Sie die beiden Tabellenobjekte berücksichtigen, die zum Speichern von Daten dieser Datentypen verwendet werden:

- *LOB-Datenobjekte:* Die Daten werden in Bereichen von 64 MB gespeichert, die sich aus Segmenten zusammensetzen, deren Größen sich aus dem Produkt der Zweierpotenzen und 1024 Byte errechnen. (Diese Segmente können also aus 1024 Byte, 2048 Byte, 4096 Byte usw. bis 64 MB bestehen.)

  Zur Verringerung des für LOB-Daten erforderlichen Plattenspeicherplatzes können Sie den Parameter COMPACT in der Klausel für die LOB-Optionen der Anweisungen CREATE TABLE und ALTER TABLE angeben. Die Option COMPACT minimiert die Größe des für die LOB-Daten erforderlichen Speicherplatzes dadurch, dass die LOB-Daten in kleinere Segmente aufgeteilt werden können. Dieser Prozess beinhaltet keine Datenkomprimierung, sondern verwendet lediglich den kleinstmöglichen Speicherbereich bis zur nächsten 1-KB-Grenze. Die Angabe der Option COMPACT kann zu einer geringeren Leistung führen, wenn Daten an LOB-Werte angehängt werden.

  Der Umfang des freien Speicherbereichs innerhalb der LOB-Datenobjekte wird vom Umfang der Aktualisierungs- und Löschaktivitäten sowie von der Größe der eingefügten LOB-Werte beeinflusst.

- *LOB-Zuordnungsobjekte:* Die Informationen zur Zuordnung und zu freien Speicherbereichen werden in Zuordnungsseiten von jeweils 4 KB gespeichert, die von den eigentlichen Daten getrennt sind. Die Anzahl dieser 4-KB-Seiten ist vom Umfang der Daten (einschließlich des nicht genutzten Speicherbereichs) abhängig, die für die LOB-Daten zugeordnet wurden. Der Systemaufwand errechnet sich wie folgt: eine 4-KB-Seite pro 64 GB plus eine 4-KB-Seite pro 8 MB.

  Wenn Zeichendaten kleiner als die Seitengröße sind und sie zusammen mit dem Rest der Daten in den Datensatz passen, sollten anstelle der Datentypen BLOB, CLOB oder DBCLOB die Datentypen CHAR, GRAPHIC, VARCHAR oder VARGRAPHIC verwendet werden.

Systemkatalogtabellen

Systemkatalogtabellen werden erstellt, wenn eine Datenbank erstellt wird. Der Umfang dieser Systemtabellen nimmt in dem Maße zu, wie der Datenbank Datenbankobjekte und Zugriffsrechte hinzugefügt werden.

Zu Beginn belegen diese Tabellen einen Plattenspeicherplatz von ungefähr 3,5 MB.

Die Größe des Speicherbereichs, der für die Katalogtabellen zugeordnet wird, hängt von der Art des Tabellenbereichs und der Größe des durch EXTENTSIZE definierten Bereichs des Tabellenbereichs mit den Katalogtabellen ab. Wenn zum Beispiel ein DMS-Tabellenbereich mit einem EXTENTSIZE-Wert von 32 verwendet wird, wird dem Katalogtabellenbereich anfangs ein Speicherbereich von 20 MB zugeordnet. Hinweis: Für Datenbanken mit mehreren Partitionen befinden sich die Katalogtabellen nur in der Datenbankpartition, von der aus der Befehl CREATE DATABASE abgesetzt wurde. Plattenspeicherplatz für die Katalogtabellen ist nur für diese Datenbankpartition erforderlich.

Temporäre Tabellen

Einige Anweisungen benötigen für die Verarbeitung temporäre Tabellen (z. B. eine Arbeitsdatei für Sortieroperationen, die nicht im Speicher vorgenommen werden können). Diese temporären Tabellen benötigen Plattenspeicherplatz. Die Größe des erforderlichen Speichers hängt von der Größe, der Anzahl und der Art der Abfragen sowie von der Größe der Ergebnistabellen ab.

Da Ihre Arbeitsumgebung einzigartig ist, lässt sich der Platzbedarf für temporäre Tabellen nur sehr schwer schätzen. Zum Beispiel kann es aufgrund der längeren Lebensdauer verschiedener temporärer Systemdateien so scheinen, als ob mehr Speicherplatz für temporäre Systemtabellen zugeordnet wäre, als in Wirklichkeit genutzt wird. Dies könnte der Fall sein, wenn die Registrierdatenbankvariable DB2_SMS_TRUNC_TMPTABLE_THRESH verwendet wird.

Mit dem Datenbanksystemmonitor können und den APIs zum Abfragen des Tabellenbereichs können Sie die Größe des Arbeitsbereichs verfolgen, der während des normalen Betriebsablaufs genutzt wird.

Mit der Registrierdatenbankvariablen DB2_OPT_MAX_TEMP_SIZE können Sie die Größe des Tabellenbereichs für temporäre Tabellen begrenzen, die von Abfragen verwendet wird.

Tabellenseitengrößen

Zeilen von Tabellendaten werden in Blöcken verwaltet, die als Seiten bezeichnet werden. Seiten können vier verschiedene Größen haben: 4, 8, 16 und 32 Kilobyte. Tabellendatenseiten enthalten keine Daten für Spalten, die mit den Datentypen LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB oder DCLOB definiert sind. Die Zeilen auf einer Tabellendatenseite enthalten jedoch einen Deskriptor dieser Spalten.

Sie können Pufferpools und Tabellenbereiche erstellen, die Seitengrößen von 4, 8, 16 oder 32 KB besitzen. Alle Tabellen, die innerhalb eines Tabellenbereichs einer bestimmten Größe erstellt wurden, besitzen eine übereinstimmende Seitengröße. Die maximale Größe für ein einzelnes Tabellen- oder Indexobjekt beträgt 512 GB bei einer einer Seitengröße von 32 KB.

Sie können maximal 1012 Spalten haben, wenn Sie eine Seitengröße von 8, 16 oder 32 KB haben. Bei einer Seitengröße von 4 KB sind maximal 500 Spalten möglich. Größere Satzkennungen (RIDs), die mehr Datenseiten pro Tabellenobjekt und mehr Sätze pro Seite zulassen, bewirken eine Änderung des erforderlichen Haupt- und Plattenspeicherplatzes, der von Protokolldateien und Tabellenbereichen für temporäre Systemtabellen belegt wird. Die maximal mögliche Anzahl von Zeilen für eine Seitengröße von 32 KB beträgt ca. 2335, wenn die minimale Zeilengröße (12) verwendet wird.

Die maximalen Zeilenlängen sind je nach verwendeter Seitengröße unterschiedlich:

- Bei einer Seitengröße von 4 KB beträgt die maximale Zeilenlänge 4.005 Byte.
- Bei einer Seitengröße von 8 KB beträgt die maximale Zeilenlänge 8.101 Byte.
- Bei einer Seitengröße von 16 KB beträgt die maximale Zeilenlänge 16.293 Byte.
- Bei einer Seitengröße von 32 KB beträgt die maximale Zeilenlänge 32.677 Byte.

Zur Ermittlung der geeigneten Seitengröße für einen Tabellenbereich müssen Sie folgende Faktoren berücksichtigen:

- Für OLTP-Anwendungen, die wahlfreie Lese- und Schreiboperationen durchführen, ist meist eine geringere Seitengröße vorzuziehen, weil dadurch weniger Pufferspeicher durch unerwünschte Zeilen belegt wird.
- Für DSS-Anwendungen, die jeweils auf eine große Anzahl aufeinander folgender Zeilen gleichzeitig zugreifen, sind in der Regel größere Seiten vorteilhafter, weil dadurch weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen. Es gibt jedoch eine Ausnahme von dieser Regel. Wenn die von Ihnen verwendete Zeilengröße kleiner als `PAGESIZE / maximale Anzahl Zeilen` ist, bleibt auf jeder Seite Speicherbereich ungenutzt. In diesem Fall ist eine geringere Seitengröße geeigneter.

Durch größere Seiten können Sie möglicherweise die Zahl der Indexstufen reduzieren. Größere Seiten unterstützen längere Zeilen. Bei Verwendung von 4-KB-Seiten sind Tabellen auf 500 Spalten begrenzt. Größere Seitengrößen (8, 16 und 32 KB) unterstützen bis zu 1012 Spalten. Die Maximalgröße des Tabellenbereichs ist proportional zur Seitengröße des Tabellenbereichs.

Speicherbedarf für Benutzertabellendaten

Standardmäßig werden die Tabellendaten auf Seiten von jeweils 4 KB gespeichert. Jede Seite enthält (unabhängig von der Seitengröße) 68 Byte Systemaufwand für den Datenbankmanager. Dadurch bleiben 4028 Byte für die Benutzerdaten (oder Zeilen), obwohl keine Zeile auf einer 4-KB-Seite die Länge von 4005 Byte überschreiten kann. Eine Zeile kann sich *nicht* über mehrere Seiten erstrecken. Wenn Sie Seiten mit jeweils 4 KB verwenden, sind maximal 500 Spalten möglich.

Tabellendatenseiten enthalten *keine* Daten für Spalten mit den Datentypen LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB oder DBCLOB. Die Zeilen einer Tabellendatenseite enthalten jedoch einen Deskriptor für diese Spalten.

Zeilen werden normalerweise an der ersten passenden Stelle in einer regulären Tabelle eingefügt. Die Datei wird nach dem ersten verfügbaren Speicherbereich durchsucht (unter Verwendung einer Liste der freien Speicherbereiche), der groß genug ist, um die neue Zeile aufzunehmen. Das Aktualisieren einer vorhandenen Zeile erfolgt an der ursprünglichen Stelle, es sei denn, der freie Bereich der 4-KB-Seite reicht nicht aus, um die aktualisierte Zeile aufzunehmen. In diesem Fall wird an der Position, an der sich die Originalzeile befand, ein Datensatz erstellt, der auf die neue Speicherposition der aktualisierten Zeile in der Tabellendatei verweist.

Bei Verwendung der Anweisung ALTER TABLE APPEND ON werden Daten immer angehängt, und es werden keine Informationen zum freien Speicherbereich der Datenseiten aufgezeichnet.

Wenn die Tabelle über einen definierten Clusterindex verfügt, versucht der Datenbankmanager, die Daten entsprechend der Schlüsselreihenfolge dieses Clusterindex physisch (zu Clustern) zusammenzufassen. Wenn eine Zeile in die Tabelle eingefügt wird, sucht der Datenbankmanager zuerst den entsprechenden Schlüsselwert im Clusterindex. Wird der Schlüsselwert gefunden, versucht der Datenbankmanager, den Datensatz auf der durch den Schlüssel angegebenen Datenseite einzufügen. Wird der Schlüsselwert nicht gefunden, wird der nächst höhere Schlüsselwert verwendet, sodass der Datensatz auf der Seite eingefügt wird, die Datensätze mit dem nächst höheren Schlüsselwert enthält. Wenn auf der Zielseite nicht genügend freier Speicherbereich zur Verfügung steht, wird die Liste der freien Speicherbereiche zur Suche nach benachbarten Seiten mit freiem Speicherbereich verwendet. Mit der Zeit wird der Speicherbereich auf den Datenseiten vollständig belegt, und Datensätze werden immer weiter von der Zielseite

entfernt in der Tabelle gespeichert. Schließlich werden die Tabellendaten als Daten ohne Clustering betrachtet. In diesem Fall kann die Clusterreihenfolge durch eine Tabellenreorganisation wiederhergestellt werden.

Wenn es sich bei der Tabelle um eine MDC-Tabelle (mit mehrdimensionalem Clustering) handelt, garantiert der Datenbankmanager, dass Datensätze immer entlang mindestens einer definierten Dimension oder entsprechend den Clusterindizes physisch in Clustern angeordnet werden. Wenn eine MDC-Tabelle mit bestimmten Dimensionen definiert wird, wird ein Blockindex für jede der Dimensionen sowie ein zusammengesetzter Blockindex erstellt, der Zellen (eindeutige Kombinationen aus Dimensionswerten) Blöcken zuordnet. Dieser zusammengesetzte Blockindex dient zur Bestimmung, zu welcher Zelle ein bestimmter Datensatz gehört und welche Blöcke oder EXTENTSIZE großen Speicherbereiche in der Tabelle Datensätze enthalten, die zu dieser Zelle gehören. Beim Einfügen von Datensätzen durchsucht der Datenbankmanager daher den zusammengesetzten Blockindex nach der Liste von Blöcken, die Datensätze mit den gleichen Dimensionswerten enthalten, und beschränkt die Suche nach freiem Speicherplatz auf diese Blöcke. Wenn die Zelle noch nicht vorhanden ist oder wenn in den vorhandenen Blöcken der Zelle kein ausreichender Speicherbereich zur Verfügung steht, wird der Zelle ein weiterer Block zugeordnet und der Datensatz in diesen eingefügt. Eine Liste der freien Speicherbereiche wird weiterhin innerhalb von Blöcken verwendet, um rasch verfügbaren Speicherbereich in den Blöcken ausfindig zu machen.

Die Anzahl der 4-KB-Seiten für jede Benutzertabelle in der Datenbank kann nach folgender Berechnung geschätzt werden:

```
ABRUNDEN(4028/(durchschnittszeilengröße + 10)) = datensätze_pro_seite
```

Das Ergebnis wird in folgende Berechnung eingefügt:

```
(zahl_der_datensätze/datensätze_pro_seite) * 1,1 = anzahl_der_seiten
```

Dabei ist die durchschnittliche Zeilenlänge die Summe der durchschnittlichen Spaltengrößen, und der Faktor ″1,1″ dient zur Kalkulation des Systemaufwands.

Anmerkung: Diese Formel liefert nur einen Schätzwert. Die Genauigkeit des Schätzwerts nimmt ab, wenn die Datensatzlänge durch Fragmentierung und Überlaufsätze variiert.

Sie können auch Pufferpools oder Tabellenbereiche mit einer Seitengröße von 8, 16 oder 32 KB erstellen. Alle Tabellen, die innerhalb eines Tabellenbereichs einer bestimmten Größe erstellt werden, besitzen eine übereinstimmende Seitengröße. Die maximale Größe für ein einzelnes Tabellen- oder Indexobjekt beträgt 512 GB bei einer Seitengröße von 32 KB. Wenn Sie Seiten mit einer Größe von 8, 16 oder 32 KB verwenden, sind maximal 1012 Spalten möglich. Für eine 4-KB-Seite beträgt die maximale Spaltenanzahl 500. Die maximalen Zeilenlängen variieren ebenfalls je nach Seitengröße:

- Bei einer Seitengröße von 4 KB beträgt die maximale Zeilenlänge 4.005 Byte.
- Bei einer Seitengröße von 8 KB beträgt die maximale Zeilenlänge 8.101 Byte.
- Bei einer Seitengröße von 16 KB beträgt die maximale Zeilenlänge 16.293 Byte.
- Bei einer Seitengröße von 32 KB beträgt die maximale Zeilenlänge 32.677 Byte.

Größere Seiten ermöglichen eine geringere Anzahl von Indexstufen in einem Index. Wenn Sie mit OLTP-Anwendungen arbeiten, die wahlfreie Lese- und Schreibzugriffe auf Zeilen durchführen, ist eine geringere Seitengröße empfehlenswert, weil weniger Pufferspeicher durch unerwünschte Zeilen belegt wird. Wenn Sie mit DSS-Anwendungen (Decision Support System) arbeiten, die jeweils auf eine große

Anzahl aufeinander folgender Zeilen zugreifen, sind größere Seiten empfehlenswert, weil dann weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen.

Ein Backup-Image kann nicht in einer anderen Seitengröße wiederherstellt werden.

Sie können keine IXF-Datendateien importieren, die mehr als 755 Spalten beinhalten.

Deklarierte temporäre Tabellen können nur in einem eigenen Tabellenbereichstyp für temporäre Benutzertabellen erstellt werden. Es gibt keinen Standardtabellenbereich für temporäre Benutzertabellen. Temporäre Tabellen können keine Daten mit LONG-Typen enthalten. Die Tabellen werden implizit gelöscht, wenn eine Anwendung die Verbindung zur Datenbank trennt. Schätzungen über den Platzbedarf solcher Tabellen sollten dies berücksichtigen.

Speicherplatzkomprimierung für Tabellen

Tabellen können Speicherplatz auf Platten durch Funktionen wie die Komprimierung von Datenzeilen, NULL-Werten und Standardwerten möglicherweise sparsamer nutzen. Durch die Datenkomprimierung lässt sich vielleicht Plattenspeicher einsparen, weil weniger Datenbankseiten zum Speichern von Daten verwendet werden. Da pro Seite mehr logische Daten gespeichert werden können, brauchen beim Zugriff auf die gleiche Menge logischer Daten weniger Seiten gelesen zu werden. Dies bedeutet, dass die Komprimierung zudem Einsparungen beim Platten-E/A-Aufwand ermöglicht. Die E/A-Geschwindigkeit kann sich ebenfalls erhöhen, da mehr logische Daten im Pufferpool zwischengespeichert werden können.

Zur Implementierung der Datenkomprimierung in einem Datenbanksystem stehen zwei Methoden zur Verfügung:

(Platzsparende) Wertkomprimierung

Diese Methode optimiert die Speicherbelegung für die Darstellung von Daten sowie die Speicherstrukturen, die vom Datenbankmanagementsystem (DBMS) intern zum Speichern von Daten verwendet werden. Bei der Wertkomprimierung werden doppelte Einträge für einen Wert entfernt, sodass nur eine Kopie gespeichert wird. Die gespeicherte Kopie zeichnet die Positionen aller Verweise auf den gespeicherten Wert auf.

Bei der Erstellung einer Tabelle können Sie mit der optionalen Klausel VALUE COMPRESSION angeben, dass die Tabelle das platzsparende Zeilenformat auf Tabellenebene und möglicherweise auf Spaltenebene verwendet.

Wenn die Klausel VALUE COMPRESSION verwendet wird, werden Nullwerte (NULL) und Daten der Länge 0, die Datentypen mit variabler Länge (VARCHAR, VARGRAPHICS, LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB und DBCLOB) zugeordnet wurden, nicht auf der Platte gespeichert. Lediglich Werte für den mit diesen Datentypen verbundenen Systemaufwand belegen Plattenspeicherplatz.

Bei Verwendung der Klausel VALUE COMPRESSION kann auch die Option COMPRESS SYSTEM DEFAULT verwendet werden, um die Plattenspeichernutzung weiter zu verringern. Eingefügte oder aktualisierte Werte, die mit dem Systemstandardwert für den Datentyp ihrer Spalte übereinstimmen, beanspruchen nur minimalen Plattenspeicherplatz. Der Standardwert selbst wird nicht auf der Platte gespeichert. Alle numerischen Datentypen, Zeichenfolgetypen mit fester Länge und Grafikzeichenfolgen

mit fester Länge für Spalten unterstützen die Option COMPRESS SYSTEM DEFAULT. Das heißt, dass Nullen und Leerzeichen komprimiert werden können.

(Datenzeilenkomprimierung) Zeilenkomprimierung
Bei dieser Methode werden Datenzeilen komprimiert, indem sich wiederholende Muster, die sich über mehrere Spaltenwerte innerhalb einer Zeile erstrecken, durch kürzere Symbolzeichenfolgen ersetzt werden. Die Datenzeilenkomprimierung zielt darauf ab, Einsparungen beim benötigten Plattenspeicherplatz zu realisieren. Sie kann außerdem zu Aufwandseinsparungen bei E/A-Operationen für Platten führen. Darüber hinaus können mehr Daten im Pufferpool zwischengespeichert werden, wodurch sich verbesserte Trefferquoten für den Pufferpool ergeben. Allerdings kommt es bei dieser Vorgehensweise zu einem erhöhten Aufwand in Form von zusätzlichen CPU-Zyklen, die für die Komprimierung und Dekomprimierung der Daten benötigt werden. Die Einsparungen an Speicher und die Auswirkungen auf die Leistung, die sich durch die Datenzeilenkomprimierung ergeben, sind eng mit den Merkmalen der Daten in der Datenbank, mit dem Layout und der Optimierung der Datenbank sowie mit der Anwendungsauslastung verbunden. Nur die Daten auf einer Datenseite oder in Protokollsätzen werden komprimiert.

Die Datenzeilenkomprimierung arbeitet mit einem statischen, auf einem Wörterverzeichnis basierenden Komprimierungsalgorithmus, um Daten zeilenweise zu komprimieren. Durch die Komprimierung von Daten auf Zeilenebene können sich wiederholende Muster, die mehrere Spaltenwerte innerhalb einer Zeile umfassen, durch kürzere Symbolzeichenfolgen ersetzt werden. Zum Komprimieren von Tabellendaten muss das Attribut COMPRESS der Tabelle auf YES gesetzt werden. Außerdem muss für die Tabelle ein Komprimierungswörterverzeichnis (Compression Dictionary) vorhanden sein.

Zur Erstellung eines Komprimierungswörterverzeichnisses (und zur nachfolgenden Komprimierung einer Tabelle) führen Sie eine klassische Tabellenreorganisation (d. h. eine Offlinereorganisation) aus. Ein Komprimierungswörterverzeichnis wird auch bei den folgenden Operationen erstellt: INSERT, einschließlich IMPORT, LOAD INSERT und LOAD REPLACE, sowie bei einigen REDISTRIBUTE-Operationen. Alle Datenzeilen, die in einer Tabelle enthalten sind, werden bei der Erstellung des Komprimierungswörterverzeichnisses berücksichtigt. Das Wörterverzeichnis wird zusammen mit den Tabellendatenzeilen in den Datenobjektabschnitten der Tabelle gespeichert.

Zur Dekomprimierung einer Tabelle setzen Sie das Attribut COMPRESS auf den Wert NO und führen anschließend eine klassische Tabellenreorganisation (d. h. eine Offlinereorganisation) aus.

Eine Datenzeile, die in eine Seite eingefügt wird, kann komprimiert werden, wenn das Attribut COMPRESS für die Tabelle den Wert YES hat und ein Wörterverzeichnis vorhanden ist. Dies gilt für jede Einfügeoperation von Zeilen, einschließlich der Einfügung durch Import- oder Ladeoperationen (Dienstprogramm LOAD). Die Komprimierung wird jeweils für die gesamte Tabelle aktiviert. Die Zeilen werden jedoch einzeln komprimiert. Daher kann eine Tabelle sowohl komprimierte als auch nicht komprimierte Zeilen zu gleicher Zeit enthalten.

Die Datenzeilenkomprimierung kann auf Indexobjekte sowie LONG-, LOB- und XML-Datenobjekte nicht angewendet werden.

Die Zeilenkomprimierung ist nicht mit der Replikationsunterstützung für Tabellendaten kompatibel.

Die Zeilenkomprimierungsstatistik kann mithilfe des Befehls RUNSTATS generiert und in der Systemkatalogtabelle SYSCAT.TABLES gespeichert werden. Eine Option zur Komprimierungsschätzung steht mit dem Dienstprogramm INSPECT zur Verfügung. Das Abfrageoptimierungsprogramm berücksichtigt den Dekomprimierungsaufwand in seinem Modell zur Aufwandsberechnung.

Abhängig von den UPDATE-Aktivitäten und der Positionierung der Aktualisierungsänderungen innerhalb einer Datenzeile kann es zu einer erhöhten Protokollbelegung kommen. Informationen zur Protokollierung von Aktualisierungen und zur Anordnung von Spalten finden Sie in „Ordnen von Spalten zur Minimierung der Aktualisierungsprotokollierung" auf Seite 274.

Bei Zeilen, die durch eine Aktualisierung größer werden, passt die neue Version möglicherweise nicht mehr auf die aktuelle Datenseite. Stattdessen wird das neue Image der Zeile auf einer Überlaufseite gespeichert. Um die Erstellung solcher Überlaufzeigerdatensätze zu minimieren, kann mehr freier Speicherbereich innerhalb einer Datenseite hinzugefügt werden. Wenn ohne Komprimierung zum Beispiel 5 % freier Speicherbereich verwendet wurde, können Sie mit Komprimierung 10 % freien Speicherbereich zuordnen. Diese Empfehlung ist insbesondere für Daten wichtig, die in großem Umfang aktualisiert werden.

Speicherplatzkomprimierung für vorhandene Tabellen

Durch Angabe der Klausel VALUE COMPRESSION kann das Zeilenformat einer vorhandenen Tabelle geändert werden, um eine Speicherplatzkomprimierung zu ermöglichen. Beachten Sie, dass die Summe der Byteanzahlen der Spalten, die eine Speicherplatzkomprimierung ermöglichen, die Summe der Byteanzahlen der Spalten, die keine Speicherplatzkomprimierung zulassen, überschreiten kann. Dies ist akzeptabel, sofern die Summe der Bytezahlen nicht die zulässige Zeilenlänge der Tabelle im Tabellenbereich überschreitet. In einem Tabellenbereich mit einer Seitengröße von 4 KB beträgt die zulässige Zeilenlänge zum Beispiel 4005 Byte. Falls die zulässige Zeilenlänge überschritten wird, wird Fehlernachricht SQL0670N zurückgegeben. Die Formel für die Byteanzahl ist im Rahmen der Informationen über die Anweisung CREATE TABLE dokumentiert.

In ähnlicher Weise können durch Entfernen der Klausel VALUE COMPRESSION Tabellenzeilen, die vorher eine Speicherplatzkomprimierung ermöglicht haben, geändert werden, sodass sie keine Speicherplatzkomprimierung mehr zulassen. Hinsichtlich der Summe der Byteanzahlen der Spalten gilt dieselbe Bedingung, ebenso wie auch die Fehlernachricht SQL0670N zurückgegeben wird, falls erforderlich.

Bei der Ermittlung, ob eine Speicherplatzkomprimierung für eine Tabelle in Betracht kommt, sollte berücksichtigt werden, dass eine Tabelle, deren Werte mehrheitlich mit den Systemstandardwerten übereinstimmen oder NULL sind, Vorteile durch das neue Zeilenformat gewinnen würde. Wenn zum Beispiel eine Tabelle eine INTEGER-Spalte enthält und 90 % der Spalte den Wert 0 (d. h. den Standardwert für den Datentyp INTEGER) oder NULL haben, würde eine Komprimierung dieser Tabelle plus dieser Spalte zu einem Vorteil aufgrund des neuen Zeilenformats führen und eine erhebliche Einsparung an Plattenspeicherplatz bedeuten.

Bei der Änderung einer Tabelle können Sie mit der Klausel VALUE COMPRESSION angeben, dass die Tabelle das Komprimierungszeilenformat auf Tabellenebene und möglicherweise auf Spaltenebene verwendet. Mit ACTIVATE VALUE COMPRESSION geben Sie an, dass die Tabelle die Speichereinspartechniken nutzen soll, während Sie mit DEACTIVATE VALUE COMPRESSION angeben, dass die Tabelle nicht länger mit den Speicherspartechniken für Daten in der Tabelle arbeiten soll.

Wenn Sie DEACTIVATE VALUE COMPRESSION verwenden, inaktiviert dies implizit alle Optionen COMPRESS SYSTEM DEFAULT, die Spalten in dieser Tabelle zugeordnet sind.

Nach der Modifizierung der Tabelle in ein neues Zeilenformat erhalten alle nachfolgenden Zeilen, die eingefügt, geladen oder aktualisiert werden, das neue Zeilenformat. Damit jede Zeile in das neue Zeilenformat geändert wird, sollten Sie eine Reorganisation der Tabelle oder eine Aktualisierungsoperation an vorhandenen Zeilen ausführen, bevor Sie das Zeilenformat ändern.

Ordnen von Spalten zur Minimierung der Aktualisierungsprotokollierung

Wenn Sie Spalten mit der Anweisung CREATE TABLE definieren, müssen Sie die Reihenfolge der Spalten insbesondere bei aktualisierungsintensiven Auslastungen berücksichtigen. Spalten, die häufig aktualisiert werden, sollten zusammengruppiert und näher zum Ende bzw. am Ende der Tabellendefinition definiert werden. Dies verbessert die Leistung, verringert die Anzahl der zu protokollierenden Byte und senkt die Anzahl der zu schreibenden Protokollseiten. Darüber hinaus vermindert diese Vorgehensweise den Speicherplatzbedarf für aktive Protokolle bei Transaktionen, die eine große Anzahl von Aktualisierungen ausführen.

Der Datenbankmanager nimmt nicht automatisch an, dass Spalten, die in der SET-Klausel einer UPDATE-Anweisung angegeben sind, hinsichtlich des Werts geändert werden sollen. Zur Begrenzung des Aufwands der Indexpflege und des zu protokollierenden Umfangs der Zeile vergleicht die Datenbank den neuen Spaltenwert mit dem alten Spaltenwert, um festzustellen, ob die Spalte geändert wird. Nur Spalten, deren Wert sich ändert, werden als Spalten behandelt, die aktualisiert werden. Ausnahmen von diesem UPDATE-Verhalten ergeben sich für Spalten, bei denen die Daten außerhalb der Datenzeile gespeichert werden (Spaltentypen LONG, LOB, ADT und XML), oder für Spalten mit fester Länge, wenn die Registrierdatenbankvariable DB2ASSUMEUPDATE aktiviert ist. In diesen Ausnahmefällen wird angenommen, dass sich der Spaltenwert ändert, sodass kein Vergleich zwischen dem neuen und dem alten Spaltenwert ausgeführt wird.

Es gibt drei unterschiedliche Typen von UPDATE-Protokollsätzen.

- Vollständige Protokollierung der Vor- und Nachzeilenimages. Das gesamte Vorimage und Nachimage der Zeile wird protokolliert. Dies ist der einzige Typ von Protokollierung, der für Tabellen mit aktiviertem DATA CAPTURE CHANGES ausgeführt wird, und hat die höchste Anzahl von Byte, die für eine Aktualisierung an einer Zeile protokolliert wird.
- Vollständige XOR-Protokollierung. Die XOR-Unterschiede zwischen dem Vorimage und dem Nachimage der Zeile, angefangen vom ersten Byte, das sich ändert, bis zum Ende der kleineren Zeile sowie alle restlichen Byte in der längeren Zeile, werden protokolliert. Dies führt dazu, dass weniger Byte protokolliert werden als bei der Protokollierung der vollständigen Vor- und Nachimages, wobei die Anzahl von Datenbyte, die über die Kopfinformationen des Protokollsatzes hinausgehen, die Größe des größten Zeilenimages ist.

- Partielle XOR-Protokollierung. Die XOR-Unterschiede zwischen dem Vorimage und dem Nachimage der Zeile, angefangen vom ersten Byte, das sich ändert, bis zum letzten Byte, das sich ändert, werden protokolliert. Bytepositionen können die ersten oder die letzten Byte einer Spalte sein. Bei diesem Verfahren wird die geringste Anzahl von Byte protokolliert und der effizienteste Typ von Protokollsatz für eine Aktualisierung an einer Zeile generiert.

Wenn DATA CAPTURE CHANGES für die Tabelle nicht aktiviert ist, hängt der Umfang der Daten, die für eine Aktualisierung protokolliert werden, von folgenden Faktoren ab:

- Die räumliche Nähe der aktualisierten Spalten (COLNO)
- Ob die aktualisierten Spalten eine feste oder variable Länge haben
- Ob die Zeilenkomprimierung (COMPRESS YES) aktiviert ist

Wenn sich die Gesamtlänge der Zeile auch bei aktivierter Zeilenkomprimierung nicht ändert, berechnet und schreibt der Datenbankmanager den optimalen partiellen XOR-Protokollsatz.

Wenn sich die Gesamtlänge der Zeile ändert, was bei der Aktualisierung von Spalten mit variabler Länge und bei aktivierter Zeilenkomprimierung der Regelfall ist, stellt der Datenbankmanager fest, welches Byte zuerst zu ändern ist und schreibt einen vollständigen XOR-Protokollsatz.

Datenzeilenkomprimierung

Der Zweck der Datenzeilenkomprimierung besteht darin, Einsparungen beim benötigten Plattenspeicherplatz zu realisieren. Zudem kann sie auch zu Einsparungen beim Platten-E/A-Aufwand führen. Darüber hinaus können mehr Daten im Pufferpool zwischengespeichert werden, wodurch sich die Trefferquoten im Pufferpool erhöhen. Die Datenzeilenkomprimierung arbeitet mit einem statischen, auf einem Wörterverzeichnis basierenden Komprimierungsalgorithmus, um Daten zeilenweise zu komprimieren.

Durch die Komprimierung von Daten auf Zeilenebene können sich wiederholende Muster, die mehrere Spaltenwerte innerhalb einer Zeile umfassen, durch kürzere Symbolzeichenfolgen ersetzt werden.

Anmerkung: Dieses Verfahren ist mit einem höheren Aufwand an CPU-Zyklen verbunden, die für die Komprimierung und Dekomprimierung der Daten benötigt werden. Die Einsparungen an Speicher und die Auswirkungen auf die Leistung, die sich durch die Datenzeilenkomprimierung ergeben, sind eng mit den Merkmalen der Daten in der Datenbank, mit dem Layout und der Optimierung der Datenbank sowie mit der Anwendungsauslastung verbunden. Nur die Daten auf einer Datenseite oder in Protokollsätzen werden komprimiert.

Die Komprimierung von Tabellendaten erfolgt nur, wenn ein Komprimierungswörterverzeichnis (Compression Dictionary) für die Tabelle vorhanden ist, wenn Sie das Tabellenattribut COMPRESS in der Anweisung CREATE oder ALTER TABLE auf YES gesetzt haben und wenn genügend Daten in der Tabelle enthalten sind. Wenn diese Komprimierungsbedingungen für die Tabelle erfüllt sind, werden Daten, die Sie der Tabelle mit einer Anweisung INSERT bzw. einem Befehl LOAD INSERT, IMPORT INSERT oder REDISTRIBUTE hinzufügen, komprimiert.

In Version 9.5 wird die Datenzeilenkomprimierung automatisch aktiviert, wenn für eine Tabelle das Attribut COMPRESS auf YES gesetzt ist und das Datenkomprimierungswörterverzeichnis erstellt wurde. Wenn Sie eine Tabelle erstellt oder geändert

und dabei das Attribut COMPRESS auf YES gesetzt haben, ist keine manuelle Operation oder Datenbankanforderung Ihrerseits erforderlich. Das heißt, Sie brauchen keine explizite klassische (d. h. offline ausgeführte) Tabellenreorganisation auszuführen, um das Datenkomprimierungswörterverzeichnis zu erstellen.

Anmerkung: Wenn Sie das Attribut COMPRESS auf YES setzen und bereits ein Komprimierungswörterverzeichnis vorhanden ist, wird die Komprimierung auf alle Zeileneinfügeoperationen, einschließlich Einfügungen durch eine IMPORT- oder LOAD-Operation, angewendet. Die Komprimierung wird jeweils für die gesamte Tabelle aktiviert. Die Zeilen werden jedoch einzeln komprimiert. Daher kann eine Tabelle gleichzeitig sowohl komprimierte als auch nicht komprimierte Zeilen enthalten.

Zur expliziten Erstellung eines Komprimierungswörterverzeichnisses (und zur nachfolgenden Komprimierung einer Tabelle) führen Sie eine klassische Tabellenreorganisation (d. h. eine Offlinetabellenreorganisation) aus. Alle Datenzeilen, die in einer Tabelle enthalten sind, werden bei der Erstellung des Komprimierungswörterverzeichnisses berücksichtigt. Das Wörterverzeichnis wird mit den Tabellendatenzeilen in den Datenobjektabschnitten der Tabelle gespeichert.

Zur Dekomprimierung einer Tabelle setzen Sie das Attribut COMPRESS auf den Wert NO und führen anschließend eine klassische Tabellenreorganisation (d. h. eine Offlinetabellenreorganisation) aus.

Einschränkungen

- Die Datenzeilenkomprimierung kann auf Indexobjekte sowie LONG-, LOB- und XML-Datenobjekte nicht angewendet werden.
- Die Zeilenkomprimierung ist nicht mit der Replikationsunterstützung für Tabellendaten kompatibel.
- Sie können mithilfe des Befehls RUNSTATS Statistikdaten zur Zeilenkomprimierung generieren. Diese werden in der Systemkatalogtabelle SYSCAT.TABLES gespeichert. Eine Schätzoption für die Komprimierung, die die Effektivität der Zeilenkomprimierung für eine Tabelle schätzt, wird mit dem Dienstprogramm INSPECT zur Verfügung gestellt. Das Abfrageoptimierungsprogramm berücksichtigt den Dekomprimierungsaufwand in seinem Modell zur Aufwandsberechnung.
- Abhängig von den Aktualisierungsaktivitäten und der Positionierung der Aktualisierungsänderungen innerhalb einer Datenzeile kann es zu einer höheren Protokollspeicherbelegung kommen.
- Wenn eine Zeile wächst, ist es möglich, dass die neue Version der Zeile nicht auf die aktuelle Datenseite passt. In diesem Fall wird das neue Image der Zeile auf einer Überlaufseite gespeichert. Um die Erstellung solcher Überlaufzeigerdatensätze zu minimieren, können Sie mehr freien Speicherbereich innerhalb einer Datenseite hinzufügen. Wenn ohne Komprimierung zum Beispiel 5 % freier Speicherbereich verwendet wurde, ordnen Sie mit Komprimierung 10 % freien Speicherbereich zu. Diese Empfehlung ist insbesondere für Daten wichtig, die in großem Umfang aktualisiert werden.

Optimistisches Sperren

Mit Version 9.5 wird durch die erweiterte Unterstützung für optimistisches Sperren eine Technik für SQL-Datenbankanwendungen bereitgestellt, bei der Sperren zwischen dem Auswählen (SELECT) und dem Aktualisieren (UPDATE) bzw. Löschen (DELETE) von Zeilen nicht durchgehend beibehalten werden.

Anwendungen können so geschrieben werden, dass sie von der optimistischen Annahme ausgehen, dass es unwahrscheinlich ist, dass nicht gesperrte Zeilen vor dem Aktualisieren oder Löschen geändert werden. Wenn die Zeilen dennoch geändert werden, schlagen die Aktualisierungs- oder Löschoperationen fehl. In diesem Fall kann die Logik der Anwendung entsprechend reagieren, zum Beispiel indem sie eine erneute Auswahl (SELECT) versucht.

Der Vorteil dieses erweiterten optimistischen Sperrens liegt im besseren gemeinsamen Zugriff, da andere Anwendungen dieselben Zeilen lesen und schreiben können. Diese Technik des optimistischen Sperrens wird in dreischichtigen Umgebungen verwendet, in denen Geschäftsanwendungen keine Korrelation zu Datenbanktransaktionen besitzen, da Sperren nicht über Geschäftstransaktionen hinweg beibehalten werden können.

Optimistisches Sperren

Beim *optimistischen Sperren* handelt es sich um ein Verfahren für SQL-Datenbankanwendungen, bei dem keine Zeilensperren zwischen dem Auswählen und Aktualisieren oder Löschen einer Zeile gehalten werden.

Die Anwendung wurde so geschrieben, dass sie von der optimistischen Annahme ausgeht, dass nicht gesperrte Zeilen vor der Aktualisierungs- oder Löschoperation wahrscheinlich nicht geändert werden. Wenn doch eine Zeile geändert wird, schlägt der Aktualisierungs- oder Löschvorgang fehl und die Anwendungslogik behandelt solche Fehler beispielsweise dadurch, dass sie die Auswahl wiederholt. Ein Vorteil des optimistischen Sperrens ist der verbesserte gemeinsame Zugriff, da andere Anwendungen diese Zeile lesen und schreiben können. In einer dreischichtigen Umgebung, in der Geschäftstransaktionen keine Korrelation mit Datenbanktransaktionen besitzen, wird das Verfahren des optimistischen Sperrens verwendet, da Sperren nicht über Geschäftstransaktionen hinweg beibehalten werden können.

Allerdings hat das optimistische Sperren nach Werten auch einige Nachteile:

- Es kann zu *falschen positiven Werten* ohne weitere Datenserverunterstützung führen, das heißt zu einer Bedingung bei Verwendung des optimistischen Sperrens, bei der eine Zeile, die seit ihrer Auswahl *geändert* wurde, nicht aktualisiert werden kann, ohne zunächst erneut ausgewählt zu werden. (Dies unterscheidet sich von *falschen negativen Werten*, die eine Bedingung darstellen, bei der eine Zeile seit ihrer Auswahl *unverändert* geblieben ist, nicht aktualisiert werden kann, ohne zunächst erneut ausgewählt zu werden.)
- Es erfordert mehr Wiederholungslogik in Anwendungen.
- Für Anwendungen ist es schwierig, die UPDATE-Suchbedingungen zu bilden.
- Für den DB2-Server ist es ineffizient, nach der Zielzeile auf der Basis von Werten zu suchen.
- Datentypabweichungen zwischen einigen Clienttypen und Datenbanktypen, z. B. Zeitmarken, verhindern die Verwendung sämtlicher Spalten in der Aktualisierung mit Suche.

Version 9.5 fügt Unterstützung für ein schnelleres und einfacheres optimistisches Sperren ohne *falsche positive Werte* hinzu. Diese Unterstützung wird mithilfe der folgenden neuen SQL-Funktionen, Ausdrücke und Funktionen hinzugefügt:

- Integrierte Zeilenkennungsfunktion (RID_BIT oder RID)
- Ausdruck ROW CHANGE TOKEN
- Zeitbasierte Aktualisierungserkennung
- Implizit verdeckte Spalten

DB2-Anwendungen können das *optimistische Sperren nach Werten* ermöglichen, indem eine UPDATE-Anweisung mit Suche gebildet wird, durch die nach der Zeile mit genau denselben ausgewählten Werten gesucht wird. Die Anweisung UPDATE mit Suche schlägt fehl, wenn sich die Spaltenwerte der Zeile geändert haben.

Anwendungen, die dieses Programmiermodell verwenden, können von der verbesserten Funktion für optimistisches Sperren profitieren. Beachten Sie, dass Anwendungen, die dieses Programmiermodell nicht verwenden, nicht als Anwendungen für optimistisches Sperren betrachtet werden. Solche Anwendungen funktionieren weiterhin wie bisher.

Integrierte Zeilenkennungsfunktion (RID_BIT oder RID)
Diese integrierte Funktion kann in der SELECT-Liste oder in Vergleichselementen der Anweisung verwendet werden. In einem Vergleichselement (z. B. WHERE RID_BIT(tabelle)=?) wird das Gleichheitsvergleichselement mit RID_BIT als neue direkte Zugriffsmethode implementiert, um die Zeile effizient zu lokalisieren. Früher wurde eine VALUES-Methode zum optimistischen Sperren mit Werten angewendet, bei der alle ausgewählten Spaltenwerte den Vergleichselementen hinzugefügt wurden. Diese weniger effiziente Zugriffsmethode war davon abhängig, dass einige eindeutige Spaltenkombinationen jeweils nur eine Zeile kennzeichneten.

Ausdruck ROW CHANGE TOKEN
Dieser neue Ausdruck gibt ein Token vom Typ BIGINT zurück. Das Token stellt einen relativen Punkt in der Änderungsabfolge einer Zeile dar. Eine Anwendung kann den aktuellen Wert des Zeilenänderungstokens einer Zeile mit dem Wert des Zeilenänderungstokens vergleichen, der beim letzten Abruf der Zeile gespeichert wurde, um zu ermitteln, ob die Zeile inzwischen geändert wurde.

Zeitbasierte Aktualisierungserkennung:
Diese Funktion wird der SQL-Anweisung mit dem Ausdruck ROW CHANGE TIMESTAMP hinzugefügt. Zur Unterstützung dieser Funktion muss die Tabelle über eine neu generierte Spalte verfügen, die für Zeitmarken von Zeilenänderungen definiert ist, die zum Speichern der Zeitmarkenwerte definiert wurde. Diese kann vorhandenen Tabellen mit der Anweisung ALTER TABLE hinzugefügt werden oder die Zeitmarkenspalte für Zeilenänderung kann bei der Erstellung einer neuen Tabelle definiert werden. Das Vorhandensein der Zeitmarkenspalte für Zeilenänderung wirkt sich auch insofern auf das Verhalten des optimistischen Sperrens aus, als dass die Spalte zur Verbesserung der Granularität des Zeilenänderungstokens von der Seitenebene auf die Zeilenebene verwendet wird, was für Anwendungen mit optimistischem Sperren von großem Vorteil sein kann. Diese Funktion wurde auch DB2 für z/OS hinzugefügt.

Implizit verdeckte Spalten:
Aus Gründen der Kompatibilität vereinfacht diese Funktion die Verwendung der hinzugefügten Zeitmarkenspalten für Zeilenänderung in vorhandenen Tabellen und Anwendungen. Implizit verdeckte Spalten werden nicht extern zugänglich gemacht, wenn implizite Spaltenlisten verwendet werden. Beispiel:

- Eine Anweisung SELECT * für die Tabelle gibt keine implizit verdeckten Spalten in der Ergebnistabelle zurück
- Eine Anweisung INSERT ohne Spaltenliste erwartet keinen Wert für implizit verdeckte Spalten. Allerdings muss die Spalte so definiert sein, dass sie Nullwerte zulässt oder einen anderen Standardwert hat.

Anmerkung: Im DB2-Glossar finden Sie die Definition von Termini zum optimistischen Sperren wie z. B. *optimistische Steuerung des gemeinsamen Zugriffs*, *pessimistisches Sperren*, *ROWID* und *Aktualisierungserkennung*.

Einschränkungen und Aspekte des optimistischen Sperrens

In diesem Abschnitt werden die Einschränkungen des optimistischen Sperrens aufgeführt, die zu beachten sind.

- Spalten für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) werden in den folgenden Schlüsseln, Spalten und Namen nicht unterstützt (falls verwendet, wird SQLSTATE-Wert 429BV zurückgegeben):
  - Primärschlüssel
  - Fremdschlüssel
  - MDC-Spalten (MDC - Multidimensional Clustering)
  - Bereichspartitionierungsspalten
  - Hashpartitionierungsschlüssel für Datenbanken
  - Spalten mit der Integritätsbedingung DETERMINED BY
  - Kurznamen
- Die Funktion RID() wird in DPF-Konfigurationen (DPF - Database Partitioning Feature) nicht unterstützt.
- Online- oder Offline-Reorganisationen von Tabellen, die zwischen den Abruf- und Aktualisierungsoperationen in einem Szenario mit optimistischem Sperren ausgeführt werden, können dazu führen, dass die Aktualisierung fehlschlägt. Dies sollte jedoch durch die normale Wiederholungslogik von Anwendungen aufgefangen werden.
- In Version 9.5 ist das Attribut IMPLICITLY HIDDEN auf Spalten für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) zu Zwecken des optimistischen Sperrens beschränkt.
- Inplace-Reorganisationen sind für Tabellen, wenn diesen vorhandenen Tabellen eine Spalte ROW CHANGE TIMESTAMP hinzugefügt wurde, eingeschränkt, bis für alle Zeilen garantiert ein Wert gespeichert wurde (bei diesem Fehler wird SQL2219 mit Ursachencode 13 zurückgegeben). Dies lässt sich mit dem Befehl LOAD REPLACE bzw. mit einer klassischen Tabellenreorganisation erreichen, sodass *falsche Positivwerte* verhindert werden. Tabellen, die bereits mit der Spalte ROW CHANGE TIMESTAMP erstellt werden, haben keine Einschränkungen.

Aspekte für implizit verdeckte Spalten

Eine mit IMPLICITLY HIDDEN definierte Spalte gehört nicht zur Ergebnistabelle einer Abfrage, in der eine SELECT-Liste mit der Notation '*' angegeben wird. Dennoch kann eine implizit verdeckte Spalte in einer Abfrage explizit angegeben werden.

Wenn in der Einfügung keine Spaltenliste angegeben wird, darf die Klausel VALUES oder die SELECT-Liste für die Einfügung diese Spalte nicht enthalten (im Allgemeinen muss es sich bei dieser Spalte um eine generierte Spalte handeln, die einen Standardwert hat oder die Nullwerte enthalten darf).

Eine implizit verdeckte Spalte kann zum Beispiel in der SELECT-Liste oder in einem Vergleichselement einer Abfrage angegeben werden. Darüber hinaus kann eine implizit verdeckte Spalte auch in einer Anweisung CREATE INDEX, ALTER TABLE, INSERT, MERGE oder UPDATE explizit angegeben werden. Eine implizit verdeckte Spalte kann in einer referenziellen Integritätsbedingung angegeben werden. Eine Klausel REFERENCES, die keine Spaltenliste enthält, bezieht sich impli-

zit auf den Primärschlüssel der übergeordneten Tabelle. Es ist möglich, dass der Primärschlüssel der übergeordneten Tabelle eine Spalte enthält, die als implizit verdeckt definiert ist. Eine solche referenzielle Integritätsbedingung ist zulässig.

- Wenn die SELECT-Liste des Fullselects der MQT-Definition explizit eine implizit verdeckte Spalte angibt, wird diese Spalte in die MQT (Materialized Query Table, gespeicherte Abfragetabelle) eingefügt. Ansonsten ist eine implizit verdeckte Spalte nicht Teil einer MQT, die sich auf eine Tabelle bezieht, die eine implizit verdeckte Spalte enthält.
- Wenn die SELECT-Liste des Fullselects einer Sichtdefinition (Anweisung CREATE VIEW) explizit eine implizit verdeckte Spalte angibt, wird diese Spalte in die Sicht eingefügt (jedoch wird die Spalte in der Sicht nicht als 'verdeckt' betrachtet). Ansonsten ist eine implizit verdeckte Spalte nicht Teil einer Sicht, die sich auf eine Tabelle bezieht, die eine implizit verdeckte Spalte enthält.

Aspekte für die kennsatzbasierte Zugriffssteuerung (LBAC)

Wenn eine Spalte durch die kennsatzbasierte Zugriffssteuerung (LBAC - Label Based Access Control) geschützt wird, wird der Zugriff eines Benutzers auf diese Spalte durch die LBAC-Richtlinien und den Sicherheitskennsatz des Benutzers bestimmt. Dieser Schutz erstreckt sich, sofern er sich auf eine Spalte für die Zeilenänderungszeitmarke bezieht, auf den Verweis auf diese Spalte durch die Ausdrücke ROW CHANGE TIMESTAMP und ROW CHANGE TOKEN, die aus dieser Spalte abgeleitet werden.

Bei der Festlegung der Sicherheitsrichtlinien für eine Tabelle muss daher sichergestellt werden, dass der Zugriff auf die Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) für alle Benutzer verfügbar ist, die mit dem optimistischen Sperren bzw. der zeitbasierten Aktualisierungserkennung arbeiten müssen. Beachten Sie, dass bei einer fehlenden Spalte für die Zeilenänderungszeitmarke der Ausdruck ROW CHANGE TOKEN nicht durch LBAC blockiert werden kann. Wenn die Tabelle jedoch geändert wird, um eine Spalte für die Zeilenänderungszeitmarke hinzuzufügen, gelten alle Aspekte der kennsatzbasierten Zugriffssteuerung (LBAC).

Granularität von Zeilenänderungstoken und *falsche negative Werte*

Die integrierte Funktion RID_BIT() und das Zeilenänderungstoken sind die einzigen Anforderungen für das optimistische Sperren. Allerdings wirkt sich das Schema der Tabelle auch auf das Verhalten des optimistischen Sperrens aus.

Beispiel: Eine Zeitmarkenspalte für Zeilenänderung, die so definiert wurde, dass sie eine der folgenden Anweisungsklauseln verwendet, veranlasst den DB2-Server zur Speicherung der Zeit der letzten Änderung (oder ersten Einfügung) einer Zeile. Dies ist eine Möglichkeit, die Zeitmarke der allerletzten Änderung einer Zeile zu erfassen. Dabei handelt es sich um eine Zeitmarkenspalte, die vom Datenbankmanager verwaltet wird, es sei denn, die Klausel GENERATED BY DEFAULT wird zum Akzeptieren eines vom Benutzer bereitgestellten Eingabewerts verwendet.

```
GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
GENERATED BY DEFAULT FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
```

Wenn eine Anwendung also den neuen Ausdruck ROW CHANGE TOKEN in einer Tabelle verwendet, sind die folgenden beiden Möglichkeiten zu beachten:

- *Die Tabelle verfügt über keine Zeitmarkenspalte für Zeilenänderung:* Der Ausdruck ROW CHANGE TOKEN gibt den abgeleiteten Wert BIGINT zurück, der von allen Zeilen gemeinsam genutzt wird, die sich auf derselben Seite befinden.

Wenn eine einzige Zeile auf einer Seite aktualisiert wird, wird das Zeilenänderungstoken für alle Zeilen auf derselben Seite geändert. Dies bedeutet, dass eine Aktualisierung fehlschlagen kann, wenn Änderungen an anderen Zeilen vorgenommen werden. Dieses Merkmal wird als 'falscher negativer Wert' bezeichnet.

Anmerkung: Verwenden Sie diesen Modus nur, wenn die Anwendung *falsche negative Werte* tolerieren kann und nicht jeder Zeile für eine Zeitmarkenspalte für Zeilenänderung (ROW CHANGE TIMESTAMP) zusätzlichen Speicher hinzufügen will.

- *Die Tabelle verfügt über eine Zeitmarkenspalte für Zeilenänderung:* Der Ausdruck ROW CHANGE TOKEN gibt den Wert BIGINT zurück, der von dem Zeitmarkenwert in der Spalte abgeleitet wird. In diesem Fall können *falsche negative Werte* auftreten, jedoch relativ selten: Wenn die Tabelle reorganisiert oder umverteilt wird, können *falsche negative Werte* auftreten, wenn die Zeile versetzt wird und eine Anwendung den früheren Wert von RID_BIT() verwendet.

Zeitbasierte Aktualisierungserkennung

Für einige Anwendungen sind Datenbankaktualisierungen für bestimmte Zeitbereiche erforderlich, die möglicherweise zur Datenreplikation, für Protokollierungsszenarios usw. verwendet werden. Diese Informationen werden vom neuen Ausdruck ROW CHANGE TIMESTAMP bereitgestellt.

Er gibt eine Zeitmarke zurück, die die Zeit der letzten Änderung einer Zeile darstellt. Diese Zeit wird als lokale Zeit ähnlich wie bei CURRENT TIMESTAMP ausgedrückt. Für eine aktualisierte Zeile wird dadurch die letzte Aktualisierung der Zeile wiedergegeben. Ansonsten entspricht der Wert der ursprünglichen Einfügung der Zeile.

Der Wert für ROW CHANGE TIMESTAMP ist für jede Zeile pro Tabelle einer Datenbank oder Tabellenpartition eindeutig. Das bedeutet, dass nicht alle Zeilen in einer Datenbankpartition eindeutige Werte haben, sondern nur die Zeilen in derselben Tabelle. Der Wert stellt die Änderungsreihenfolge der Zeile dar. Zeilen, die später geändert wurden, haben immer spätere Werte als Zeilen, die früher geändert wurden. Da der Wert stets von früher zu später wächst, kann er unter folgenden Umständen vom Wert der Systemuhr abweichen:

- Wenn die Systemuhr geändert wurde
- Wenn die Zeitmarkenspalte für Zeilenänderung GENERATED BY DEFAULT lautet (nur für die Datenweitergabe gedacht) und eine Zeile mit einem Wert bereitgestellt wird, der nicht synchron ist

Als Voraussetzung für die Verwendung des Ausdrucks ROW CHANGE TIMESTAMP muss die Tabelle über eine definierte Zeitmarkenspalte für Zeilenänderung verfügen. Jede Zeile gibt die Zeitmarke ihrer Einfügung oder ihrer letzten Aktualisierung zurück. Es gibt zwei Methoden, durch die die Zeitmarkenspalte für Zeilenänderung Teil der Tabelle sein kann:

- *Die Tabelle wurde mit einer Zeitmarkenspalte für Zeilenänderung erstellt.* Der Ausdruck ROW CHANGE TIMESTAMP gibt den Wert der Spalte zurück. Für diese Kategorie ist die Zeitmarke genau. Allgemein gilt, dass die Zeitmarke für Zeilenänderung, wenn sie durch die Datenbank generiert wird, von der Geschwindigkeit von Einfügeoperationen sowie von möglichen Systemuhrumstellungen, zum Beispiel aufgrund der Sommerzeit, beeinflusst wird.
- *Die Tabelle wurde nicht mit einer Zeitmarkenspalte für Zeilenänderung erstellt,* jedoch wurde ihr mit der Anweisung ALTER TABLE zu einem späteren Zeitpunkt eine hinzugefügt. Der Ausdruck ROW CHANGE TIMESTAMP gibt den Wert der Spalte zurück. Bei dieser Kategorie erhalten die alten Zeilen (d. h. die Zeilen vor

der ALTER-Operation) die tatsächliche Zeitmarke erst, wenn sie das erste Mal aktualisiert werden oder eine Offlinetabellenreorganisation durchgeführt wird.

Anmerkung: Diese Zeitmarke ist ein ungefährer Wert für den Zeitpunkt, zu dem die tatsächliche Aktualisierung in der Datenbank stattfand. Dieser Wert basiert auf der Systemuhr zum jeweiligen Zeitpunkt und berücksichtigt die Einschränkung, dass keine Zeitmarke innerhalb einer Datenbank- bzw. Tabellenpartition wiederholt werden kann. In der Praxis ist dies in der Regel eine sehr genaue Darstellung der Zeit der Aktualisierung. Allgemein gilt, dass die Zeitmarke für Zeilenänderung, wenn sie durch die Datenbank generiert wird, von der Geschwindigkeit von Einfügeoperationen sowie von möglichen Systemuhrumstellungen, zum Beispiel aufgrund der Sommerzeit, beeinflusst wird.

Zeilen, die seit der Anweisung ALTER TABLE nicht aktualisiert wurden, geben den Standardwert für den Typ der Spalte zurück. Dieser Standardwert ist Mitternacht, 01. Januar, Jahr 1. Nur Zeilen, die aktualisiert wurden, haben eine eindeutige Zeitmarke. Zeilen, bei denen die Zeitmarke über eine Offlinetabellenreorganisation gespeichert wird, geben eine eindeutige Zeitmarke zurück, die bei der Reorganisation der Tabelle generiert wird. Die Verwendung des Befehls REORG mit der Option INPLACE genügt nicht, da Schemaänderungen damit nicht gespeichert werden.

In beiden Fällen kann die Zeitmarke auch aktualisiert werden, wenn eine Umverteilung durchgeführt wird. Wenn die Zeile bei der Umverteilung von einer Datenbankpartition in eine andere versetzt wird, muss eine neue Zeitmarke generiert werden, die für das Ziel garantiert eindeutig ist.

Für ROW CHANGE TIMESTAMP generierte Zeitwerte

Es gibt einige Grenzwertbedingungen zu den genauen Werten, die für die Zeitmarkenspalten für Zeilenänderung aufgrund der Umsetzung eindeutiger Werte pro Partition generiert wurden.

Wenn die Systemuhr in der Vergangenheit wegen Uhrzeitfehlern oder einer Sommerzeitrichtlinie auf dem DB2-Server angepasst wird, werden Zeitmarken möglicherweise in der Zukunft relativ zum aktuellen Wert der Systemuhr oder dem Wert des Sonderregisters CURRENT TIMESTAMP angezeigt. Dazu kommt es, wenn eine Zeitmarke vor der Anpassung der Systemuhr generiert wurde, d. h., später als die angepasste Zeit, da die Zeitmarken zur Wahrung der Eindeutigkeit immer aufsteigend generiert werden.

Wenn Zeitmarken für Spalten generiert werden, die der Tabelle durch eine REORG-Operation oder im Rahmen einer LOAD-Operation hinzugefügt wurden, werden die Zeitmarken an einem Punkt während der Verarbeitung des Dienstprogramms nacheinander generiert; dabei wird bei einem Anfangszeitmarkenwert begonnen. Wenn das Dienstprogramm Zeilen schneller als die Zeitmarkengranularität (mehr als 1 Million Zeilen pro Sekunde) verarbeiten kann, können die für einige Zeilen generierten Werte in der Zukunft relativ zur Systemuhr oder dem Sonderregister CURRENT TIMESTAMP sein.

In jedem Fall gibt es eine enge Annäherung an die Zeit, zu der die Zeile eingefügt wurde, wenn die Systemuhr dieselben Werte für Zeitmarkenspalten für Zeilenänderung hat. Bis zu dieser Zeit werden Zeitmarken in aufsteigender Reihenfolge auf der Basis der feinsten Granularität, die für den Zeitmarkentyp zulässig ist, generiert.

Integrierte Funktion RID_BIT() und RID()

Die Funktion RID_BIT() und das *Zeilenänderungstoken* (engl. *row change token*) können für jede Zeile in einer Tabelle ausgewählt werden. Die SELECT-Operation kann auf einer beliebigen Isolationsstufe erfolgen, die für die Anwendung erforderlich ist.

Die Anwendung kann dieselbe (ungeänderte) Zeile mit optimistischem Sperren aktualisieren (UPDATE), indem sie unter Verwendung folgender Elemente sucht:

- Die Funktion RID_BIT(), um direkt auf die zu aktualisierende Zielzeile zuzugreifen (ohne Suche)
- Das Zeilenänderungstoken, um sicherzustellen, dass es sich um dieselbe ungeänderte Zeile handelt

Diese Aktualisierung (oder Löschung) kann zu einem beliebigen Zeitpunkt nach der Auswahl (SELECT) innerhalb derselben UOW (Unit of Work, Arbeitseinheit) erfolgen oder sogar über Verbindungsgrenzen hinweg. Die einzige Voraussetzung besteht darin, dass die beiden oben genannten Werte für eine bestimmte Zeile zu einem früheren Zeitpunkt abgerufen worden sein müssen.

Die Technik des optimistischen Sperrens wird im „WebSphere-Oriented Programming Model" verwendet. Zum Beispiel verwendet Microsoft .NET dieses Modell, um SELECT-Anweisungen, auf die UPDATE- oder DELETE-Anweisungen folgen, wie folgt zu verarbeiten:

- Die Verbindung zum Datenbankserver wird hergestellt, und die gewünschten Zeilen werden per SELECT aus einer Tabelle ausgewählt.
- Die Verbindung zur Datenbank wird getrennt oder die Zeilensperren werden freigegeben, sodass andere Anwendungen Daten lesen, aktualisieren, löschen und einfügen können, ohne auf Konflikte beim gemeinsamen Zugriff aufgrund von Sperren und Ressourcen zu treffen, die von der Anwendung genutzt werden. (Die Isolationsstufe UR („Uncommited Read" - nicht festgeschriebener Lesevorgang) lässt einen höheren gemeinsamen Zugriff zu und unter der Annahme, dass andere Anwendungen ihre Aktualisierungs- und Löschtransaktionen festschreiben (COMMIT), liest diese Anwendung mit optimistischem Sperren die aktualisierten Werte und die optimistisch gesuchte Aktualisierung bzw. Löschung wird erfolgreich ausgeführt.)
- Es werden einige lokale Berechnungen an den ausgewählten Zeilendaten ausgeführt.
- Die Verbindung zum Datenbankserver wird erneut hergestellt, und es erfolgt eine Suche in einer oder mehreren Zielzeilen für die UPDATE- oder DELETE-Operation. (Wenn die Zielzeile geändert wurde, schlagen die UPDATE- oder DELETE-Anweisungen fehl und werden entsprechend verarbeitet).

Anwendungen, die dieses Programmiermodell verwenden, können von der verbesserten Funktion für optimistisches Sperren profitieren. Beachten Sie, dass Anwendungen, die dieses Programmiermodell nicht verwenden, nicht als Anwendungen mit optimistischem Sperren betrachtet werden, und dass solche Anwendungen weiterhin wie zuvor funktionieren.

Merkmale der integrierten Funktion RID_BIT() und RID()

Die folgenden neuen Funktionen werden zum Zweck eines verbesserten optimistischen Sperrens und der Aktualisierungserkennung implementiert:

RID_BIT(<tabellenbezeichnung>)
Eine neue integrierte Funktion, die die Satzkennung (RID, Record Identifier) einer Zeile mit dem Datentyp VARCHAR(16) FOR BIT DATA zurückgibt.

Anmerkung: DB2 für z/OS implementiert eine integrierte Funktion RID mit dem Rückgabetyp BIGINT, der jedoch für Linux-, UNIX- und Windows-RIDs nicht groß genug ist. Aus Kompatibilitätsgründen gibt diese integrierte Funktion RID() ebenso wie die Funktion RID_BIT() Daten des Typs BIGINT zurück.

Die integrierte Funktion RID() funktioniert nicht in DPF-Umgebungen und liefert keine Tabellenversionsinformationen. Ansonsten funktioniert sie in der gleichen Weise wie die Funktion RID_BIT. Sie sollten diese Funktion nur zur Codierung von Anwendungen verwenden, die auf z/OS-Server portiert werden sollen. Die Ausführungen dieses Abschnitts beziehen sich nur auf die Funktion RID_BIT, sofern nichts anderes erforderlich ist.

Integrierte Funktion RID_BIT()
Diese integrierte Funktion kann in der Anweisung der SELECT-Liste oder der Vergleichselemente verwendet werden. In einem Vergleichselement (z. B. `WHERE RID_BIT(tabelle)=?`) wird das Gleichheitsvergleichselement mit RID_BIT als neue direkte Zugriffsmethode implementiert, um die Zeile effizient zu lokalisieren. Früher wurde eine VALUES-Methode zum *optimistischen Sperren mit Werten* angewendet, bei der alle ausgewählten Spaltenwerte den Vergleichselementen hinzugefügt wurden. Diese weniger effiziente Zugriffsmethode war davon abhängig, dass einige eindeutige Spaltenkombinationen jeweils nur eine Zeile kennzeichneten.

ROW CHANGE TOKEN FOR <tabellenbezeichnung>
Ein neuer Ausdruck, der ein Token vom Typ BIGINT zurückgibt. Das Token stellt einen relativen Punkt in der Änderungsabfolge einer Zeile dar. Eine Anwendung kann den aktuellen Wert des Zeilenänderungstokens einer Zeile mit dem Wert des Zeilenänderungstokens vergleichen, der beim letzten Abruf der Zeile gespeichert wurde, um zu ermitteln, ob die Zeile inzwischen geändert wurde.

Spalte ROW CHANGE TIMESTAMP
Eine generierte Spalte (GENERATED) mit dem Standarddatentyp TIMESTAMP, die wie folgt definiert werden kann:

```
GENERATED ALWAYS FOR EACH ROW ON UPDATE
AS ROW CHANGE TIMESTAMP
```

oder (nur für Operationen zur Datenweitergabe bzw. zum Entladen und erneuten Laden empfohlen):

```
GENERATED BY DEFAULT FOR EACH ROW ON UPDATE
AS ROW CHANGE TIMESTAMP
```

Die Daten in dieser Spalte ändern sich bei jeder Änderung der Zeile. Wenn diese Spalte definiert ist, wird der ROW CHANGE TOKEN-Wert aus dieser Spalte abgeleitet. Beachten Sie, dass der Datenbankmanager bei Angabe von GENERATED ALWAYS dafür sorgt, dass dieser Wert innerhalb der Datenbankpartition bzw. der Tabellenpartition eindeutig ist, um sicherzustellen, dass keine *falschen Positivwerte* zugelassen werden.

Zur Verwendung der ersten beiden Elemente RID_BIT und ROW CHANGE TOKEN sind keine weiteren Änderungen am Datenbankschema erforderlich.

Beachten Sie jedoch, dass ohne die Spalte ROW CHANGE TIMESTAMP das Zeilenänderungstoken von allen Zeilen auf der gleichen Seite gemeinsam verwendet wird. Aktualisierungen an einer Zeile auf der Seite können *falsche Negativwerte* für andere, auf derselben Seite gespeicherten Zeilen zur Folge haben. Mit dieser Spalte wird das Zeilenänderungstoken (ROW CHANGE TOKEN) aus der Zeitmarke abgeleitet und von keinen anderen Zeilen in der Tabelle bzw. der Datenbankpartition mit verwendet. Siehe „Granularität von Zeilenänderungstoken und *falsche negative Werte*" auf Seite 280.

Funktion zur zeitbasierten Aktualisierungserkennung

Ein neuer Ausdruck, der einen Zeitmarkenwert zurückgibt, der den Zeitpunkt darstellt, zu dem die Zeile in der durch die Tabellenbezeichnung angegebenen Tabelle zuletzt geändert wurde.

```
ROW CHANGE TIMESTAMP FOR <tabellenbezeichnung>
```

Der Ausdruck ROW CHANGE TIMESTAMP wird nicht für Tabellen unterstützt, die keine Spalte ROW CHANGE TIMESTAMP enthalten.

Der Ausdruck ROW CHANGE TIMESTAMP wird nur für Szenarios mit zeitbasierter Aktualisierungserkennung verwendet und setzt voraus, dass eine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) für die Tabelle definiert ist, die durch die Tabellenbezeichnung angegeben wird. Diese Spalte wird vom Datenbankmanager verwaltet und dient zum Speichern des Zeitmarkenwerts, der durch den Ausdruck ROW CHANGE TIMESTAMP zurückgegeben wird. Diese Zeitmarke unterscheidet sich von der aktuellen Zeitmarke (CURRENT TIMESTAMP) insofern, als dass sie garantiert eindeutig ist, wenn sie von der Datenbank pro Zeile pro Datenbankpartition zugewiesen wird. Es handelt sich um eine lokale Zeitmarkennäherung des Änderungszeitpunkts jeder einzelnen Zeile, die eingefügt oder aktualisiert wird.

Anmerkung: Trotz der gegenseitigen Beziehung dieser beiden Funktionsmerkmale, das heißt, der integrierten Funktion RID_BIT() bzw. RID() und der Funktion zur zeitbasierten Aktualisierungserkennung, ist es wichtig zu beachten, dass die Ausdrücke ROW CHANGE TOKEN und ROW CHANGE TIMESTAMP nicht gegenseitig austauschbar verwendet werden können. Insbesondere ist zu beachten, dass der Ausdruck ROW CHANGE TIMESTAMP kein Bestandteil der Verwendung des optimistischen Sperrens ist.

Planen der Aktivierung des optimistischen Sperrens

Seitdem sich die neuen SQL-Ausdrücke und Attribute für optimistisches Sperren ohne DDL-Änderungen an den Tabellen verwenden lassen, können Sie das optimistische Sperren problemlos in Ihren Testanwendungen ausprobieren.

Beachten Sie, dass Anwendungen mit optimistischem Sperren ohne DDL-Änderungen möglicherweise mehr *falsche Negativwerte* empfangen als mit DDL-Änderungen. Eine Anwendung, die falsche Negativwerte empfängt, lässt sich in einer Produktionsumgebung eventuell nicht gut skalieren, weil die falschen Negativwerte zu viele Wiederholungsversuche verursachen. Zur Vermeidung falscher Negativwerte sollten Zieltabellen mit optimistischem Sperren wie folgt behandelt werden:

- Sie sollten mit einer Spalte ROW CHANGE TIMESTAMP erstellt werden.
- Sie sollten geändert werden, sodass sie eine Spalte ROW CHANGE TIMESTAMP enthalten.

Wenn die empfohlenen DDL-Änderungen ausgeführt wurden, verringert sich die Anzahl der falschen Negativwerte. Die einzigen falschen Negativwerte, die möglicherweise auftreten, werden die Folge von Operationen auf Tabellenebene sein, wie zum Beispiel eine Reorganisation (REORG), bei der gleichzeitig ausgeführte Anwendungen verschiedene Zeilen bearbeiten.

Im Allgemeinen lässt der Datenbankmanager falsche Negativwerte (z. B. bei Online- oder Offline-Reorganisation) zu, und das Vorhandensein einer Spalte für die Zeilenänderungszeitmarke (ROWCHANGETIMESTAMP) ist ausreichend, um zu bestimmen, ob die Granularität auf Seiten- oder Zeilenebene verwendet wird. Sie können auch SYSCAT.COLUMNS nach einer Tabelle abfragen, die Zeilen mit dem Wert YES in der Spalte ROWCHANGETIMESTAMP enthält.

Eine gründliche Analyse der Anwendung und der Datenbank kann möglicherweise zu dem Ergebnis kommen, dass diese DDL-Anweisungen nicht erforderlich sind, wenn zum Beispiel nur eine Zeile pro Seite vorhanden ist oder wenn die Aktualisierungs- und Löschoperationen nur sporadisch und selten oder auch nie an derselben Datenseite ausgeführt werden. Eine solche Analyse ist jedoch die Ausnahme.

Für den Zweck der Erkennung von Aktualisierungszeitmarken müssen Sie Änderungen an den DDL-Anweisungen für die Tabelle vornehmen und möglicherweise die Tabelle reorganisieren, um die Werte real zu speichern. Wenn Sorge besteht, dass diese Änderungen negative Auswirkungen auf die Produktionsdatenbank haben könnten, sollten die Änderungen zunächst an Prototypen in einer Testumgebung getestet werden. Zum Beispiel können sich die zusätzlichen Spalten auf die Zeilengrößenbegrenzen und die Planauswahl auswirken.

Zu beachtende Bedingungen

- Sie sollten sich über Bedingungen im Klaren sein, die die Systemuhr und die Granularität der Zeitmarkenwerte betreffen. Wenn eine Tabelle eine Spalte ROW CHANGE TIMESTAMP besitzt, enthält die neue Zeile nach einer Einfügung oder Aktualisierung einen eindeutigen ROW CHANGE TIMESTAMP-Wert in dieser Tabelle in dieser Datenbankpartition.
- Zur Sicherstellung der Eindeutigkeit wird die generierte Zeitmarke einer Zeile immer erhöht, unabhängig davon, ob die Systemuhr zurückgestellt wird oder ob die Aktualisierung bzw. Einfügung von Daten schneller erfolgt, als dies durch die Granularität der Zeitmarke erfasst wird. Daher kann ROW CHANGE TIMESTAMP in Relation zur Systemzeit und zum DB2-Sonderregister CURRENT TIMESTAMP einen Wert in der Zukunft enthalten. Sofern die Systemuhr nicht vollkommen aus der Synchronisation gerät oder der Datenbankmanager nicht mehr als eine Million Zeilen pro Sekunde einfügt oder aktualisiert, sollte dieser Wert in der Regel sehr nahe an der tatsächlichen Zeit liegen. Im Unterschied zum Wert des Sonderregisters CURRENT TIMESTAMP wird dieser Wert außerdem pro Zeile zum Zeitpunkt der Aktualisierung generiert, sodass er normalerweise wesentlich näher an der Realzeit liegt als der Wert in CURRENT TIMESTAMP, der nur einmal für die gesamte Anweisung generiert wird, deren Ausführung je nach Komplexität und Anzahl der betroffenen Zeilen geraume Zeit in Anspruch nehmen kann.

Aktivieren des optimistischen Sperrens in Anwendungen

Zur Aktivierung der Unterstützung für optimistisches Sperren in Anwendungen müssen Sie eine Reihe von Schritten ausführen.

1. Rufen Sie in der einleitenden Abfrage mit SELECT die Satz-ID (RID, siehe „Integrierte Funktion RID_BIT() und RID()" auf Seite 283) und das Zeilenänderungstoken (ROW CHANGE TOKEN) für jede Zeile ab, die Sie verarbeiten müssen.
2. Geben Sie die Zeilensperren frei, sodass andere Anwendungen SELECT-, INSERT-, UPDATE- und DELETE-Anweisungen an der Tabelle ausführen können.
3. Führen Sie eine UPDATE- oder DELETE-Anweisung durch eine gezielte Suche mit der Satz-ID und dem Zeilenänderungstoken in der Suchbedingung an den Zielzeilen aus, wobei Sie optimistisch annehmen, dass die nicht gesperrte Zeile seit Ausführung der ursprünglichen SELECT-Anweisung nicht geändert wurde.
4. Falls die Zeile geändert wurde, schlägt die UPDATE-Operation fehl und die Anwendungslogik muss den Fehler behandeln. Die Anwendung kann zum Beispiel die SELECT- und die UPDATE-Operation wiederholen.

Nach der Ausführung der obigen Schritte:

- Wenn die Anzahl der von Ihrer Anwendung ausgeführten Wiederholversuche höher scheint als erwartet oder als gewünscht wird, können Sie Ihrer Tabelle eine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) hinzufügen, um sicherzustellen, dass nur Änderungen an der Zeile, die von der Funktion RID_BIT ermittelt wird, nur das Zeilenänderungstoken ungültig machen, und keine anderen Aktivitäten an derselben Seite.
- Zum Anzeigen von Zeilen, die innerhalb eines bestimmten Zeitrahmens eingefügt oder aktualisiert wurden, erstellen oder ändern Sie die Tabelle in der Weise, dass sie eine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) enthält. Diese Spalte wird automatisch vom Datenbankmanager gepflegt und kann entweder über den Spaltennamen oder mit dem Ausdruck ROW CHANGE TIMESTAMP abgefragt werden.
- Nur für Spalten der Zeilenänderungszeitmarke gilt, dass die Spalte, wenn sie mit dem Attribut IMPLICTLY HIDDEN als implizit verdeckt definiert wurde, nicht extern bereitgestellt wird, wenn ein impliziter Verweis auf die Spalten der Tabelle erfolgt. Eine implizit verdeckte Spalte kann in SQL-Anweisungen jedoch immer explizit angegeben werden. Dies kann nützlich sein, wenn eine Spalte, die einer Tabelle hinzugefügt wird, dazu führt, dass vorhandene Anwendungen mit impliziten Spaltenlisten fehlschlagen.

Tabellenpartitionierung und Datenorganisationsschemata

Die Tabellenpartitionierung ist ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Datenpartitionen entsprechend den Werten einer oder mehrerer Partitionierungsspalten der Tabelle verteilt werden. Daten aus einer Tabelle werden in mehrere Speicherobjekte partitioniert, die sich in verschiedenen Tabellenbereichen befinden können.

Vollständige und detaillierte Informationen zur Tabellenpartitionierung und zu Datenorganisationsschemata finden Sie im Handbuch *Partitionierung und Clustering*.

Erstellen von Tabellen

Der Datenbankmanager steuert den Zugriff auf die in den Tabellen gespeicherten Daten und deren Änderungen. Tabellen können mit der Anweisung CREATE TABLE erstellt werden. Mithilfe komplexer Anweisungen können alle Attribute und Merkmale von Tabellen definiert werden. Wenn jedoch alle Standardwerte genutzt werden, ist die Anweisung zur Erstellung einer Tabelle recht einfach.

```
CREATE TABLE <tabellenname> (<spaltenname> <datentyp> <spaltenoptionen>,
                             (<spaltenname> <datentyp> <spaltenoptionen>, ...)
```

Die Angabe <tabellenname> kann optional ein Qualifikationsmerkmal enthalten. Der Name muss beim Vergleich mit allen Tabellen-, Sicht- und Aliasnamen im Systemkatalog eindeutig sein. Darüber hinaus darf der Name nicht SYSIBM, SYSCAT, SYSFUN oder SYSSTAT sein.

Die Angabe <spaltenname> benennt die Spalten in der Tabelle. Dieser Name kann nicht qualifiziert werden und muss unter allen anderen Spalten der Tabelle eindeutig sein.

Mit der Angabe <spaltenoptionen> für eine Spalte werden weitere Attribute der Spalte definiert. Zu den Optionen gehören die Option NOT NULL, mit der verhindert wird, dass Nullwerte in der Spalte gespeichert werden, bestimmte Optionen für LOB-Datentypen und die Angabe SCOPE (Bereich) für Verweistypspalten sowie Integritätsbedingungen und Standardwerte für die Spalten. Weitere Informationen finden Sie in der Beschreibung der Anweisung CREATE TABLE.

Deklarieren globaler temporärer Tabellen

Zur Erstellung globaler temporärer Tabellen aus Anwendungen heraus verwenden Sie die Anweisung DECLARE GLOBAL TEMPORARY TABLE.

Globale temporäre Tabellen, die auch als benutzerdefinierte temporäre Tabellen bezeichnet werden, werden von Anwendungen verwendet, die mit Daten in der Datenbank arbeiten. Ergebnisse aus der Bearbeitung der Daten müssen temporär in einer Tabelle gespeichert werden. Ein Tabellenbereich für temporäre Benutzertabellen muss vorhanden sein, bevor globale temporäre Tabellen erstellt werden.

Anmerkung: Die Beschreibung globaler temporärer Tabellen wird im Systemkatalog nicht aufgeführt, sodass sie von anderen Anwendungen nicht identifiziert und gemeinsam verwendet werden kann. Wenn die Anwendung, die diese Tabelle verwendet, beendet wird oder die Verbindung zur Datenbank trennt, werden alle Daten in dieser Tabelle gelöscht und auch die Tabelle wird implizit gelöscht. Folgende Spalten werden von globalen temporären Tabellen nicht unterstützt:

- LOB-Spalten (oder Spalten eines einzigartiger Datentyps, die ebenfalls auf LOB basieren)
- Spalten mit benutzerdefinierten Datentypen
- LONG VARCHAR-Spalten
- XML-Spalten

Beispiel

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

Diese Anweisung erstellt eine temporäre Tabelle mit dem Namen 'gbl_temp'. Diese Tabelle wird mit Spalten definiert, die exakt dieselben Namen und Beschreibungen wie die Spalten der Tabelle 'empltabl' haben. Die implizite Definition umfasst nur den Spaltennamen, den Datentyp, das Merkmal für die Optionalität der Dateneingabe sowie die Standardwertattribute für die Spalte. Alle anderen Spaltenattribute einschließlich der eindeutigen Integritätsbedingungen, Trigger und Indizes werden nicht definiert. Bei der Durchführung eines Commits werden alle Daten in der

Tabelle gelöscht, wenn für die Tabelle kein WITH HOLD-Cursor geöffnet wurde. Die an der temporären Tabelle vorgenommenen Änderungen werden nicht protokolliert. Die temporäre Tabelle wird im angegebenen Tabellenbereich für temporäre Benutzertabellen gespeichert. Dieser Tabellenbereich muss vorhanden sein, da andernfalls die Deklaration der Tabelle fehlschlägt.

Wenn bei der Erstellung dieser Tabelle ROLLBACK oder ROLLBACK TO SAVEPOINT angegeben wird, können Sie entweder angeben, dass alle Zeilen in der Tabelle gelöscht werden sollen (DELETE ROWS, Standardwert), oder Sie können angeben, dass die Zeilen der Tabelle erhalten bleiben sollen (PRESERVE ROWS).

Die Tabelle wird implizit gelöscht, wenn die Anwendung die Verbindung zur Datenbank trennt.

Erstellen von Tabellen nach vorhandenen Tabellen

Die Erstellung einer neuen Quellentabelle kann erforderlich werden, wenn bei der Ausführung der Anweisung ALTER TABLE mit der Klausel ATTACH PARTITION die Merkmale der Zieltabelle nicht ausreichend mit den Merkmalen der Quelle übereinstimmen. Vor der Erstellung einer neuen Quellentabelle können Sie versuchen, die Abweichungen zwischen der vorhandenen Quellentabelle und der Zieltabelle zu korrigieren.

Zum Erstellen einer Tabelle müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen und eines der folgenden Zugriffsrechte beinhalten:

- Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für den Tabellenbereich sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
  - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
  - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht
- Berechtigung SYSADM oder DBADM

Wenn Versuche, die Abweichung zu korrigieren, fehlschlagen, wird der Fehler SQL20408N oder SQL20307N zurückgegeben.

Gehen Sie wie folgt vor, um eine neue Quellentabelle zu erstellen:

1. Generieren Sie mit dem Befehl db2look eine Anweisung CREATE TABLE zur Erstellung einer Tabelle, die mit der Zieltabelle identisch ist:

   ```
   db2look -d <quellendatenbankname> -t <zieldatenbankname> -e -p
   ```

2. Entfernen Sie die Partitionierungsklausel aus der Ausgabe von 'db2look', und ändern Sie den Namen der erstellten Tabelle in einen neuen Namen (in diesem Beispiel 'sourceC').
3. Laden Sie im nächsten Schritt alle Daten aus der ursprünglichen Quellentabelle in die neu erstellte Quellentabelle 'sourceC' mit dem Befehl LOAD FROM CURSOR:

   ```
   DECLARE mycurs CURSOR FOR SELECT * FROM source
   LOAD FROM mycurs OF CURSOR REPLACE INTO sourceC
   ```

Wenn dieser Befehl fehlschlägt, weil die ursprünglichen Daten mit der Definition der Tabelle 'sourceC' nicht kompatibel sind, müssen Sie die Daten in der ursprünglichen Tabelle beim Übertragen in die Tabelle 'sourceC' umwandeln.

4. Wenn die Daten erfolgreich in 'sourceC' übertragen wurden, führen Sie die Anweisung ALTER TABLE ziel ...ATTACH sourceC aus.

Erstellen von Tabellen zum Zwischenspeichern von Daten

Eine *Zwischenspeichertabelle* (engl. staging table) ermöglicht eine Unterstützung von Teilaktualisierungen (inkrementelle Pflege) für MQTs (Materialized Query Table, gespeicherte Abfragetabelle), die mit REFRESH DEFERRED definiert sind. Die Zwischenspeichertabelle sammelt Änderungen, die auf die MQT angewendet werden müssen, um diese mit dem Inhalt der zugrunde liegenden Tabellen zu synchronisieren. Durch die Nutzung von Zwischenspeichertabellen wird die starke Sperrenkonkurrenz beseitigt, die durch sofort zu aktualisierende Daten verursacht wird, wenn eine unverzügliche Aktualisierung (REFRESH IMMEDIATE) der MQT angefordert ist. Außerdem brauchen MQTs nicht länger völlig neu generiert zu werden, wenn ein Befehl REFRESH TABLE ausgeführt wird.

MQTs sind eine leistungsfähige Methode zum Verbessern der Antwortzeit komplexer Abfragen, insbesondere solcher Abfragen, die eventuell eine der folgenden Operationen erfordern:

- Ermitteln von Ergebnisdaten aus einer oder mehreren Dimensionen
- Joins und Ergebnisberechnung von Daten einer Gruppe von Tabellen
- Verarbeiten von Daten aus einer häufig verwendeten Teilmenge von Daten
- Erneutes Partitionieren von Daten aus einer Tabelle bzw. einem Teil einer Tabelle in einer partitionierten Datenbankumgebung

Es folgt eine Übersicht über einige der wichtigsten Einschränkungen für Zwischenspeichertabellen:

1. Die Abfrage, die zur Definition der MQT verwendet wird, für die die Zwischenspeichertabelle erstellt wird, muss inkrementell aktualisierbar sein. Das bedeutet, sie muss den gleichen Regeln wie eine MQT mit der Option zur sofortigen Aktualisierung (REFRESH IMMEDIATE) genügen.
2. Eine unterstützende Zwischenspeichertabelle ist nur für eine verzögerte Aktualisierung (REFRESH DEFERRED) möglich. Die Abfrage definiert auch die MQT, der die Zwischenspeichertabelle zugeordnet wird. Die MQT muss mit der Option REFRESH DEFERRED definiert sein.
3. Bei der Aktualisierung über die Zwischenspeichertabellen wird nur eine Aktualisierung bis zum aktuellen Zeitpunkt unterstützt.
4. Partitionierte Hierarchietabellen und partitionierte typisierte Tabellen werden nicht unterstützt. (Partitionierte Tabellen sind Tabellen, bei denen die Daten auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen in mehrere Speicherobjekte untergliedert werden.)

Eine Zwischenspeichertabelle in einem inkonsistenten, unvollständigen oder schwebenden (pending) Status kann nicht zur inkrementellen Aktualisierung (Teilaktualisierung) der zugeordneten MQT verwendet werden, sofern nicht einige weitere Operationen stattfinden. Diese Operationen sorgen dafür, dass der Inhalt der Zwischenspeichertabelle mit der ihr zugeordneten MQT und deren zugrunde liegenden Tabellen konsistent wird, und nehmen die Zwischenspeichertabelle aus dem schwebenden Status heraus. Nach der Aktualisierung einer MQT wird der Inhalt der zugehörigen Zwischenspeichertabelle gelöscht und die Zwischenspeichertabelle in den Normalstatus versetzt. Eine Zwischenspeichertabelle kann

mithilfe der Anweisung SET INTEGRITY und den entsprechenden Optionen auch gezielt abgeschnitten werden (PRUNE). Das Abschneiden versetzt die Zwischenspeichertabelle in einen inkonsistenten Status. Zum Beispiel erzwingt die folgende Anweisung das Abschneiden einer Zwischenspeichertabelle mit dem Namen STAGTAB1:

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

Eine Zwischenspeichertabelle wird bei ihrer Erstellung in einen schwebenden Status versetzt und besitzt einen Anzeiger, der darauf hinweist, dass die Tabelle inkonsistent und unvollständig im Hinblick auf den Inhalt der zugrunde liegenden Tabellen und der zugeordneten MQT ist. Die Zwischenspeichertabelle muss aus dem schwebenden und inkonsistenten Status herausgenommen werden, um mit der Erfassung der Änderungen aus den zugrunde liegenden Tabellen beginnen zu können. Während des schwebenden Status schlagen alle Versuche, Änderungen an einer beliebigen der zugrunde liegenden Tabellen der Zwischenspeichertabelle vorzunehmen, ebenso fehl wie alle Versuche, die zugeordnete MQT zu aktualisieren.

Eine Zwischenspeichertabelle kann durch mehrere Methoden aus einem schwebenden Status herausgenommen werden. Zum Beispiel:

- SET INTEGRITY FOR <zwischenspeichertabelle> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <zwischenspeichertabelle> IMMEDIATE CHECKED

Ändern von Tabellen

Dieser Abschnitt enthält Themen, die das Ändern von Tabellen erläutern.

Ändern der Merkmale einer MQT (Materialized Query Table)

Mit einigen Einschränkungen können Sie eine MQT (Materialized Query Table, gespeicherte Abfragetabelle) in eine reguläre Tabelle bzw. eine reguläre Tabelle in eine MQT ändern. Bei anderen Tabellentypen als regulären Tabellen und MQTs ist eine Änderung des Tabellentyps nicht möglich. Sie können zum Beispiel keine replizierte MQT in eine reguläre Tabelle oder umgekehrt ändern.

Nach dem Ändern einer regulären Tabelle in eine MQT wird die Tabelle in den Status 'Festlegen der Integrität anstehend' versetzt. Bei einer derartigen Änderung muss der `Fullselect` in der Definition der MQT mit der ursprünglichen Tabellendefinition übereinstimmen. Dies bedeutet Folgendes:

- Die Anzahl der Spalten muss gleich sein.
- Die Spaltennamen und -positionen müssen übereinstimmen.
- Die Datentypen müssen identisch sein.

Wenn die MQT für eine Originaltabelle definiert wird, kann diese Originaltabelle nicht selbst in eine MQT geändert werden. Wenn für die Originaltabelle Trigger, Prüfungen auf Integritätsbedingungen, referenzielle Integritätsbedingungen oder definierte eindeutige Indizes definiert sind, kann diese nicht in eine MQT geändert werden. Wenn Sie die Tabellenmerkmale ändern, um eine MQT zu definieren, ist es nicht zulässig, die Tabelle innerhalb derselben Anweisung ALTER TABLE in irgendeiner Form zu ändern.

Beim Ändern einer regulären Tabelle in eine MQT kann der `Fullselect` der Definition der MQT nicht direkt oder indirekt über Sichten, Aliasnamen oder andere MQTs auf die Originaltabelle verweisen.

Geben Sie Folgendes ein, um eine MQT in eine reguläre Tabelle zu ändern:

```
ALTER TABLE sumtable
   SET SUMMARY AS DEFINITION ONLY
```

Geben Sie Folgendes ein, um eine reguläre Tabelle in eine MQT zu ändern:

```
ALTER TABLE regtable
  SET SUMMARY AS <fullselect>
```

Die Einschränkungen, die beim Ändern einer regulären Tabelle in eine MQT für den `Fullselect` gelten, stimmen größtenteils mit den Einschränkungen überein, die für das Erstellen einer Übersichtstabelle mit der Anweisung CREATE SUMMARY TABLE gelten.

Aktualisieren der Daten in einer MQT (Materialized Query Table)

Sie können die Daten in einer oder mehreren MQTs (Materialized Query Table, gespeicherte Abfragetabelle) mithilfe der Anweisung REFRESH TABLE aktualisieren. Die Anweisung kann in ein Anwendungsprogramm eingebettet oder dynamisch abgesetzt werden. Zur Verwendung dieser Anweisung sind die Berechtigungen SYSADM oder DBADM bzw. das Zugriffsrecht CONTROL für die zu aktualisierende Tabelle erforderlich.

Das folgende Beispiel zeigt, wie Sie die Daten in einer MQT aktualisieren können:

```
REFRESH TABLE SUMTAB1
```

Ändern von Spaltenmerkmalen

Mithilfe der Anweisung ALTER TABLE können Sie Spaltenmerkmale wie die Optionalität der Dateneingabe, LOB-Optionen, Bereich, Attribute für Integritätsbedingungen und Komprimierung, Datentypen usw. ändern. Die vollständigen Informationen finden Sie in der Beschreibung der Anweisung ALTER TABLE.

Zum Ändern einer Tabelle müssen Sie mindestens über eines der folgenden Zugriffsrechte (bzw. Berechtigungen) für die zu ändernde Tabelle verfügen:

- Zugriffsrecht ALTER
- Zugriffsrecht CONTROL
- Berechtigung SYSADM oder DBADM
- Zugriffsrecht ALTERIN für das Schema der Tabelle

Sie müssen über die Berechtigung DBADM verfügen, um die Definition einer vorhandenen Spalte ändern, SQL beim Ändern von Tabellenspalten editieren und testen oder zugehörige Objekte beim Ändern von Tabellenspalten prüfen zu können.

Geben Sie über die Befehlszeile zum Beispiel Folgendes ein:

```
ALTER TABLE EMPLOYEE
   ALTER COLUMN WORKDEPT
      SET DEFAULT '123'
```

Hinzufügen und Löschen von Spalten

Mithilfe der Anweisung ALTER TABLE können Sie vorhandenen Tabellen Spalten hinzufügen, indem Sie die Klausel ADD COLUMN verwenden, und Spalten aus vorhandenen Tabellen entfernen, indem Sie die Klausel DROP COLUMN verwenden. Die Tabelle darf keine typisierte Tabelle sein.

In allen vorhandenen Zeilen der Tabelle wird die neue Spalte auf ihren Standardwert gesetzt. Die neue Spalte ist die letzte Spalte der Tabelle. Das heißt, wenn zuvor *n* Spalten vorhanden waren, ist die hinzugefügte Spalte die Spalte *n*+1. Durch das Hinzufügen der neuen Spalte darf die Gesamtbytezahl aller Spalten die Zeilengrößenbegrenzung nicht überschreiten.

Zum Hinzufügen einer Spalte führen Sie die folgende Anweisung aus:

```
ALTER TABLE SALES
  ADD COLUMN SOLD_QTY
  SMALLINT NOT NULL DEFAULT 0
```

Zum Löschen einer Spalte führen Sie die folgende Anweisung aus:

```
ALTER TABLE SALES
  DROP COLUMN SOLD_QTY
```

Ändern von Spaltendefinitionen mit der Klausel DEFAULT

Die Klausel DEFAULT stellt einen Standardwert für eine Spalte für den Fall bereit, dass bei einer Einfügung (INSERT) kein Wert angegeben wird, oder wird als DEFAULT in INSERT- oder UPDATE-Anweisungen angegeben. Wenn nach dem Schlüsselwort DEFAULT kein bestimmter Wert angegeben wird, hängt der Standardwert vom Datentyp ab. Ist eine Spalte mit dem Datentyp XML oder mit einem strukturierten Typ definiert, kann die Klausel DEFAULT nicht angegeben werden.

Wenn die Klausel DEFAULT in einer Spaltendefinition nicht angegeben wird, wird der Nullwert als Standardwert für die Spalte verwendet (siehe „Standardspalten- und Datentypdefinitionen” auf Seite 264).

Bestimmte Typen von Werten, die mit dem Schlüsselwort DEFAULT angegeben werden können, finden Sie in den Informationen zur Anweisung ALTER TABLE.

Modifizieren des Merkmals GENERATED oder IDENTITY einer Spalte

Das Merkmal GENERATED oder IDENTITY einer Spalte in einer Tabelle kann mithilfe der Klausel ALTER COLUMN der Anweisung ALTER TABLE hinzugefügt oder gelöscht werden.

Sie können eine der folgenden Aktionen ausführen:

- Wenn Sie mit einer vorhandenen, nicht generierten Spalte (ohne das Merkmal GENERATED) arbeiten, können Sie ein Attribut mit einem GENERATED-Ausdruck hinzufügen. Die modifizierte Spalte wird dann zu einer generierten Spalte.
- Wenn Sie mit einer vorhandenen generierten Spalte (mit dem Merkmal GENERATED) arbeiten, können Sie das Attribut mit dem GENERATED-Ausdruck löschen. Die modifizierte Spalte wird dann zu einer normalen, nicht generierten Spalte.
- Wenn Sie mit einer vorhandenen Nicht-Identitätsspalte (ohne das Merkmal IDENTITY) arbeiten, können Sie ein IDENTITY-Attribut hinzufügen. Die modifizierte Spalte wird dann zu einer Identitätsspalte.
- Wenn Sie mit einer vorhandenen Identitätsspalte arbeiten, können Sie das IDENTITY-Attribut löschen. Die modifizierte Spalte wird dann zu einer normalen, nicht generierten Nicht-Identitätsspalte.
- Wenn Sie mit einer vorhandenen generierten Spalte arbeiten, können Sie die generierte Spalte vom Attribut GENERATED ALWAYS in das Attribut GENERATED BY DEFAULT ändern. Der umgekehrte Fall ist ebenfalls möglich. Das heißt,

Sie können eine generierte Spalte vom Attribut GENERATED BY DEFAULT in das Attribut GENERATED ALWAYS ändern. Dies ist nur bei der Arbeit mit einer generierten Spalte möglich.

- Sie können das DEFAULT-Attribut aus einer benutzerdefinierten Standardspalte löschen. Wenn Sie dies tun, ist der neue Standardwert null.
- Sie können innerhalb derselben Anweisung ALTER COLUMN das vorhandene DEFAULT-, IDENTITY- oder GENERATED-Attribut löschen und anschließend ein neues DEFAULT-, IDENTITY- oder GENERATED-Attribut definieren.
- Für die Anweisungen CREATE TABLE und ALTER TABLE ist das Wort „ALWAYS" in der GENERATED-Klausel optional. Das heißt, in der Anweisung ALTER TABLE ist der Ausdruck GENERATED ALWAYS mit dem Ausdruck GENERATED äquivalent.

Ändern von Spaltendefinitionen

Mit der Anweisung ALTER TABLE können Sie Spalten löschen oder Typen und Attribute von Spalten ändern. Sie können zum Beispiel den Wert für die Länge einer vorhandenen VARCHAR- oder VARGRAPHIC-Spalte erhöhen. Die Anzahl der Zeichen kann bis zu einem Wert erhöht werden, der von der verwendeten Seitengröße abhängig ist.

Wenn Sie zum Ändern des Standardwerts einer Spalte den neuen Standardwert definiert haben, wird der neue Wert bei allen nachfolgenden SQL-Operationen verwendet, wenn die Verwendung des Standardwerts für die Spalte angegeben ist. Der neue Wert muss den Regeln für die Zuordnung entsprechen und unterliegt ebenfalls den Einschränkungen, die unter der Anweisung CREATE TABLE dokumentiert sind.

Anmerkung: Der Standardwert generierter Spalten kann mithilfe dieser Anweisung nicht geändert werden.

Wenn Sie diese Tabellenattribute mithilfe von SQL ändern, ist es nicht mehr erforderlich, die Tabelle zu löschen und erneut zu stellen, da dies ein zeitaufwendiger und komplexer Prozess sein kann, wenn Objektabhängigkeiten vorhanden sind.

Geben Sie in die Befehlszeile Folgendes ein, um die Länge und den Typ einer Spalte einer vorhandenen Tabelle zu ändern:

```
ALTER TABLE <tabellenname>
   ALTER COLUMN <spaltenname>
   <modifikationstyp>
```

Mit der folgenden Anweisung können Sie zum Beispiel die Zeichenzahl einer Spalte auf 4.000 Zeichen erhöhen:

```
ALTER TABLE t1
   ALTER COLUMN spalte1
   SET DATA TYPE VARCHAR(4000)
```

Das folgende Beispiel zeigt, wie Sie eine Anweisung wie die folgende verwenden, um einer Spalte einen neuen VARGRAPHIC-Wert zuzuordnen:

```
ALTER TABLE t1
   ALTER COLUMN spalte2
   SET DATA TYPE VARGRAPHIC(2000)
```

Sie können die Spalte einer typisierten Tabelle nicht ändern. Sie können einer vorhandenen Verweistypspalte ohne definierten Bereich jedoch einen Bereich hinzufügen. Beispiel:

```
ALTER TABLE t1
   ALTER COLUMN spaltet1
   ADD SCOPE typtab1
```

Geben Sie in die Befehlszeile Folgendes ein, um den Standardwert einer Spalte einer vorhandenen Tabelle zu ändern:

```
ALTER TABLE <tabellenname>
   ALTER COLUMN <spaltenname>
   SET DEFAULT 'neuer_standardwert'
```

Mit einer Anweisung wie der folgenden können Sie zum Beispiel den Standardwert für eine Spalte ändern:

```
ALTER TABLE t1
   ALTER COLUMN spalte1
   SET DEFAULT '123'
```

Umbenennen von Tabellen

Mithilfe der Anweisung RENAME können Sie eine vorhandene Tabelle umbenennen.

Wenn Tabellen umbenannt werden, dürfen keine vorhandenen Definitionen (von Sichten oder MQTs), Trigger, SQL-Funktionen oder Integritätsbedingungen auf die Quellentabelle verweisen. Darüber hinaus darf die Tabelle keine generierten Spalten (außer Identitätsspalten) enthalten, und sie darf keine übergeordnete oder abhängige Tabelle sein. Katalogeinträge werden mit dem neuen Tabellennamen aktualisiert. Weitere Informationen und Beispiele finden Sie in der Beschreibung der Anweisung RENAME.

Informationen zum Ändern der Definition vorhandener Spalten finden Sie in „Ändern von Spaltenmerkmalen” auf Seite 292 sowie in der Beschreibung zur Anweisung ALTER TABLE.

Recovery funktionsunfähiger Übersichtstabellen

Übersichtstabellen können infolge eines widerrufenen Zugriffsrechts SELECT in einer zugrunde liegenden Tabelle *funktionsunfähig* werden.

Die folgenden Schritte können Sie bei der Recovery einer funktionsunfähigen Übersichtstabelle unterstützen:

- Stellen Sie fest, mit welcher Anweisung die Übersichtstabelle zu Anfang erstellt wurde. Diese Information können Sie der Spalte TEXT der Katalogsicht SYSCAT.VIEW entnehmen.
- Erstellen Sie die Übersichtstabelle erneut, indem Sie die Anweisung CREATE SUMMARY TABLE mit demselben Übersichtstabellennamen und derselben Definition verwenden.
- Verwenden Sie die Anweisung GRANT, um alle Zugriffsrechte, die zuvor für die Übersichtstabelle erteilt waren, erneut zu erteilen. (Beachten Sie, dass alle für eine funktionsunfähig gewordene Übersichtstabelle erteilten Zugriffsrechte widerrufen werden.)

Wenn Sie eine funktionsunfähige Übersichtstabelle nicht wiederherstellen möchten, können Sie sie explizit mit der Anweisung DROP TABLE löschen, oder Sie können eine neue Übersichtstabelle mit demselben Namen, aber einer anderen Definition erstellen.

Eine funktionsunfähige Übersichtstabelle hat nur noch Einträge in den Katalogsichten SYSCAT.TABLES und SYSCAT.VIEWS. Alle Einträge in den Katalogsichten SYSCAT.TABDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS und SYSCAT.COLAUTH werden entfernt.

Anzeigen von Tabellendefinitionen

Mithilfe der Katalogsicht SYSCAT.COLUMNS können Sie Tabellendefinitionen anzeigen. Jede Zeile dieser Katalogsichten stellt eine für eine Tabelle, eine Sicht oder einen Kurznamen definierte Spalte dar. Zum Anzeigen der Daten in den Spalten können Sie die Anweisung SELECT verwenden.

Aliasnamen von Tabellen oder Sichten

Ein *Aliasname* ist ein alternativer Name für eine Tabelle oder eine Sicht. Er kann zum Verweisen auf eine Tabelle oder eine Sicht verwendet werden, sofern auf eine vorhandene Tabelle oder Sicht verwiesen werden *kann*.

Ein Aliasname darf nicht in allen Kontexten verwendet werden. Er darf beispielsweise nicht in der Prüfbedingung einer Prüfung auf Integritätsbedingung verwendet werden. Ein Aliasname darf nicht auf eine deklarierte temporäre Tabelle verweisen.

Ähnlich wie Tabellen oder Sichten kann auch ein Aliasname erstellt, gelöscht oder mit Kommentaren versehen werden. Im Gegensatz zu Tabellen können Aliasnamen jedoch in einem Prozess, der *Verketten* genannt wird, aufeinander verweisen. Bei Aliasnamen handelt es sich um Namen, auf die öffentlich verwiesen wird; daher ist für Ihre Verwendung keine Sonderberechtigung bzw. kein besonderes Zugriffsrecht erforderlich. Für den Zugriff auf die Tabelle oder Sicht, auf die durch einen Aliasnamen verwiesen wird, ist jedoch die Berechtigung erforderlich, die diesen Objekten zugeordnet ist.

Es gibt noch andere Aliasnamentypen, zum Beispiel Aliasnamen für Datenbanken und Netzwerke. Es können auch Aliasnamen für *Kurznamen* erstellt werden, die sich auf Datentabellen oder auf Sichten beziehen, die sich auf föderierten Systemen befinden.

Löschen von Tabellen

Eine Tabelle kann mit einer Anweisung DROP TABLE gelöscht werden. Beim Löschen einer Tabelle wird die Zeile in der Systemkatalogsicht SYSCAT.TABLES gelöscht, die die Informationen über die Tabelle enthält, während alle anderen von der Tabelle abhängigen Objekte auf verschiedene Weise davon betroffen sind.

Zum Beispiel:

- Alle Spaltennamen werden gelöscht.
- Die für Spalten der Tabelle erstellten Indizes werden gelöscht.
- Alle Sichten, die auf der Tabelle basieren, werden als funktionsunfähig markiert.
- Alle Zugriffsrechte auf die gelöschte Tabelle und abhängige Sichten werden implizit widerrufen.
- Alle referenziellen Integritätsbedingungen, in denen die Tabelle eine übergeordnete oder abhängige Tabelle ist, werden gelöscht.
- Alle Pakete und im Cache zwischengespeicherten dynamischen SQL- und XQuery-Anweisungen, die von der gelöschten Tabelle abhängig sind, werden als ungültig markiert und bleiben ungültig, bis die abhängigen Objekte neu erstellt

werden. Dazu gehören Pakete, die von einer zu löschenden übergeordneten Tabelle über der untergeordneten Tabelle in der Hierarchie abhängen.
- Verweisspalten, für die die gelöschte Tabelle als Bereich des Verweises definiert ist, haben keinen Bereich mehr.
- Die Definition eines Aliasnamens für die Tabelle wird nicht berührt, da ein Aliasname auch ohne Tabelle vorhanden sein kann.
- Alle von der gelöschten Tabelle abhängigen Trigger werden als unbrauchbar markiert.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um eine Tabelle zu löschen:

```
DROP TABLE <tabellenname>
```

Mit der folgenden Anweisung wird die Tabelle DEPARTMENT gelöscht:

```
DROP TABLE DEPARTMENT
```

Eine einzelne Tabelle kann nicht gelöscht werden, wenn sie eine untergeordnete Tabelle hat. Alle Tabellen, die einer Tabellenhierarchie angehören, können jedoch mit einer einzigen Anweisung DROP TABLE HIERARCHY gelöscht werden, wie im folgenden Beispiel gezeigt:

```
DROP TABLE HIERARCHY person
```

In der Anweisung DROP TABLE HIERARCHY muss die Stammtabelle der zu löschenden Hierarchie angegeben werden.

Zwischen dem Löschen einer Tabellenhierarchie und dem Löschen einer bestimmten Tabelle bestehen folgende Unterschiede:
- DROP TABLE HIERARCHY aktiviert keine Löschtrigger, wie sie von einzelnen Anweisungen DROP TABLE aktiviert würden. Beim Löschen einer einzelnen untergeordneten Tabelle würden zum Beispiel Löschtrigger für die dazugehörigen übergeordneten Tabellen aktiviert.
- DROP TABLE HIERARCHY erstellt keine Protokolleinträge für die einzelnen Zeilen der gelöschten Tabellen. Das Löschen der Hierarchie wird stattdessen als ein einziges Ereignis protokolliert.

Löschen von MQTs oder Zwischenspeichertabellen

Sie können eine MQT (Materialized Query Table, gespeicherte Abfragetabelle) oder Zwischenspeichertabelle nicht ändern, sondern nur löschen. Alle Indizes, Primärschlüssel, Fremdschlüssel und Prüfungen auf Integritätsbedingungen, die auf die Tabelle verweisen, werden gelöscht. Alle Sichten und Trigger, die auf die Tabelle verweisen, werden funktionsunfähig gemacht. Alle Pakete, die von einem gelöschten oder als funktionsunfähig markierten Objekt abhängen, werden ungültig gemacht.

Geben Sie in die Befehlszeile die folgende Anweisung ein, um eine MQT oder Zwischenspeichertabelle zu löschen:

```
DROP TABLE <tabellenname>
```

Mit der folgenden Anweisung wird die MQT 'XT' gelöscht:

```
DROP TABLE XT
```

Eine MQT kann explizit mit der Anweisung DROP TABLE oder implizit durch das Löschen einer der zugrunde liegenden Tabellen gelöscht werden.

Eine Zwischenspeichertabelle kann explizit mit der Anweisung DROP TABLE oder implizit durch das Löschen ihrer zugeordneten MQT gelöscht werden.

Szenarios und Beispiele für Tabellen

Dieser Abschnitt enthält Beschreibungen von Szenarios und Beispielen für Tabellen.

Szenarios: Optimistisches Sperren und zeitbasierte Erkennung

Drei Szenarios werden bereitgestellt, die zeigen, wie Sie das optimistische Sperren in Ihren Anwendungen mit und ohne zeitbasierte Erkennung sowie mit und ohne implizit verdeckte Spalten aktivieren und implementieren.

Szenario: Verwenden des optimistischen Sperrens in einem Anwendungsprogramm

Dieses Szenario veranschaulicht, wie das optimistische Sperren in einem Anwendungsprogramm implementiert wird. Es behandelt sechs verschiedene Szenarios.

Betrachten Sie die folgende Abfolge von Ereignissen in einer Anwendung, die zur Nutzung des optimistischen Sperrens entworfen und eingerichtet wurde:

```
SELECT QUANTITY, row change token FOR STOCK, RID_BIT(STOCK)
 INTO :h_quantity, :h_rct, :h_rid
 FROM STOCK WHERE PARTNUM = 3500
```

In diesem Szenario liest die Anwendungslogik jede Zeile. Da diese Anwendung wie in „Aktivieren des optimistischen Sperrens in Anwendungen" auf Seite 286 beschrieben für das optimistische Sperren aktiviert ist, enthält die SELECT-Liste den Wert der Funktion RID_BIT(), der in der Hostvariablen :h_rid gespeichert wird, und den Wert des Zeilenänderungstokens ('row change token'), der in der Hostvariablen :h_rct gespeichert wird.

Bei aktiviertem optimistischen Sperren nimmt die Anwendung optimistischerweise an, dass alle Zeilen, die Zielzeilen für UPDATE- oder DELETE-Operationen sind, unverändert bleiben, auch wenn sie nicht durch Sperren geschützt werden. Zur Verbesserung des gemeinsamen Zugriffs auf die Datenbank entfernt die Anwendung die Zeilensperre(n) durch eine der folgenden Methoden:

- Sie schreibt die UOW (Unit of Work, Arbeitseinheit) mit COMMIT fest, wobei die Zeilensperren entfernt werden.
- Sie schließt den Cursor mit der Klausel WITH RELEASE, wobei die Zeilensperren entfernt werden.
- Sie verwendet eine niedrigere Isolationsstufe:
  - CURSOR STABILITY (CS, Cursorstabilität), wobei die Zeile nicht gesperrt ist, nachdem der Cursor die nächste Zeile abgerufen hat bzw. bis das Ende der Ergebnistabelle erreicht ist.
  - UNCOMMITED READ (UR, nicht festgeschriebener Lesevorgang), wobei alle nicht festgeschriebenen Daten einen neuen (nicht festgeschriebenen) Wert für das Zeilenänderungstoken haben. Wenn die nicht festgeschriebenen Daten rückgängig gemacht werden (Rollback), erhält das alte, festgeschriebene Zeilenänderungstoken einen anderen Wert.

    Anmerkung: Unter der Annahme, dass Aktualisierungen im Normalfall nicht rückgängig gemacht werden, bietet die Isolationsstufe UR den meisten gemeinsamen Zugriff.

- Sie trennt die Verbindung zur Datenbank, sodass alle DB2-Serverressourcen für die Anwendung freigegeben werden. (.NET-Anwendungen verwenden diesen Modus häufig.)

Die Anwendung verarbeitet die Zeilen und entscheidet, eine von ihnen optimistisch zu aktualisieren:

```
UPDATE STOCK SET QUANTITY = QUANTITY – 1
 WHERE row change token FOR STOCK = :h_rct AND
 RID_BIT(STOCK) = :h_rid
```

Die Anweisung UPDATE aktualisiert die in der oben gezeigten SELECT-Anweisung identifizierte Zeile.

Das UPDATE-Vergleichselement mit Suche wird als Direktabruf aus der Tabelle eingeplant:

```
RID_BIT(STOCK) = :h_rid
```

Der Direktabruf ist ein sehr effizienter Zugriffsplan, dessen Aufwand vom DB2-Optimierungsprogramm problemlos kalkuliert werden kann. Wenn das Vergleichselement mit der Funktion RID_BIT() keine Zeile findet, wurde die Zeile gelöscht und die UPDATE-Operation schlägt wegen der nicht gefundenen Zeile fehl.

Vorausgesetzt, das Vergleichselement mit RID_BIT() findet eine Zeile, findet das Vergleichselement 'row change token FOR STOCK = :h_rct' die Zeile, sofern die Zeile nicht geändert wurde. Wenn sich das Zeilenänderungstoken seit Ausführung der Anweisung SELECT geändert hat, schlägt die UPDATE-Operation mit Suche aufgrund nicht gefundener Zeile fehl.

In Tabelle 50 sind die möglichen Szenarios aufgeführt, die bei aktiviertem optimistischem Sperren auftreten können.

Tabelle 50. Mögliche Szenarios bei aktiviertem optimistischem Sperren

Szenario-ID	Aktion	Ergebnis
Szenario 1	Es ist keine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) in der Tabelle definiert, und keine andere Anwendung hat die Zeile geändert.	Die UPDATE-Operation wird erfolgreich ausgeführt, da das Vergleichselement mit der Zeilenänderungszeitmarke für die durch ':h_rid' angegebene Zeile erfolgreich ausgewertet wird.
Szenario 2	In der Tabelle ist eine Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) definiert. Eine andere Anwendung aktualisiert die Zeile nach der SELECT-Operation und vor der UPDATE-Operation (und der COMMIT-Operation), wobei auch die Spalte für die Zeilenänderungszeitmarke aktualisiert wird.	Der Vergleich des Vergleichselements mit dem Zeilenänderungstoken zwischen dem Wert des Tokens, der aus der Zeitmarke in der Zeile zum Zeitpunkt der SELECT-Operation generiert wurde, und dem Zeitmarkenwert des Tokens, der aktuell in der Zeile enthalten ist, schlägt fehl. Infolgedessen kann die Anweisung UPDATE keine Zeile finden.

Tabelle 50. Mögliche Szenarios bei aktiviertem optimistischem Sperren (Forts.)

Szenario-ID	Aktion	Ergebnis
Szenario 3	In der Tabelle ist eine Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) definiert. Eine andere Anwendung aktualisiert die Zeile, sodass die Zeile ein neues Zeilenänderungstoken erhält. Diese Anwendung wählt die Zeile mit der Isolationsstufe UR (Uncommitted Read, nicht festgeschriebener Lesevorgang) aus und ruft das neue, nicht festgeschriebene Zeilenänderungstoken ab.	Diese Anwendung führt die UPDATE-Operation aus, die wartet, bis die andere Anwendung die Zeilensperre freigibt. Das Vergleichselement mit dem Zeilenänderungstoken wird erfolgreich ausgewertet, wenn die andere Anwendung die Änderung mit dem neuen Token festschreibt (COMMIT), sodass die UPDATE-Operation erfolgreich ausgeführt wird. Der Vergleich durch das Vergleichselement mit dem Zeilenänderungstoken schlägt fehl, wenn die andere Anwendung die Änderung der Zeile auf den vorherigen Stand mit dem alten Token rückgängig macht (Rollback), sodass die UPDATE-Operation keine Zeile finden kann.
Szenario 4	Es ist keine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) in der Tabelle definiert. Eine andere Zeile wird auf derselben Seite nach der SELECT-Operation, jedoch vor der UPDATE-Operation aktualisiert, gelöscht oder eingefügt.	Der Vergleich durch das Vergleichselement mit dem Zeilenänderungstoken schlägt fehl, weil sich der Wert des Zeilenänderungstokens für alle Zeile auf der Seite geändert hat, sodass die UPDATE-Anweisung keine Zeile finden kann, selbst wenn die fragliche Zeile selbst in Wirklichkeit nicht geändert wurde. Dieses Szenario eines falschen Negativwerts würde nicht zu einem UPDATE-Fehler führen, wenn eine Spalte für die Zeilenänderungszeitmarke hinzugefügt worden wäre.
Szenario 5	Die Tabelle wurde geändert, sodass sie eine Spalte für die Zeilenänderungszeitmarke enthält, und die Zeile, die von der Anweisung SELECT zurückgegeben wurde, wurde seit dem Zeitpunkt der Anweisung ALTER TABLE nicht geändert. Eine andere Anwendung aktualisiert die Zeile, indem sie die Spalte für die Zeilenänderungszeitmarke dieser Zeile im Prozess mit der aktuellen Zeitmarke hinzufügt.	Der Vergleich durch das Vergleichselement mit dem Zeilenänderungstoken zwischen dem zuvor generierten Token und dem Tokenwert aus der Spalte der Zeilenänderungszeitmarke schlägt fehl, sodass die Anweisung UPDATE keine Zeile finden kann. Da die fragliche Zeile tatsächlich geändert wurde, handelt es sich in diesem Fall nicht um ein Szenario mit falschem Negativwert.

Tabelle 50. Mögliche Szenarios bei aktiviertem optimistischem Sperren (Forts.)

Szenario-ID	Aktion	Ergebnis
Szenario 6	Die Tabelle wird nach der SELECT-Operation und vor der UPDATE-Operation reorganisiert. Die Satz-ID (RID), die durch ':h_rid' angegeben wird, findet keine Zeile bzw. enthält eine Zeile mit einem anderen Token, sodass die Aktualisierung fehlschlägt. Dies ist die Form eines falschen Negativwerts, die sich nicht vermeiden lässt, selbst wenn eine Spalte für die Zeilenänderungszeitmarke in der Zeile vorhanden ist.	Die Zeile selbst wird durch die Reorganisation nicht aktualisiert. Der RID_BIT-Teil des Vergleichselements kann jedoch die ursprüngliche Zeile nach der Reorganisation nicht mehr identifizieren.

Szenarios: Optimistisches Sperren mit implizit verdeckten Spalten

Die folgenden Szenarios veranschaulichen, wie das optimistische Sperren in einem Anwendungsprogramm unter Verwendung implizit verdeckter Spalten, das heißt, mit Spalten, die mit dem Attribut IMPLICITLY HIDDEN definiert sind, implementiert wird.

Nehmen Sie für diese Szenarios an, dass die Tabelle SALARY_INFO mit drei Spalten definiert ist. Die erste Spalte ist eine implizit verdeckte Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP), deren Werte immer generiert werden.

Szenario 1:

In der folgenden Anweisung wird die implizit verdeckte Spalte in der Spaltenliste explizit aufgeführt und ein Wert für sie in der Klausel VALUES vorgesehen:

```
INSERT INTO SALARY_INFO (UPDATE_TIME, LEVEL, SALARY)
  VALUES (DEFAULT, 2, 30000)
```

Szenario 2:

Die folgende Anweisung INSERT enthält eine implizite Spaltenliste. Eine implizite Spaltenliste umfasst keine implizit verdeckten Spalten. Daher enthält die Klausel VALUES nur Werte für die beiden anderen Spalten:

```
INSERT INTO SALARY_INFO
  VALUES (2, 30000)
```

In diesem Fall muss die Spalte UPDATE_TIME so definiert sein, dass sie einen Standardwert besitzt. Dieser Standardwert für die Zeile verwendet, die eingefügt wird.

Szenario 3:

In der folgenden Anweisung wird die implizit verdeckte Spalte explizit in der SELECT-Liste aufgeführt, und in der Ergebnismenge wird ein Wert für sie angezeigt:

```
SELECT UPDATE_TIME, LEVEL, SALARY FROM SALARY_INFO
  WHERE LEVEL = 2

  UPDATE_TIME                   LEVEL    SALARY
  --------------------------- ----------- -----------
  2006-11-28-10.43.27.560841      2      30000
```

Szenario 4:

In der folgenden Anweisung wird die Spaltenliste implizit durch die Notation '*' generiert, sodass die implizit verdeckte Spalte in der Ergebnismenge nicht auftritt:

```
SELECT * FROM SALARY_INFO
  WHERE LEVEL = 2

  LEVEL       SALARY
  ----------- -----------
       2      30000
```

Szenario 5:

In der folgenden Anweisung wird die Spaltenliste implizit durch die Verwendung der Notation '*' generiert. Der Wert der implizit verdeckten Spalte wird durch die Verwendung des Ausdrucks ROW CHANGE TIMESTAMP FOR ebenfalls zurückgegeben:

```
SELECT ROW CHANGE TIMESTAMP FOR SALARY_INFO AS ROW_CHANGE_STAMP, SALARY_INFO.*
FROM SALARY_INFO WHERE LEVEL = 2
```

Die Ergebnistabelle ist der in Szenario 3 ähnlich (die Spalte UPDATE_TIME wird als ROW_CHANGE_STAMP zurückgegeben).

Szenario: Zeitbasierte Aktualisierungserkennung

Dieses Szenario veranschaulicht, wie das optimistische Sperren in einem Anwendungsprogramm unter Verwendung der Aktualisierungserkennung mithilfe einer Zeitmarke implementiert wird. Es behandelt drei verschiedene Szenarios.

In diesem Szenario wählt die Anwendung alle Zeilen aus, die in den letzten 30 Tagen geändert wurden.

```
SELECT * FROM TAB WHERE
  ROW CHANGE TIMESTAMP FOR TAB <=
  CURRENT TIMESTAMP AND
  ROW CHANGE TIMESTAMP FOR TAB >=
  CURRENT TIMESTAMP - 30 days;
```

Szenario 1:

Es ist keine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) in der Tabelle definiert. Die Ausführung der Anweisung schlägt mit SQL20431N fehl. Dieser SQL-Ausdruck wird nur für Tabellen mit einer definierten Spalte für die Zeilenänderungszeitmarke unterstützt.

Anmerkung: Dieses Szenario funktioniert unter z/OS.

Szenario 2:

Eine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) wurde beim Erstellen der Tabelle definiert:

```
CREATE TABLE TAB ( ..., RCT TIMESTAMP NOT NULL
                        GENERATED ALWAYS
                        FOR EACH ROW ON UPDATE AS
                        ROW CHANGE TIMESTAMP)
```

Diese Anweisung gibt alle Zeilen zurück, die in den letzten 30 Tagen eingefügt oder aktualisiert wurden.

Szenario 3:

Eine Spalte für die Zeilenänderungszeitmarke wurde der Tabelle zu einem Zeitpunkt während der letzten 30 Tage mithilfe der Anweisung ALTER TABLE hinzugefügt:

```
ALTER TABLE TAB ADD COLUMN RCT TIMESTAMP NOT NULL
                               GENERATED ALWAYS
                               FOR EACH ROW ON UPDATE AS
                               ROW CHANGE TIMESTAMP
```

Diese Anweisung gibt alle Zeilen in der Tabelle zurück. Alle Zeilen, die seit Ausführung der Anweisung ALTER TABLE nicht geändert wurden, verwenden als Standardwert die Zeitmarke der Anweisung ALTER TABLE selbst, während alle anderen Zeilen, die seit dem geändert wurden, eine eindeutige Zeitmarke haben.

Kapitel 12. Integritätsbedingungen

In jeder Geschäftsumgebung müssen Daten häufig bestimmten Rahmenbedingungen oder Regeln genügen. Zum Beispiel muss eine Personalnummer eindeutig sein. Der Datenbankmanager stellt *Integritätsbedingungen* bereit, die zur Umsetzung solcher Regeln verwendet werden können.

Die folgenden Typen von Integritätsbedingungen sind verfügbar:

- NOT NULL, Integritätsbedingung
- Eindeutige Integritätsbedingungen (oder eindeutige Integritätsbedingungen über Schlüssel)
- Integritätsbedingungen über Primärschlüssel
- Integritätsbedingungen über Fremdschlüssel (oder referenzielle Integritätsbedingungen)
- Prüfungen auf Integritätsbedingungen (in Tabellen)
- Informative Integritätsbedingungen

Integritätsbedingungen werden nur Tabellen zugeordnet; sie werden entweder im Rahmen des Tabellenerstellungsprozesses (mithilfe der Anweisung CREATE TABLE) erstellt, oder sie werden einer Tabellendefinition nach der Erstellung der Tabelle (mithilfe der Anweisung ALTER TABLE) hinzugefügt. Mithilfe der Anweisung ALTER TABLE können Sie Integritätsbedingungen ändern. In den meisten Fällen, können vorhandene Integritätsbedingungen jederzeit gelöscht werden; diese Aktion beeinflusst die Struktur der Tabelle oder die darin gespeicherten Daten nicht.

Anmerkung: Eindeutige Integritätsbedingungen über Schlüssel und Integritätsbedingungen über Primärschlüssel werden nur Tabellenobjekten zugeordnet; sie werden häufig durch die Verwendung eines oder mehrerer Indizes für den eindeutigen Schlüssel oder den Primärschlüssel erzwungen.

Typen von Integritätsbedingungen

Eine *Integritätsbedingung* ist eine Regel, die zu Optimierungszwecken verwendet wird.

Es gibt fünf Typen von Integritätsbedingungen:

- Bei der *Integritätsbedingung NOT NULL* handelt es sich um eine Regel, die vermeidet, dass in eine oder mehrere Spalten innerhalb einer Tabelle Nullwerte eingegeben werden.
- Eine *eindeutige Integritätsbedingung* (auch als *Integritätsbedingung über eindeutige Schlüssel* bezeichnet) ist eine Regel, die untersagt, dass in einer oder mehreren Spalten innerhalb einer Tabelle doppelte Werte auftreten. Eindeutige Integritätsbedingungen werden in Form von eindeutigen Schlüsseln und Primärschlüsseln unterstützt. Zum Beispiel kann eine eindeutige Integritätsbedingung für die Lieferantenkennung einer Lieferantentabelle definiert werden, um sicherzustellen, dass nicht ein und dieselbe Kennung zwei Lieferanten zugewiesen wird.
- Eine *Integritätsbedingung über Primärschlüssel* ist eine Spalte oder eine Kombination von Spalten, die die gleichen Merkmale wie eine eindeutige Integritätsbedingung besitzt. Ein Primärschlüssel und Integritätsbedingungen über Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

- Eine *Integritätsbedingung über Fremdschlüssel* (auch als *referenzielle Integritätsbedingung* bezeichnet) ist eine logische Regel über Werte in einer oder mehreren Spalten in mindestens einer Tabelle. Zum Beispiel enthält eine Gruppe von Tabellen gemeinsame Informationen über die Lieferanten einer Firma. Gelegentlich ändert sich der Name eines Lieferanten. Sie können eine referenzielle Integritätsbedingung definieren, die besagt, dass die Kennung (ID) des Lieferanten in einer Tabelle mit einer Lieferanten-ID in den Lieferanteninformationen übereinstimmen muss. Diese Integritätsbedingung verhindert Einfüge-, Aktualisierungs- oder Löschoperationen, die ansonsten zu fehlenden Lieferanteninformationen führen würden.
- Eine *Prüfung auf Integritätsbedingung (in Tabellen)* (oder auch einfach *Prüfung auf Integritätsbedingung*) definiert Einschränkungen für Daten, die einer bestimmten Tabelle hinzugefügt werden. Zum Beispiel kann eine Prüfung auf Integritätsbedingung in einer Tabelle gewährleisten, dass das Gehaltsniveau eines Mitarbeiters stets mindestens $20.000 beträgt, wenn Gehaltsdaten in einer Tabelle mit Personalinformationen hinzugefügt oder aktualisiert werden.

Eine *informative Integritätsbedingung* ist ein Attribut eines bestimmten Typs von Integritätsbedingung, das nicht vom Datenbankmanager umgesetzt wird.

NOT NULL, Integritätsbedingung

Die Integritätsbedingung NOT NULL verhindert, dass Nullwerte in eine Spalte eingegeben werden.

Der Nullwert dient in Datenbanken zur Darstellung eines unbekannten Status. Standardmäßig unterstützen alle integrierten Datentypen, die im Datenbankmanager verfügbar sind, das Vorhandensein von Nullwerten. Einige Geschäftsregeln verlangen jedoch möglicherweise, dass in jedem Fall ein Wert angegeben werden muss (z. B. muss jeder Mitarbeiter Kontaktinformationen für den Notfall angeben). Die Integritätsbedingung NOT NULL dient zur Sicherstellung, dass einer bestimmten Spalte einer Tabelle nie der Nullwert zugeordnet wird. Wenn die Integritätsbedingung NOT NULL für eine bestimmte Spalte definiert wurde, schlägt jede Einfüge- oder Aktualisierungsoperation fehl, die versucht, in dieser Spalte einen Nullwert zu speichern.

Da Integritätsbedingungen nur für eine bestimmte Tabelle gelten, werden sie in der Regel zusammen mit den Attributen einer Tabelle während des Tabellenerstellungsprozesses definiert. Die folgende Anweisung CREATE TABLE zeigt, wie die Integritätsbedingung NOT NULL für eine bestimmte Spalte definiert wird:

```
CREATE TABLE EMPLOYEES    (. . .
                           EMERGENCY_PHONE    CHAR(14)  NOT NULL,
                           . . .
                           );
```

Eindeutige Integritätsbedingungen

Eindeutige Integritätsbedingungen stellen sicher, dass die Werte in einer Gruppe von Spalten für alle Zeilen in der Tabelle eindeutig und nicht null sind. Die Spalten, die in einer eindeutigen Integritätsbedingung angegeben werden, müssen mit NOT NULL (keine Nullwerte möglich) definiert werden. Der Datenbankmanager stellt die Eindeutigkeit des Schlüssels bei Änderungen an den Spalten der eindeutigen Integritätsbedingung mithilfe eines eindeutigen Index sicher.

Eindeutige Integritätsbedingungen können in der Anweisung CREATE TABLE oder ALTER TABLE durch die Klausel UNIQUE definiert werden. Zum Beispiel könnte eine typische eindeutige Integritätsbedingung in einer Tabelle DEPARTMENT mit Daten über Abteilungen darin bestehen, dass die Abteilungsnummer (DEPTNO) eindeutig und nicht null ist.

In Abb. 21 sehen Sie, dass ein doppelter Datensatz nicht zu einer Tabelle hinzugefügt wird, wenn für die Tabelle eine eindeutige Integritätsbedingung vorhanden ist:

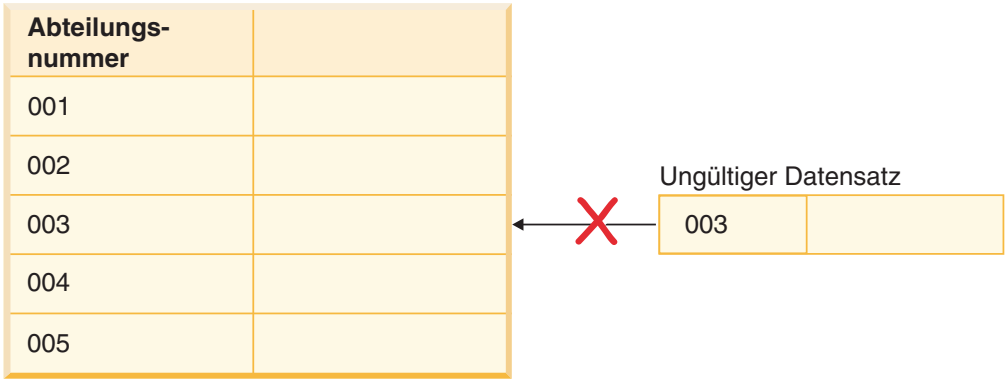


Abbildung 21. Eindeutige Integritätsbedingungen verhindern Datenduplikate

Der Datenbankmanager beachtet die Integritätsbedingung bei Operationen zum Einfügen und Aktualisieren von Daten, um die Datenintegrität zu gewährleisten.

Eine Tabelle kann eine beliebige Anzahl von Integritätsbedingungen für Eindeutigkeit haben, wobei maximal eine eindeutige Integritätsbedingung als Primärschlüssel definiert sein kann. Eine Tabelle kann nicht mehr als eine eindeutige Integritätsbedingung für dieselbe Gruppe von Spalten haben.

Eine eindeutige Integritätsbedingung, auf die im Fremdschlüssel einer referenziellen Integritätsbedingung verwiesen wird, heißt *übergeordneter Schlüssel*.

- Wenn eine eindeutige Integritätsbedingung in einer Anweisung CREATE TABLE definiert wird, wird vom Datenbankmanager automatisch ein eindeutiger Index erstellt und als systemerforderlicher Primärindex bzw. eindeutiger Index gekennzeichnet.
- Wenn eine eindeutige Integritätsbedingung in einer Anweisung ALTER TABLE definiert wird und ein Index für die gleichen Spalten vorhanden ist, wird dieser vorhandene Index als eindeutig und systemerforderlich gekennzeichnet. Falls kein solcher Index vorhanden ist, wird vom Datenbankmanager automatisch ein eindeutiger Index erstellt und als systemerforderlicher Primärindex bzw. eindeutiger Index gekennzeichnet.

Anmerkung: Es besteht ein Unterschied zwischen dem Definieren einer eindeutigen Integritätsbedingung und dem Erstellen eines eindeutigen Index. Zwar dienen beide zur Erhaltung der Eindeutigkeit, jedoch lässt ein eindeutiger Index Spalten mit möglichen Nullwerten zu und kann in der Regel nicht als übergeordneter Schlüssel verwendet werden.

Integritätsbedingungen über Primärschlüssel

Integritätsbedingungen über Primärschlüssel und Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

Ein Primärschlüssel ist eine Spalte oder eine Kombination von Spalten, die die gleichen Merkmale wie eine eindeutige Integritätsbedingung besitzen. Da ein Primärschlüssel zur Angabe einer Zeile in einer Tabelle verwendet wird, muss er eindeutig sein und über das Attribut NOT NULL verfügen. Eine Tabelle kann nicht mehr als einen Primärschlüssel, jedoch mehrere eindeutige Schlüssel besitzen. Primärschlüssel sind wahlfrei und können definiert werden, wenn eine Tabelle erstellt oder geändert wird. Sie besitzen zudem den weiteren Vorteil, dass sie für eine Reihenfolge der Daten sorgen, wenn Daten exportiert oder reorganisiert werden.

Prüfungen auf Integritätsbedingungen (in Tabellen)

Eine *Prüfung auf Integritätsbedingung* (auch als *Prüfung auf Integritätsbedingung in Tabellen* bezeichnet) ist eine Datenbankregel, mit der die zulässigen Werte in einer oder mehreren Spalten jeder Zeile einer Tabelle angegeben werden. Die Angabe von Prüfungen auf Integritätsbedingungen erfolgt durch ein eingeschränktes Format einer Suchbedingung.

Integritätsbedingungen über Fremdschlüssel (referenzielle Integritätsbedingungen)

Integritätsbedingungen über Fremdschlüssel (die auch als *referenzielle Integritätsbedingungen* bezeichnet werden) geben Ihnen die Möglichkeit, erforderliche Beziehungen zwischen und innerhalb von Tabellen zu definieren.

Zum Beispiel könnte eine typische Integritätsbedingung über Fremdschlüssel festlegen, dass jeder Mitarbeiter in der Tabelle EMPLOYEE ein Mitglied einer bestehenden, in der Tabelle DEPARTMENT definierten Abteilung sein muss.

Als *referenzielle Integrität* wird der Status einer Datenbank bezeichnet, in der alle Werte aller Fremdschlüssel gültig sind. Ein *Fremdschlüssel* ist eine Spalte oder eine Gruppe von Spalten in einer Tabelle, deren Werte mit mindestens einem Wert des Primärschlüssels oder eines eindeutigen Schlüssels einer Zeile in der übergeordneten Tabelle übereinstimmen müssen. Eine *referenzielle Integritätsbedingung* ist die Regel, dass die Werte des Fremdschlüssels nur dann gültig sind, wenn eine der folgenden Bedingungen gilt:

- Sie treten als Werte eines übergeordneten Schlüssels auf.
- Eine Komponente des Fremdschlüssels ist NULL.

Zur Herstellung dieser Beziehung würden Sie die Abteilungsnummer (WORKDEPT) der Tabelle EMPLOYEE als Fremdschlüssel und die Abteilungsnummer (DEPTNO) der Tabelle DEPARTMENT als Primärschlüssel definieren.

In Abb. 22 auf Seite 309 sehen Sie, wie ein Datensatz mit einem ungültigen Schlüssel nicht zu einer Tabelle hinzugefügt wird, wenn zwischen zwei Tabellen eine Integritätsbedingung über Fremdschlüssel vorhanden ist:

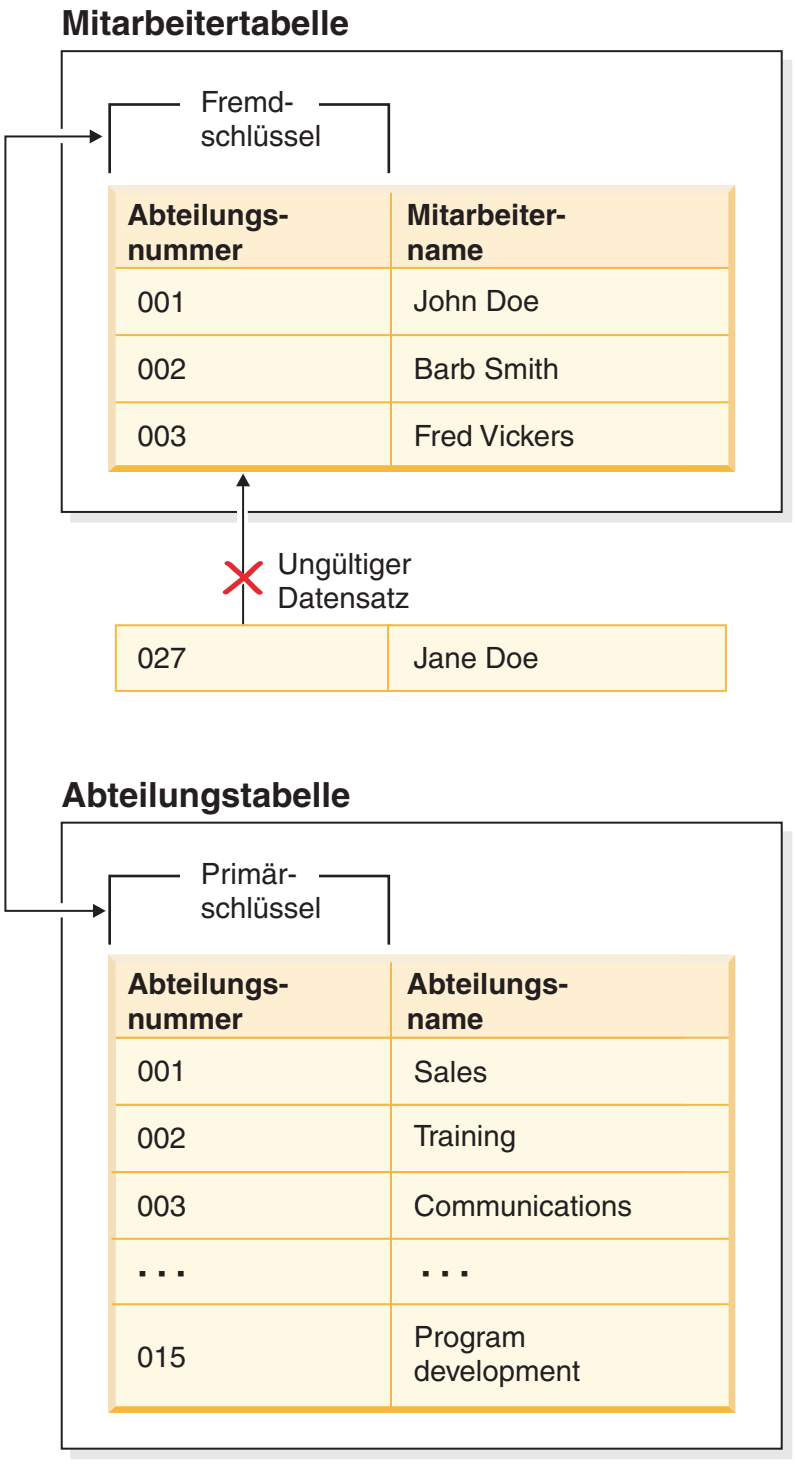


Abbildung 22. Integritätsbedingungen über Fremd- und Primärschlüssel

Die Tabelle, die den übergeordneten Schlüssel enthält, wird als *übergeordnete Tabelle* der referenziellen Integritätsbedingung bezeichnet, während die Tabelle, die den Fremdschlüssel enthält, als *abhängige Tabelle* dieser Tabelle bezeichnet wird.

Referenzielle Integritätsbedingungen können in der Anweisung CREATE TABLE oder ALTER TABLE definiert werden. Referenzielle Integritätsbedingungen werden

durch den Datenbankmanager bei der Ausführung der Anweisungen INSERT, UPDATE, DELETE, ALTER TABLE, MERGE, ADD CONSTRAINT und SET INTEGRITY umgesetzt.

Die Regeln der referenziellen Integrität lassen sich durch die folgenden Termini definieren:

Tabelle 51. Termini zur referenziellen Integrität

Konzept	**Termini**
Übergeordneter Schlüssel	Ein Primärschlüssel bzw. ein eindeutiger Schlüssel einer referenziellen Integritätsbedingung.
Übergeordnete Zeile	Eine Zeile, die mindestens eine von ihr abhängige Zeile hat.
Übergeordnete Tabelle	Eine Tabelle, die den übergeordneten Schlüssel einer referenziellen Integritätsbedingung enthält. Eine Tabelle kann für beliebig viele referenzielle Integritätsbedingungen die übergeordnete Tabelle sein. Eine Tabelle, die in einer referenziellen Integritätsbedingung übergeordnete Tabelle ist, kann gleichzeitig abhängige Tabelle in einer referenziellen Integritätsbedingung sein.
Abhängige Tabelle	Eine Tabelle, die in ihrer Definition mindestens eine referenzielle Integritätsbedingung enthält. Eine Tabelle kann für beliebig viele referenzielle Integritätsbedingungen eine abhängige Tabelle sein. Eine Tabelle, die in einer referenziellen Integritätsbedingung eine abhängige Tabelle ist, kann gleichzeitig die übergeordnete Tabelle in einer referenziellen Integritätsbedingung sein.
Untergeordnete Tabelle	Eine Tabelle ist eine untergeordnete Tabelle von Tabelle T, wenn sie eine abhängige Tabelle von T oder eine untergeordnete Tabelle einer abhängigen Tabelle von T ist.
Abhängige Zeile	Eine Zeile, die mindestens eine übergeordnete Zeile hat.
Untergeordnete Zeile	Eine Zeile ist eine untergeordnete Zeile von Zeile Z, wenn sie eine abhängige Zeile von Z oder eine untergeordnete Zeile einer abhängigen Zeile von Z ist.
Referenzieller Zyklus	Eine Gruppe referenzieller Integritätsbedingungen, in der jede Tabelle in der Gruppe eine untergeordnete Tabelle zu sich selbst ist.
Auf sich selbst verweisende Tabelle	Eine Tabelle, die in der gleichen referenziellen Integritätsbedingung sowohl übergeordnete als auch abhängige Tabelle ist. Die entsprechende Integritätsbedingung wird als *auf sich selbst verweisende Integritätsbedingung* bezeichnet.
Auf sich selbst verweisende Zeile	Eine Zeile, die eine übergeordnete Zeile zu sich selbst ist.

Der Zweck einer referenziellen Integritätsbedingung ist, zu garantieren, dass Tabellenbeziehungen gepflegt und Dateneingaberegeln befolgt werden. Dies bedeutet Folgendes: Solange eine referenzielle Integritätsbedingung gültig ist, garantiert der Datenbankmanager, dass für jede Zeile in einer untergeordneten Tabelle, die über einen Wert ungleich null in ihren Fremdschlüsselspalten verfügt, eine Zeile in einer entsprechenden übergeordneten Tabelle vorhanden ist, die über einen übereinstimmenden Wert in ihrem Primärschlüssel verfügt.

Wenn bei einer SQL-Operation versucht wird, Daten so zu ändern, dass die referenzielle Integrität beeinträchtigt wird, könnte gegen eine Integritätsbedingung über Fremdschlüssel (referenzielle Integritätsbedingung) verstoßen werden. Der Datenbankmanager meistert derartige Situationen, indem er eine Gruppe von

Regeln umsetzt, die den einzelnen referenziellen Integritätsbedingungen zugeordnet sind. Diese Gruppe von Regeln besteht aus folgenden Regeln:

- Einer Einfügeregel
- Einer Aktualisierungsregel
- Einer Löschregel

Wenn bei einer SQL-Operation versucht wird, Daten so zu ändern, dass die referenzielle Integrität beeinträchtigt wird, könnte gegen eine referenzielle Integritätsbedingung verstoßen werden. Beispiel:

- Eine Einfügeoperation könnte versuchen, einer untergeordneten Tabelle, die in ihren Fremdschlüsselspalten über einen Wert verfügt, der keinem Wert im Primärschlüssel der entsprechenden übergeordneten Tabelle entspricht, eine Zeile mit Daten hinzuzufügen.
- Eine Aktualisierungsoperation könnte versuchen, den Wert in den Fremdschlüsselspalten einer untergeordneten Tabelle in einen Wert zu ändern, der über keinen übereinstimmenden Wert im Primärschlüssel der entsprechenden übergeordneten Tabelle verfügt.
- Eine Aktualisierungsoperation könnte versuchen, den Wert im Primärschlüssel einer übergeordneten Tabelle in einen Wert zu ändern, der über keinen übereinstimmenden Wert in den Fremdschlüsselspalten einer untergeordneten Tabelle verfügt.
- Eine Löschoperation könnte versuchen, einen Datensatz aus einer übergeordneten Tabelle, die über einen übereinstimmenden Wert in den Fremdschlüsseltabellen einer untergeordneten Tabelle verfügt, zu entfernen.

Der Datenbankmanager meistert derartige Situationen, indem er eine Gruppe von Regeln umsetzt, die den einzelnen referenziellen Integritätsbedingungen zugeordnet sind. Diese Gruppe von Regeln besteht aus folgenden Regeln:

- Einer Einfügeregel
- Einer Aktualisierungsregel
- Einer Löschregel

Einfügeregel

Die Einfügeregel einer referenziellen Integritätsbedingung legt fest, dass ein einzufügender Nicht-Nullwert des Fremdschlüssels mit einem Wert des übergeordneten Schlüssels der übergeordneten Tabelle übereinstimmen muss. Der Wert eines zusammengesetzten Fremdschlüssels ist NULL, wenn eine Komponente des Werts NULL ist. Diese Regel gilt implizit, wenn ein Fremdschlüssel definiert wird.

Aktualisierungsregel

Die Aktualisierungsregel einer referenziellen Integritätsbedingung wird bei der Definition der referenziellen Integritätsbedingung angegeben. Angegeben werden kann NO ACTION oder RESTRICT. Die Aktualisierungsregel wird angewendet, wenn eine Zeile der übergeordneten Tabelle oder eine Zeile der abhängigen Tabelle aktualisiert wird.

Für eine übergeordnete Zeile gelten folgende Regeln, wenn ein Wert in einer Spalte des übergeordneten Schlüssels aktualisiert wird:

- Wenn irgendeine Zeile in der abhängigen Tabelle mit dem Ursprungswert des Schlüssels übereinstimmt, wird die UPDATE-Operation zurückgewiesen, wenn als Aktualisierungsregel RESTRICT definiert ist.

- Wenn irgendeine Zeile der abhängigen Tabelle nach Ausführung der UPDATE-Anweisung (ausgenommen AFTER-Trigger) keinen entsprechenden übergeordneten Schlüssel hat, wird die Aktualisierung zurückgewiesen, wenn als Aktualisierungsregel NO ACTION definiert ist.

Der Wert der übergeordneten eindeutigen Schlüssel kann nicht geändert werden, wenn es sich bei der Aktualisierungsregel um RESTRICT handelt und mindestens eine abhängige Zeile vorhanden ist. Handelt es sich jedoch um die Aktualisierungsregel NO ACTION, können übergeordnete eindeutige Schlüssel aktualisiert werden, so lange wie jedes untergeordnete Element über einen übergeordneten Schlüssel verfügt, wenn die Aktualisierungsanweisung beendet wird. Ein Aktualisierungswert eines Fremdschlüssels, der nicht NULL ist, muss mit einem Wert des Primärschlüssels aus der übergeordneten Tabelle der Beziehung identisch sein.

Mit der Verwendung von NO ACTION oder RESTRICT als Aktualisierungsregeln für referenzielle Integritätsbedingungen wird darüber hinaus festgestellt, wann die Integritätsbedingung erzwungen wird. Eine Aktualisierungsregel RESTRICT wird vor allen anderen Integritätsbedingungen, einschließlich der referenziellen Integritätsbedingungen mit Modifizierungsregeln, wie z. B. CASCADE oder SET NULL, erzwungen. Eine Aktualisierungsregel NO ACTION wird nach anderen referenziellen Integritätsbedingungen erzwungen. Beachten Sie, dass der zurückgegebene SQLSTATE-Wert sich unterscheidet, je nachdem ob die Aktualisierungsregel RESTRICT oder NO ACTION lautet.

Für eine abhängige Zeile gilt die Aktualisierungsregel NO ACTION implizit, wenn ein Fremdschlüssel angegeben wird. NO ACTION bedeutet, dass ein Aktualisierungswert eines Fremdschlüssels, der nicht NULL ist, mit einem Wert des übergeordneten Schlüssels der übergeordneten Tabelle übereinstimmen muss, wenn die UPDATE-Anweisung ausgeführt ist.

Der Wert eines zusammengesetzten Fremdschlüssels ist NULL, wenn eine Komponente des Werts NULL ist.

Löschregel

Die Löschregel einer referenziellen Integritätsbedingung wird bei der Definition der referenziellen Integritätsbedingung angegeben. Angegeben werden kann NO ACTION, RESTRICT, CASCADE oder SET NULL. SET NULL kann nur angegeben werden, wenn eine Spalte des Fremdschlüssels Nullwerte zulässt.

Wenn die angegebene Tabelle oder die Basistabelle der angegebenen Sicht eine übergeordnete Tabelle ist, dürfen die zum Löschen ausgewählten Zeilen keine abhängigen Zeilen in einer Beziehung mit einer Löschregel RESTRICT aufweisen und DELETE darf nicht an die untergeordneten Zeilen weitergegeben werden, die abhängige Zeilen in einer Beziehung mit einer Löschregel RESTRICT aufweisen.

Wenn die Löschoperation nicht von einer Löschregel RESTRICT verhindert wird, werden die ausgewählten Zeilen gelöscht. Jede Zeile, die von den ausgewählten Zeilen abhängig ist, ist ebenso betroffen:

- Die Spalten, die Nullwerte enthalten dürfen, der Fremdschlüssel von beliebigen Zeilen, die ihre abhängigen Zeilen in einer Beziehung mit einer Löschregel SET NULL darstellen, werden auf den Nullwert gesetzt.

- Beliebige Zeilen, die ihre abhängigen Zeilen in einer Beziehung mit einer Löschregel CASCADE darstellen, werden ebenso gelöscht, und die obigen Regeln gelten wiederum für diese Zeilen.

Die Löschregel NO ACTION wurde aktiviert, um zu erzwingen, dass jeder Fremdschlüssel, der einen Wert ungleich null aufweist, auf eine vorhandene übergeordnete Zeile verweist, nachdem die anderen referenziellen Integritätsbedingungen erzwungen wurden.

Die Löschregel einer referenziellen Integritätsbedingung wird nur angewendet, wenn *eine Zeile* der übergeordneten Tabelle gelöscht wird. Genauer gesagt, die Regel wird nur angewendet, wenn *eine Zeile* der übergeordneten Tabelle das Objekt einer Löschoperation bzw. einer weitergegebenen Löschoperation (weiter unten definiert) ist und die Zeile abhängige Zeilen in der abhängigen Tabelle der referenziellen Integritätsbedingung hat. Betrachten Sie ein Beispiel, in dem P die übergeordnete Tabelle, D die abhängige Tabelle und p die übergeordnete Zeile sind, die das Objekt einer Löschoperation bzw. einer weitergegebenen Löschoperation ist. Die Löschregel funktioniert wie folgt:

- Bei RESTRICT oder NO ACTION tritt ein Fehler auf und keine Zeilen werden gelöscht.
- Bei CASCADE wird die Löschoperation an die abhängigen Zeilen von p in Tabelle D weitergegeben.
- Bei SET NULL wird jede Spalte mit optionaler Dateneingabe (NULLABLE) des Fremdschlüssels jeder abhängigen Zeile von p in Tabelle D auf den Wert NULL gesetzt.

Jede Tabelle, die an einer Löschoperation von Tabelle P beteiligt sein kann, wird als durch *übergreifendes Löschen* mit P verbunden bezeichnet. Das heißt, eine Tabelle ist mit Tabelle P durch übergreifendes Löschen verbunden, wenn sie eine abhängige Tabelle von P oder einer Tabelle ist, an die Löschoperationen von P weitergegeben werden.

Die folgenden Einschränkungen gelten für die Abhängigkeiten zwischen Tabellen, die durch übergreifendes Löschen miteinander verbunden sind:

- Wenn eine Tabelle durch übergreifendes Löschen in einem referenziellen Zyklus von mehr als einer Tabelle mit sich selbst verbunden ist, darf der Zyklus keine Löschregel RESTRICT oder SET NULL enthalten.
- Eine Tabelle darf nicht gleichzeitig eine abhängige Tabelle in einer CASCADE-Abhängigkeit (auf sich selbst oder auf eine andere Tabelle verweisend) sein und eine auf sich selbst verweisende Beziehung mit einer Löschregel RESTRICT oder SET NULL haben.
- Wenn eine Tabelle durch übergreifendes Löschen mit einer anderen Tabelle über mehrere Abhängigkeiten verbunden ist, wobei diese Abhängigkeiten überlappende Fremdschlüssel haben, müssen diese Abhängigkeiten die gleiche Löschregel haben und keine dieser Löschregeln kann SET NULL sein.
- Wenn eine Tabelle durch übergreifendes Löschen mit einer anderen Tabelle über mehrere Abhängigkeiten verbunden ist, wobei eine der Beziehungen mit der Löschregel SET NULL angegeben ist, darf die Fremdschlüsseldefinition dieser Abhängigkeit keine Verteilungsschlüssel- oder MDC-Schlüsselspalte enthalten oder eine Datenpartitionierungsschlüsselspalte oder RCT-Schlüsselspalte hinzufügen.
- Wenn zwei Tabellen durch übergreifendes Löschen mit derselben Tabelle über CASCADE-Abhängigkeiten verbunden sind, dürfen die beiden Tabellen nicht durch ein übergreifendes Löschen miteinander verbunden werden, bei dem die

Pfade der durch übergreifendes Löschen definierten Abhängigkeiten mit der Löschregel RESTRICT oder SET NULL enden.

Informative Integritätsbedingungen

Eine *informative Integritätsbedingung* ist ein Integritätsbedingungsattribut, das vom SQL-Compiler zur Verbesserung des Zugriffs auf Daten verwendet werden kann. Informative Integritätsbedingungen werden nicht durch den Datenbankmanager umgesetzt und dienen nicht zur weiteren Überprüfung von Daten, sondern zur Verbesserung der Abfrageleistung.

Informative Integritätsbedingungen werden mithilfe der Anweisungen CREATE TABLE oder ALTER TABLE definiert. Sie fügen zuerst referenzielle Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen hinzu und ordnen diesen dann Integritätsbedingungsattribute zu, indem Sie angeben, ob der Datenbankmanager die Integritätsbedingung umsetzen soll und ob die Integritätsbedingung zur Abfrageoptimierung verwendet werden soll.

Entwerfen von Integritätsbedingungen

Beim Entwerfen und Erstellen von Integritätsbedingungen empfiehlt es sich, eine Namenskonvention zu verwenden, die eine geeignete Unterscheidung der verschiedenen Typen von Integritätsbedingungen ermöglicht. Dies spielt insbesondere für die Diagnose von Fehlern, die möglicherweise auftreten, eine wichtige Rolle.

Sie können die folgenden Integritätsbedingungstypen entwerfen:

- NOT NULL, Integritätsbedingung
- Eindeutige Integritätsbedingungen
- Integritätsbedingungen über Primärschlüssel
- Prüfungen auf Integritätsbedingungen (in Tabellen)
- Integritätsbedingungen über Fremdschlüssel (referenzielle Integritätsbedingungen)
- Informative Integritätsbedingungen

Entwerfen eindeutiger Integritätsbedingungen

Eindeutige Integritätsbedingungen stellen sicher, dass jeder Wert im angegebenen Schlüssel eindeutig ist. Eine Tabelle kann über mehrere eindeutige Integritätsbedingungen verfügen, wobei eine eindeutige Integritätsbedingung als Primärschlüssel definiert sein kann.

Einschränkungen

- Eine eindeutige Integritätsbedingung kann nicht in einer untergeordneten Tabelle definiert werden.
- Es kann nur einen Primärschlüssel pro Tabelle geben.

Eine eindeutige Integritätsbedingung wird mit der Klausel UNIQUE in der Anweisung CREATE TABLE bzw. ALTER TABLE definiert. Ein eindeutiger Schlüssel kann aus mehr als einer Spalte bestehen. In einer Tabelle ist mehr als eine eindeutige Integritätsbedingung zulässig.

Wenn die eindeutige Integritätsbedingung definiert ist, wird sie vom Datenbankmanager automatisch umgesetzt, wenn eine INSERT- oder UPDATE-Anweisung die Daten in der Tabelle ändert. Die eindeutige Integritätsbedingung wird mithilfe eines eindeutigen Index realisiert.

Wenn eine eindeutige Integritätsbedingung in einer Anweisung ALTER TABLE definiert wird und ein Index für dieselbe Gruppe von Spalten dieses eindeutigen Schlüssels existiert, wird dieser Index zum eindeutigen Index und wird von der Integritätsbedingung verwendet.

Sie können jede einzelne eindeutige Integritätsbedingung als Primärschlüssel verwenden. Der Primärschlüssel kann als übergeordneter Schlüssel in einer referenziellen Integritätsbedingung (zusammen mit anderen eindeutigen Integritätsbedingungen) verwendet werden. Ein Primärschlüssel wird mit der Klausel PRIMARY KEY in der Anweisung CREATE TABLE bzw. ALTER TABLE definiert. Der Primärschlüssel kann aus mehr als einer Spalte bestehen.

Der Primärindex sorgt zwingend dafür, dass der Primärschlüssel eindeutig ist. Wenn eine Tabelle mit einem Primärschlüssel erstellt wird, legt der Datenbankmanager einen Primärindex für diesen Schlüssel an.

Einige Hinweise zur Leistungsoptimierung für Indizes, die für eindeutige Integritätsbedingungen verwendet werden:

Beim einleitenden Laden einer leeren Tabelle mit Indizes können mit LOAD bessere Leistungen erzielt werden als mit IMPORT. Dies ist von der Verwendung des Modus INSERT oder REPLACE für LOAD unabhängig. Beim Anhängen einer beträchtlichen Datenmenge an eine vorhandene Tabelle mit Indizes (unter Verwendung von IMPORT INSERT oder LOAD INSERT) ist LOAD etwas leistungsstärker als IMPORT. Wenn Sie den Befehl IMPORT für das einleitende Laden einer großen Datenmenge verwenden, erstellen Sie den eindeutigen Schlüssel nach dem Import bzw. Laden der Daten. Dadurch wird der Aufwand für die Verwaltung des Index während des Ladens der Tabelle vermieden. Darüber hinaus verwendet der Index auf diese Weise die geringste Menge an Speicher. Wenn Sie das Dienstprogramm zum Laden (LOAD) im Modus REPLACE verwenden, erstellen Sie den eindeutigen Schlüssel, bevor Sie die Daten laden. In diesem Fall ist die Erstellung des Index während des Ladens effizienter als die Verwendung der Anweisung CREATE INDEX nach der Ladeoperation.

Entwerfen von Integritätsbedingungen über Primärschlüssel

Jede Tabelle kann einen und nur einen Primärschlüssel besitzen. Ein Primärschlüssel ist eine Spalte oder eine Kombination von Spalten, die die gleichen Merkmale wie eine eindeutige Integritätsbedingung besitzen. Ein Primärschlüssel und Integritätsbedingungen über Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

Da ein Primärschlüssel zur Angabe einer Zeile in einer Tabelle verwendet wird, muss er eindeutig sein und möglichst wenigen Hinzufüge- oder Löschoperationen unterliegen. Eine Tabelle kann nicht mehr als einen Primärschlüssel, jedoch mehrere eindeutige Schlüssel besitzen. Primärschlüssel sind optional und können mit der Klausel PRIMARY KEY definiert werden, wenn eine Tabelle erstellt oder geändert wird. Sie besitzen zudem den weiteren Vorteil, dass sie für eine Reihenfolge der Daten sorgen, wenn Daten exportiert oder reorganisiert werden.

Integritätsbedingung über Primärschlüssel werden wie eindeutige Integritätsbedingungen entworfen (siehe „Entwerfen eindeutiger Integritätsbedingungen” auf Seite 314). Der einzige Unterschied besteht darin, dass im Gegensatz zu eindeutigen Integritätsbedingungen jeweils nur ein Primärschlüssel pro Tabelle definiert werden kann.

Anmerkung: Integritätsbedingungen über Primärschlüssel, die auf zusammengesetzten Primärschlüsseln basieren, sind zulässig.

Entwerfen von Prüfungen auf Integritätsbedingungen

Bei der Erstellung von Prüfungen auf Integritätsbedingungen können zwei Fälle auftreten: (i) Alle Zeilen entsprechen der Prüfung auf Integritätsbedingung. (ii) Einige oder alle Zeilen entsprechen der Prüfung auf Integritätsbedingung nicht.

Alle Zeilen entsprechen der Prüfung auf Integritätsbedingung
Wenn alle Zeilen der Prüfung auf Integritätsbedingung entsprechen, wird die Prüfung auf Integritätsbedingung erfolgreich erstellt. Nachfolgende Versuche, Daten einzufügen oder zu aktualisieren, die der Integritätsgeschäftsregel nicht entsprechen, werden zurückgewiesen.

Einige oder alle Zeilen entsprechen der Prüfung auf Integritätsbedingung nicht
Wenn einige Zeilen vorhanden sind, die der Prüfung auf Integritätsbedingung nicht entsprechen, wird die Prüfung auf Integritätsbedingung nicht erstellt (d. h. die Ausführung der Anweisung ALTER TABLE schlägt fehl). Das folgende Beispiel zeigt eine Anweisung ALTER TABLE, die einer Tabelle EMPLOYEE eine neue Integritätsbedingung hinzufügt. Die Prüfung auf Integritätsbedingung hat den Namen CHECK_JOB. Der Datenbankmanager verwendet diesen Namen, um Sie darüber zu informieren, gegen welche Integritätsbedingung verstoßen wurde, wenn eine INSERT- oder UPDATE-Anweisung fehlschlägt. Die Klausel CHECK wird zur Definition einer Prüfung auf Integritätsbedingung in einer Tabelle verwendet.

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT check_job
  CHECK (JOB IN ('Engineer', 'Sales', 'Manager'));
```

Es wurde eine Anweisung ALTER TABLE verwendet, da die Tabelle bereits definiert war. Falls in der Tabelle EMPLOYEE Werte vorhanden sind, die gegen die zu definierende Integritätsbedingung verstoßen, wird die Anweisung ALTER TABLE nicht erfolgreich ausgeführt.

Da Prüfungen auf Integritätsbedingungen und andere Typen von Integritätsbedingungen zur Implementierung von Geschäftsregeln dienen, müssen sie möglicherweise ab und zu geändert werden. Dies kann zum Beispiel der Fall sein, wenn sich Geschäftsregeln in einem Unternehmen ändern. Wenn eine Prüfung auf Integritätsbedingung geändert werden muss, kann dies nur dadurch geschehen, dass sie gelöscht und eine neue Prüfung auf Integritätsbedingung erstellt wird. Prüfungen auf Integritätsbedingungen können jederzeit gelöscht werden, und die Tabelle oder die darin enthaltenen Daten werden von dieser Aktion nicht berührt. Wenn Sie eine Prüfung auf Integritätsbedingung löschen, müssen Sie beachten, dass die Datenprüfung, die durch die Integritätsbedingung ausgeführt wurde, nicht mehr stattfindet.

Vergleich zwischen Prüfungen auf Integritätsbedingungen und BEFORE-Triggern

Sie müssen den Unterschied zwischen Prüfungen auf Integritätsbedingungen und Triggern berücksichtigen, wenn Sie die Verwendung von Triggern oder Prüfungen auf Integritätsbedingungen zur Wahrung der Integrität Ihrer Daten in Betracht ziehen.

In einer relationalen Datenbank muss die Datenintegrität gewährleistet werden, wenn mehrere Benutzer auf die Daten zugreifen oder sie ändern. Immer dann, wenn Daten gemeinsam verwendet werden, muss die Richtigkeit der Werte innerhalb von Datenbanken sichergestellt werden.

Prüfungen auf Integritätsbedingungen

Eine Prüfung auf Integritätsbedingung (in Tabellen) definiert Einschränkungen für Daten, die einer bestimmten Tabelle hinzugefügt werden. Mithilfe einer Prüfung auf Integritätsbedingung in Tabellen können Sie Einschränkungen über die Datentypeinschränkungen hinaus für die Werte definieren, die für eine Spalte in der Tabelle zulässig sind. Prüfungen auf Integritätsbedingungen in Tabellen werden in Form von Bereichsprüfungen oder Prüfungen auf andere Werte in derselben Zeile derselben Tabelle durchgeführt.

Wenn die Regel für alle Anwendungen gilt, die die Daten verwenden, müssen Sie eine Prüfung auf Integritätsbedingung in Tabellen verwenden, um Ihre Einschränkung für die in der Tabelle zulässigen Daten umzusetzen. Mithilfe von Prüfungen auf Integritätsbedingungen in Tabellen kann die Einschränkung allgemein angewendet und kann einfacher verwaltet werden.

Die Umsetzung der Prüfungen auf Integritätsbedingungen ist für die Pflege der Datenintegrität wichtig; darüber hinaus wird dabei aber auch eine bestimmte Menge an Systemaufwand generiert, der die Leistung beeinflussen kann, sobald große Datenvolumen geändert werden.

BEFORE-Trigger

Durch die Verwendung von Triggern, die vor einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, können Werte, die aktualisiert oder eingefügt wurden, geändert werden, *bevor* die Datenbank tatsächlich geändert wird. Falls dies gewünscht wird, können sie zur Umsetzung von Eingaben aus dem Anwendungsformat (Benutzersicht der Daten) in ein internes Datenbankformat verwendet werden. BEFORE-Trigger können auch zur Aktivierung anderer Operationen als Datenbankoperationen über benutzerdefinierte Funktionen verwendet werden.

Neben der Änderung ist die Datenprüfung ein häufiges Einsatzgebiet von BEFORE-Triggern, wobei die Klausel SIGNAL verwendet wird.

Zwischen BEFORE-Triggern und Prüfungen auf Integritätsbedingungen bestehen zwei Unterschiede, wenn sie für die Datenprüfung verwendet werden:

1. BEFORE-Trigger unterliegen im Gegensatz zu Prüfungen auf Integritätsbedingungen keiner Einschränkung hinsichtlich des Zugriffs auf andere Werte in derselben Zeile derselben Tabelle.
2. Bei einer SET INTEGRITY-Operation für eine Tabelle nach einer LOAD-Operation werden Trigger (einschließlich BEFORE-Trigger) nicht ausgeführt. Prüfungen auf Integritätsbedingungen werden jedoch ausgeführt.

Entwerfen von (referenziellen) Integritätsbedingungen über Fremdschlüssel

Referenzielle Integrität wird implementiert, indem den Tabellen- und Spaltendefinitionen Integritätsbedingungen über Fremdschlüssel (oder referenzielle Integritätsbedingungen) hinzugefügt werden und um einen Index für alle Fremdschlüsselspalten zu erstellen. Wenn der Index und die Integritätsbedingungen über Fremdschlüssel definiert sind, werden Änderungen an Daten in den Tabellen und Spalten an der definierten Integritätsbedingung überprüft. Die Durchführung der angeforderten Aktion hängt von dem Ergebnis der Prüfung der Integritätsbedingung ab.

Referenzielle Integritätsbedingungen werden mit der Klausel FOREIGN KEY und der Klausel REFERENCES in der Anweisung CREATE TABLE oder ALTER TABLE definiert. Es gibt Auswirkungen einer referenziellen Integritätsbedingung auf eine typisierte Tabelle bzw. auf eine übergeordnete Tabelle, die eine typisierte Tabelle ist, die Sie vor der Erstellung einer referenziellen Integritätsbedingung berücksichtigen sollten.

Die Definition von Fremdschlüsseln implementiert Integritätsbedingungen für die Werte innerhalb der Zeilen einer Tabelle oder zwischen den Zeilen zweier Tabellen. Der Datenbankmanager prüft die in einer Tabellendefinition angegebenen Integritätsbedingungen und verwaltet die Abhängigkeitsbeziehungen entsprechend. Das Ziel besteht darin, die Integrität ohne Leistungseinbußen zu erhalten, wenn ein Datenbankobjekt auf ein anderes verweist.

Zum Beispiel enthalten sowohl der Primärschlüssel als auch der Fremdschlüssel eine Spalte für die Abteilungsnummer. In der Tabelle EMPLOYEE heißt der Spaltenname WORKDEPT, während in der Tabelle DEPARTMENT der Name DEPTNO ist. Die Abhängigkeitsbeziehung zwischen diesen beiden Tabellen wird durch die folgenden Integritätsbedingungen definiert:

- Für jeden Mitarbeiter in der Tabelle EMPLOYEE gibt es nur eine Abteilungsnummer, und diese Nummer ist in der Tabelle DEPARTMENT vorhanden.
- Jede Zeile in der Tabelle EMPLOYEE verweist auf nur eine Zeile in der Tabelle DEPARTMENT. Es gibt eine eindeutige Beziehung zwischen den Tabellen.
- Jede Zeile in der Tabelle EMPLOYEE, die einen Nichtnullwert in der Spalte WORKDEPT enthält, hat eine Beziehung zu einer Zeile in der Spalte DEPTNO der Tabelle DEPARTMENT.
- Die Tabelle DEPARTMENT ist eine übergeordnete Tabelle, und die Tabelle EMPLOYEE ist die abhängige Tabelle.

Die Anweisung zur Definition der übergeordneten Tabelle DEPARTMENT sieht folgendermaßen aus:

```
CREATE TABLE DEPARTMENT
       (DEPTNO    CHAR(3)     NOT NULL,
        DEPTNAME  VARCHAR(29) NOT NULL,
        MGRNO     CHAR(6),
        ADMRDEPT  CHAR(3)     NOT NULL,
        LOCATION  CHAR(16),
           PRIMARY KEY (DEPTNO))
    IN RESOURCE
```

Die Anweisung zur Definition der abhängigen Tabelle EMPLOYEE sieht folgendermaßen aus:

```
CREATE TABLE EMPLOYEE
       (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
        FIRSTNME  VARCHAR(12) NOT NULL,
        LASTNAME  VARCHAR(15) NOT NULL,
        WORKDEPT  CHAR(3),
        PHONENO   CHAR(4),
        PHOTO     BLOB(10m)   NOT NULL,
           FOREIGN KEY DEPT (WORKDEPT)
           REFERENCES DEPARTMENT ON DELETE NO ACTION)
    IN RESOURCE
```

Durch Angeben der Spalte DEPTNO als Primärschlüssel der Tabelle DEPARTMENT und der Spalte WORKDEPT als Fremdschüssel der Tabelle EMPLOYEE definieren Sie eine referenzielle Integritätsbedingung für die Werte der Spalte WORKDEPT. Durch diese Integritätsbedingung wird die referenzielle Integrität zwischen den Werten der beiden Tabellen implementiert. In vorliegenden Fall müssen alle Mitarbeiter (Employees), die der Tabelle EMPLOYEE hinzugefügt werden, eine Abteilungsnummer erhalten, die in der Tabelle DEPARTMENT enthalten ist.

Die Löschbedingung für die referenzielle Integritätsbedingung in der Tabelle EMPLOYEE ist NO ACTION. Das heißt, dass eine Abteilung (Department) nicht aus der Tabelle DEPARTMENT gelöscht werden kann, wenn es Mitarbeiter in dieser Abteilung gibt.

Zum Hinzufügen einer referenziellen Integritätsbedingung kann nicht nur die in den Beispielen gezeigte Anweisung CREATE TABLE, sondern auch die Anweisung ALTER TABLE verwendet werden.

Ein weiteres Beispiel: Es werden dieselben Tabellendefinitionen wie im vorigen Beispiel verwendet. Außerdem wird die Tabelle DEPARTMENT vor der Tabelle EMPLOYEE erstellt. Jede Abteilung hat einen Manager, und dieser Manager wird in der Tabelle EMPLOYEE aufgeführt. Die Spalte MGRNO der Tabelle DEPARTMENT stellt tatsächlich einen Fremdschlüssel der Tabelle EMPLOYEE dar. Aufgrund dieses referenziellen Zyklus führt diese Integritätsbedingung zu einem kleinen Problem. Sie könnten einen Fremdschlüssel später hinzufügen. Sie könnten außerdem die Anweisung CREATE SCHEMA verwenden, um die Tabellen EMPLOYEE und DEPARTMENT gleichzeitig zu erstellen.

Siehe auch „Fremdschlüssel in referenziellen Integritätsbedingungen" auf Seite 322.

Beispiele für die Interaktion zwischen Triggern und referenziellen Integritätsbedingungen

Aktualisierungsoperationen (UPDATE) können zu Interaktionen von Triggern mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen führen.

Abb. 23 auf Seite 320 und die zugehörige Beschreibung sind für die Verarbeitung repräsentativ, die für eine Anweisung ausgeführt wird, die Daten in der Datenbank aktualisiert.

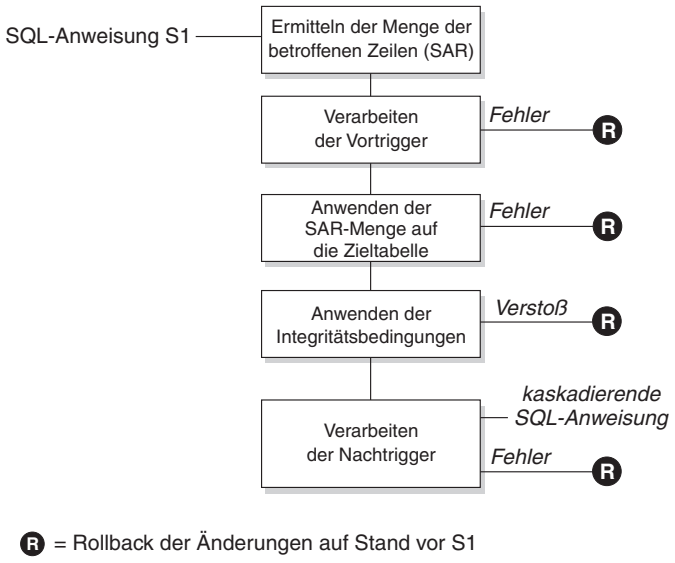


Abbildung 23. Verarbeitung einer Anweisung mit definierten Triggern und Integritätsbedingungen

Abb. 23 zeigt den allgemeinen Verarbeitungsablauf für eine Anweisung, die eine Tabelle aktualisiert. Sie geht von einer Situation aus, bei der die Tabelle Vortrigger, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Nachtrigger enthält, die hintereinander (kaskadierend) verarbeitet werden. Im Folgenden werden die Kästchen und anderen Elemente in Abb. 23 beschrieben.

- Anweisung S_1

  Dies ist die DELETE-, INSERT- oder UPDATE-Anweisung, die den Prozess erfordert. Die Anweisung S_1 gibt eine Tabelle (bzw. eine aktualisierbare Sicht für eine Tabelle) an, die in dieser Beschreibung als *Subjekttabelle* bezeichnet wird.

- Ermitteln der Menge der betroffenen Zeilen

  Dieser Schritt ist der Ausgangspunkt für einen Prozess, der für die Löschregeln CASCADE und SET NULL referenzieller Integritätsbedingungen sowie für kaskadierende Anweisungen aus Nachtriggern wiederholt ausgeführt wird.

  Zweck dieses Schritts ist die Ermittlung der *Menge der betroffenen Zeilen* (Set of Affected Rows - SAR) für die Anweisung. Die in dieser Menge enthaltenen Zeilen hängen von der Anweisung ab:

  – Für DELETE: Alle Zeilen, die die Suchbedingung der Anweisung erfüllen (oder die aktuelle Zeile bei einer positionierten DELETE-Operation)
  – Für INSERT: Die Zeilen, die durch die Klausel VALUES oder den Fullselect angegeben werden
  – Für UPDATE: Alle Zeilen, die die Suchbedingung erfüllten (oder die aktuelle Zeile bei einer positionierten UPDATE-Operation)

  Wenn die Menge der betroffenen Zeilen leer ist, sind keine Vortrigger, keine Änderungen zur Anwendung auf die Subjekttabelle und keine Integritätsbedingungen für die Anweisung zu verarbeiten.

- Verarbeiten der Vortrigger

  Alle Vortrigger (BEFORE) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet. Jeder Vortrigger verarbeitet die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

  Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

Wenn keine Vortrigger definiert sind oder die Menge der betroffenen Zeilen leer ist, wird dieser Schritt übersprungen.

- Anwenden der Menge der betroffenen Zeilen auf die Subjekttabelle

  Die tatsächliche DELETE-, INSERT- oder UPDATE-Operation wird unter Verwendung der Menge der betroffenen Zeilen auf die Subjekttabelle (Zieltabelle) in der Datenbank angewendet.

  Beim Anwenden der Menge der betroffenen Zeilen kann ein Fehler auftreten (z. B. wenn versucht wird, eine Zeile mit einem bereits vorhandenen Schlüssel einzufügen, obwohl ein eindeutiger Index vorhanden ist). In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Anwenden der Integritätsbedingungen

  Die der Subjekttabelle zugeordneten Integritätsbedingungen werden angewendet, wenn die Menge der betroffenen Zeilen nicht leer ist. Dazu gehören eindeutige Integritätsbedingungen, eindeutige Indizes, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Prüfungen der Klausel WITH CHECK OPTION für Sichten. Referenzielle Integritätsbedingungen mit den Löschregeln CASCADE oder SET NULL können die Aktivierung weiterer Trigger bewirken.

  Ein Verstoß gegen eine Integritätsbedingung oder die Klausel WITH CHECK OPTION führt zu einem Fehler. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Verarbeiten der Nachtrigger

  Alle von S_1 aktivierten Nachtrigger (AFTER) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet.

  Trigger mit der Definition FOR EACH STATEMENT verarbeiten die ausgelöste Aktion genau einmal, auch wenn die Menge der betroffenen Zeilen leer ist. Trigger mit der Definition FOR EACH ROW verarbeiten die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

  Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

  Die ausgelöste Aktion eines Triggers kann ausgelöste Anweisungen enthalten, bei denen es sich um DELETE-, INSERT- oder UPDATE-Anweisungen handelt. Zum Zweck dieser Beschreibung wird eine solche Anweisung als *kaskadierende Anweisung* betrachtet.

  Eine kaskadierende Anweisung ist eine DELETE-, INSERT- oder UPDATE-Anweisung, die als Teil der ausgelösten Aktion eines Nachtriggers verarbeitet wird. Eine solche Anweisung wird auf einer nachfolgenden Ebene der Triggerverarbeitung gestartet. Dies lässt sich in etwa so beschreiben, dass die ausgelöste Anweisung als neue Anweisung S_1 zugeordnet wird und alle hier beschriebenen Schritte wiederum rekursiv ausgeführt werden.

  Wenn alle ausgelösten Anweisungen aus allen Nachtriggern, die von jeder Anweisung S_1 aktiviert werden, vollständig verarbeitet wurden, wird die Verarbeitung der ursprünglichen Anweisung S_1 abgeschlossen.

- R = Rollback der Änderungen auf Stand vor S_1

  Jeder Fehler (einschließlich Verstößen gegen Integritätsbedingungen) während der Verarbeitung führt zu einem Rollback aller Änderungen, die direkt oder indirekt infolge der ursprünglichen Anweisung S_1 ausgeführt wurden. Die Datenbank wird in den Zustand zurückversetzt, den sie unmittelbar vor der Ausführung der ursprünglichen Anweisung S_1 hatte.

Fremdschlüssel in referenziellen Integritätsbedingungen

Ein Fremdschlüssel verweist auf einen Primärschlüssel oder einen eindeutigen Schlüssel in derselben oder einer anderen Tabelle. Die Zuordnung eines Fremdschlüssels zeigt an, dass die referenzielle Integrität gemäß der angegebenen referenziellen Integritätsbedingungen erhalten bleiben soll.

Ein Fremdschlüssel wird mit der Klausel FOREIGN KEY in der Anweisung CREATE TABLE bzw. ALTER TABLE definiert. Durch einen Fremdschlüssel wird die Tabelle von einer anderen Tabelle abhängig gemacht, die als übergeordnete Tabelle bezeichnet wird. Die Werte in der Spalte bzw. der Gruppe von Spalten, die den Fremdschlüssel in der einen Tabelle bilden, müssen mit den Werten des eindeutigen Schlüssels bzw. des Primärschlüssels der übergeordneten Tabelle übereinstimmen.

Die Anzahl der Spalten im Fremdschlüssel muss mit der Anzahl der Spalten im entsprechenden Primärschlüssel oder eindeutigen Schlüssel (auch als übergeordneter Schlüssel bezeichnet) der übergeordneten Tabelle übereinstimmen. Darüber hinaus müssen sich entsprechende Teile der Schlüsselspaltendefinitionen dieselben Datentypen und Datenlängen haben. Dem Fremdschlüssel kann ein *Integritätsbedingungsname* zugeordnet werden. Wenn Sie keinen Namen zuordnen, wird der Name automatisch zugeordnet. Zur leichteren Handhabung wird empfohlen, einen *Integritätsbedingungsnamen* zuzuordnen und nicht den vom System generierten Namen zu verwenden.

Der Wert eines zusammengesetzten Fremdschlüssels stimmt mit dem Wert eines Primärschlüssels überein, **wenn** der Wert jeder Spalte des Fremdschlüssels gleich dem Wert der entsprechenden Spalte des Primärschlüssels ist. Ein Fremdschlüssel, der Nullwerte enthält, kann nicht mit den Werten des Primärschlüssels übereinstimmen, da per Definition ein Primärschlüssels keine Nullwerte enthalten darf. Ein Nullwert eines Fremdschlüssels ist immer gültig, ungeachtet der Werte seiner Nichtnullwertspalten.

Für die Definition von Fremdschlüsseln gelten die folgenden Regeln:

- Eine Tabelle kann viele Fremdschlüssel enthalten.
- Ein Fremdschlüssel erfordert keine Eingabe (d. h. kann Nullwerte enthalten), wenn für irgendeinen seiner Teile keine Eingabe erforderlich ist (Nullwerte zulässig sind).
- Ein Fremdschlüsselwert ist ein Nullwert, wenn irgendeiner seiner Teile ein Nullwert ist.

Bei der Arbeit mit Fremdschlüsseln haben Sie folgende Möglichkeiten:

- Erstellen einer Tabelle ohne oder mit beliebig vielen Fremdschlüsseln
- Definieren von Fremdschlüsseln beim Erstellen oder Ändern einer Tabelle
- Löschen von Fremdschlüssels beim Ändern einer Tabelle

Auswirkungen von Tabellenintegritätsbedingungen auf Dienstprogrammoperationen

Wenn für die Tabelle, in die Daten geladen werden, referenzielle Integritätsbedingungen definiert sind, versetzt das Dienstprogramm LOAD die Tabelle in den Status 'Festlegen der Integrität anstehend', um Ihnen mitzuteilen, dass die Anweisung SET INTEGRITY für die Tabelle ausgeführt werden muss, um die referenzielle Integrität der geladenen Zeilen zu prüfen. Nach Abschluss des Dienstprogramms LOAD müssen Sie die Anweisung SET INTEGRITY absetzen, um eine

Prüfung der referenziellen Integrität für die geladenen Zeilen auszuführen und die Tabelle aus dem Status 'Festlegen der Integrität anstehend' herauszuholen.

Wenn beispielsweise die Tabellen DEPARTMENT und EMPLOYEE die einzigen Tabellen sind, die in den Status 'Festlegen der Integrität anstehend' gesetzt wurden, können Sie die folgende Anweisung ausführen:

```
SET INTEGRITY FOR DEPARTMENT ALLOW WRITE ACCESS,
   EMPLOYEE ALLOW WRITE ACCESS,
   IMMEDIATE CHECKED FOR EXCEPTION IN DEPARTMENT,
   USE DEPARTMENT_EX,
   IN EMPLOYEE USE EMPLOYEE_EX
```

Das Dienstprogramm zum Importieren (IMPORT) wird auf folgende Weise durch die referenziellen Integritätsbedingungen beeinflusst:

- Die Funktionen REPLACE und REPLACE CREATE sind nicht zulässig, wenn die Objekttabelle außer sich selbst noch andere abhängige Tabellen aufweist.

  Um diese Funktionen verwenden zu können, müssen erst alle Fremdschlüssel gelöscht werden, in denen die Tabelle eine übergeordnete Tabelle ist. Wenn die IMPORT-Operation abgeschlossen ist, erstellen Sie die Fremdschlüssel mit der Anweisung ALTER TABLE erneut.
- Der Erfolg eines Imports in eine Tabelle mit auf sich selbst verweisenden Integritätsbedingungen hängt von der Reihenfolge ab, in der die Zeilen importiert werden.

Anweisungsabhängigkeiten beim Ändern von Objekten

Anweisungsabhängigkeiten gelten für Pakete und im Cache zwischengespeicherte dynamische SQL- und XQuery-Anweisungen. Ein *Paket* ist ein Datenbankobjekt, das die Informationen enthält, die vom Datenbankmanager für den effizientesten Zugriff auf Daten für ein bestimmtes Anwendungsprogramm benötigt werden. Das *Binden* ist der Prozess, bei dem das Paket erstellt wird, das der Datenbankmanager für den Zugriff auf die Datenbank bei der Ausführung der Anwendung benötigt.

Pakete und im Cache zwischengespeicherte dynamische SQL- und XQuery-Anweisungen können von vielen Typen von Objekten abhängig sein.

Auf diese Objekte kann explizit verwiesen werden, zum Beispiel, indem eine Tabelle oder eine benutzerdefinierte Funktion in einer SQL-Anweisung SELECT angegeben wird. Auf die Objekte kann auch implizit verwiesen werden, zum Beispiel, wenn eine abhängige Tabelle überprüft werden muss, um sicherzustellen, dass keine **referenziellen Integritätsbedingungen** verletzt werden, wenn eine Zeile in einer übergeordneten Tabelle gelöscht wird. Pakete sind außerdem von den Zugriffsrechten abhängig, die dem Ersteller des Pakets erteilt sind.

Wenn ein Paket oder eine dynamische Abfrageanweisung von einem Objekt abhängig ist und dieses Objekt gelöscht wird, wird das Paket bzw. die im Cache zwischengespeicherte dynamische Abfrageanweisung in den Status „ungültig" (engl. invalid) versetzt. Wenn ein Paket von einer benutzerdefinierten Funktion abhängig ist und diese Funktion gelöscht wird, wird das Paket in den Status „unbrauchbar" (engl. inoperative) mit folgenden Bedingungen versetzt:

- Eine im Cache zwischengespeicherte dynamische SQL- oder XQuery-Anweisung, die sich im Status „ungültig" befindet, wird bei ihrer nächsten Verwendung automatisch erneut optimiert. Wenn ein für die Anweisung erforderliches Objekt gelöscht wurde, schlägt die Ausführung der dynamischen SQL- oder XQuery-Anweisung möglicherweise mit einer entsprechenden Fehlernachricht fehl.

- Für ein Paket, das sich im Status „ungültig" befindet, wird bei der nächsten Verwendung ein impliziter Rebind durchgeführt. Für ein solches Paket kann aber auch ein expliziter Rebind durchgeführt werden. Wurde ein Paket als ungültig markiert, weil ein Trigger gelöscht wurde, kann das erneut gebundene Paket keinen Trigger mehr aufrufen.
- Für ein Paket, das sich im Status „unbrauchbar" befindet, muss ein expliziter Rebind durchgeführt werden, damit es verwendet werden kann.

Für Objekte föderierter Datenbanken gelten ähnliche Abhängigkeiten. Durch das Löschen eines Servers werden z. B. alle diesem Server zugeordneten Pakete oder im Cache zwischengespeicherte dynamische SQL-Anweisungen mit Verweisen auf Kurznamen ungültig gemacht.

In einigen Fällen ist es nicht möglich, einen Rebind für das Paket durchzuführen. Zum Beispiel, wenn eine Tabelle gelöscht und nicht wieder erstellt wurde, kann für das Paket kein Rebind durchgeführt werden. In diesem Fall muss entweder das Objekt neu erstellt oder die Anwendung so geändert werden, dass sie nicht mehr auf das gelöschte Objekt zugreift.

In vielen anderen Fällen, zum Beispiel, wenn eine **Integritätsbedingung** gelöscht wurde, kann für das Paket ein Rebind durchgeführt werden.

Mithilfe der folgenden Systemkatalogsichten können Sie den Status eines Pakets und die Abhängigkeiten des Pakets feststellen:

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

Entwerfen von informativen Integritätsbedingungen

Integritätsbedingungen, die der Datenbankmanager beim Einfügen oder Aktualisieren von Datensätzen umsetzt, können zu einem hohen Systemaufwand führen, besonders dann, wenn große Mengen an Datensätzen geladen werden, die über referenzielle Integritätsbedingungen verfügen. Wenn eine Anwendung Informationen bereits vor dem Einfügen eines Datensatzes in die Tabelle überprüft hat, ist die Verwendung von informativen Integritätsbedingungen möglicherweise effizienter als die Verwendung normaler Integritätsbedingungen.

Informative Integritätsbedingungen teilen dem Datenbankmanager mit, welchen Regeln die Daten entsprechen; allerdings werden die Regeln nicht vom Datenbankmanager umgesetzt. Diese Informationen können jedoch vom DB2-Optimierungsprogramm verwendet werden und zu einer besseren Leistung von SQL-Abfragen beitragen.

Im folgenden Beispiel wird die Verwendung von informativen Integritätsbedingungen und deren Funktionsweise dargestellt. In dieser einfachen Tabelle sind Informationen zum Alter und zum Geschlecht des Antragstellers enthalten:

```
CREATE TABLE APPLICANTS
  (
   AP_NO INT NOT NULL,
     GENDER  CHAR(1) NOT NULL,
CONSTRAINT GENDEROK
   CHECK (GENDER IN ('M', 'F'))
   NOT ENFORCED
   ENABLE QUERY OPTIMIZATION,
      AGE  INT NOT NULL,
CONSTRAINT AGEOK
```

```
  CHECK (AGE BETWEEN 1 AND 80)
  NOT ENFORCED
  ENABLE QUERY OPTIMIZATION,
);
```

In diesem Beispiel sind zwei Klauseln enthalten, die das Verhalten der Integritätsbedingungen für Spalten ändern. Bei der ersten Option handelt es sich um die Option NOT ENFORCED; diese weist den Datenbankmanager an, die Prüfung dieser Spalte nicht umzusetzen, wenn Daten eingefügt oder aktualisiert werden.

Bei der zweiten Option handelt es sich um die Option ENABLE QUERY OPTIMIZATION, die vom Datenbankmanager verwendet wird, wenn SELECT-Anweisungen für diese Tabelle ausgeführt werden. Wird dieser Wert angegeben, verwendet der Datenbankmanager bei der SQL-Optimierung die Informationen in der Integritätsbedingung.

Wenn in der Tabelle die Option NOT ENFORCED enthalten ist, kann das Verhalten von Einfügeanweisungen ungewohnt erscheinen. Die folgende SQL-Anweisung schlägt nicht fehl, wenn sie für die Tabelle APPLICANTS ausgeführt wird:

```
INSERT INTO APPLICANTS VALUES
 (1, 'M', 54),
 (2, 'F', 38),
 (3, 'M', 21),
 (4, 'F', 89),
 (5, 'C', 10),
 (6, 'S',100),
```

Der Antragsteller Nummer fünf hat den Wert C für GENDER (C = Child), und der Antragsteller Nummer sechs hat einen ungültigen Wert für GENDER (S = Senior) und überschreitet die Altersbegrenzung der Spalte AGE. In beiden Fällen ermöglicht der Datenbankmanager die Einfügeoperation, da die Integritätsbedingungen nicht umgesetzt (NOT ENFORCED) sind. Im Folgenden sehen Sie das Ergebnis einer SELECT-Anweisung für die Tabelle:

```
SELECT * FROM APPLICANTS
  WHERE GENDER = 'C';

APPLICANT  GENDER  AGE
---------  ------  ---

0 Satz/Sätze ausgewählt.
```

Der Datenbankmanager hat die falsche Antwort auf die Abfrage zurückgegeben, obwohl der Wert 'C' in der Tabelle gefunden wurde; allerdings wird dem Datenbankmanager über die Integritätsbedingung dieser Spalte mitgeteilt, dass der einzig gültige Wert 'M' bzw. 'F' ist. Auch das Schlüsselwort ENABLE QUERY OPTIMIZATION ermöglichte dem Datenbankmanager bei der Optimierung der Anweisung die Verwendung dieser Integritätsbedingungsinformationen. Wenn es sich bei diesem Verhalten nicht um das gewünschte Verhalten handelt, muss die Integritätsbedingung mithilfe der Anweisung ALTER TABLE geändert werden; dies wird im folgenden Beispiel verdeutlicht:

```
ALTER TABLE APPLICANTS
  ALTER CHECK AGEOK DISABLE QUERY OPTIMIZATION
```

Wenn die Abfrage erneut abgesetzt wird, gibt der Datenbankmanager die folgenden richtigen Ergebnisse zurück:

```
SELECT * FROM APPLICANTS
  WHERE SEC = 'C';

APPLICANT GENDER  AGE
```

```
---------  ------  ---
        5  C       10

1 Satz/Sätze ausgewählt.
```

Das beste Szenario für die Verwendung von informativen Integritätsbedingungen ist dann gegeben, wenn Sie garantieren können, dass es sich bei dem Anwendungsprogramm um die einzige Anwendung handelt, die die Daten einfügt und aktualisiert. Wenn die Anwendung bereits all Informationen im Vorfeld prüft (z. B. Geschlecht und Alter), kann sich aus der Verwendung von informativen Integritätsbedingungen eine schnellere Leistung und ein um die Hälfte reduzierter Aufwand ergeben. Ein weiterer möglicher Einsatzbereich informativer Integritätsbedingungen ist das Entwerfen von Data Warehouses.

Erstellen und Ändern von Integritätsbedingungen

Integritätsbedingungen können vorhandenen Tabellen mithilfe der Anweisung ALTER TABLE hinzugefügt werden.

Der Name der Integritätsbedingung darf nicht mit dem Namen einer anderen Integritätsbedingung innerhalb einer Anweisung ALTER TABLE übereinstimmen und muss innerhalb der Tabelle eindeutig sein (dies schließt auch Namen bereits definierter referenzieller Integritätsbedingungen mit ein). Vorhandene Daten werden gegen die neue Bedingung abgeglichen, bevor die Anweisung erfolgreich beendet wird.

Erstellen und Ändern eindeutiger Integritätsbedingungen

Eindeutige Integritätsbedingungen können einer vorhandenen Tabelle hinzugefügt werden. Der Name der Integritätsbedingung darf nicht mit dem Namen einer anderen Integritätsbedingung innerhalb derselben Anweisung ALTER TABLE übereinstimmen und muss innerhalb der Tabelle eindeutig sein (dies schließt auch Namen bereits definierter referenzieller Integritätsbedingungen mit ein). Vorhandene Daten werden anhand der neuen Bedingung überprüft, bevor die Anweisung erfolgreich ausgeführt wird.

Zur Definition von eindeutigen Integritätsbedingungen über die Befehlszeile verwenden Sie die Option ADD CONSTRAINT der Anweisung ALTER TABLE. Mit der folgenden Anweisung wird der Tabelle EMPLOYEE zum Beispiel eine eindeutige Integritätsbedingung hinzugefügt, die eine neue Methode zur eindeutigen Identifizierung von Mitarbeitern in der Tabelle darstellt:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Erstellen und Ändern von Integritätsbedingungen über Primärschlüssel

Einer vorhandenen Tabelle kann eine Integritätsbedingung über Primärschlüssel hinzugefügt werden. Der Name der Integritätsbedingung muss innerhalb der Tabelle (einschließlich der Namen aller definierten referenziellen Integritätsbedingungen) eindeutig sein. Vorhandene Daten werden gegen die neue Bedingung abgeglichen, bevor die Anweisung erfolgreich beendet wird.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um Primärschlüssel hinzuzufügen:

```
ALTER TABLE <name>
   ADD CONSTRAINT <spaltenname>
   PRIMARY KEY <spaltenname>
```

Eine vorhandene Integritätsbedingung kann nicht geändert werden. Wenn Sie eine andere Spalte bzw. eine andere Gruppe von Spalten als Primärschlüssel definieren wollen, muss die vorhandene Primärschlüsseldefinition zunächst gelöscht und anschließend erneut erstellt werden.

Erstellen und Ändern von Prüfungen auf Integritätsbedingungen

Wenn eine Prüfung auf Integritätsbedingung in einer Tabelle hinzugefügt wird, werden Pakete und im Cache zwischengespeicherte dynamische SQL-Anweisungen, die INSERT- oder UPDATE-Operationen für die Tabelle ausführen, eventuell als ungültig markiert.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um eine Prüfung auf Integritätsbedingung in der Tabelle hinzuzufügen:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Erstellen und Ändern von (referenziellen) Integritätsbedingungen über Fremdschlüssel

Ein Fremdschlüssel ist ein Verweis auf die Datenwerte in einer anderen Tabelle. Es gibt verschiedene Typen von Integritätsbedingungen über Fremdschlüssel.

Wenn ein Fremdschlüssel einer Tabelle hinzugefügt wird, werden Pakete und im Cache zwischengespeicherte dynamische SQL-Anweisungen, die folgende Arten von Anweisungen enthalten, eventuell als ungültig markiert:

- Anweisungen, die die Tabelle mit dem Fremdschlüssel aktualisieren oder dort Zeilen einfügen
- Anweisungen, die die übergeordnete Tabelle aktualisieren oder löschen

Geben Sie die folgende Anweisung in die Befehlszeile ein, um Fremdschlüssel hinzuzufügen:

```
ALTER TABLE <name>
   ADD CONSTRAINT <spaltenname>
    FOREIGN KEY <spaltenname>
    ON DELETE <aktionstyp>
    ON UPDATE <aktionstyp>
```

Die folgenden Beispiele zeigen die Verwendung der Anweisung ALTER TABLE zum Hinzufügen von Primär- und Fremdschlüsseln in einer Tabelle:

```
ALTER TABLE PROJECT
  ADD CONSTRAINT PROJECT_KEY
      PRIMARY KEY (PROJNO)
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
      PRIMARY KEY (EMPNO, PROJNO, ACTNO)
  ADD CONSTRAINT ACT_EMP_REF
      FOREIGN KEY (EMPNO)
      REFERENCES EMPLOYEE
      ON DELETE  RESTRICT
```

```
ADD CONSTRAINT ACT_PROJ_REF
    FOREIGN KEY (PROJNO)
    REFERENCES PROJECT
    ON DELETE CASCADE
```

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Erstellen und Ändern informativer Integrationsbedingungen

Zur Verbesserung der Leistung von Abfragen können Sie Ihren Tabellen informative Integritätsbedingungen hinzufügen. Sie können informative Integrationsbedingungen mithilfe der Anweisung CREATE TABLE oder ALTER TABLE hinzufügen, wenn Sie die Option NOT ENFORCED in der DDL-Anweisung angeben.

Einschränkung: Nach der Definition informativer Integritätsbedingungen in einer Tabelle können die Spaltennamen für diese Tabelle erst geändert werden, wenn die informativen Integritätsbedingungen entfernt wurden.

Zur Angabe informativer Integrationsbedingungen für eine Tabelle über die Befehlszeile geben Sie den folgenden Befehl für eine neue Tabelle ein:

```
ALTER TABLE <name> <attribute_der_integritätsbedingung> NOT ENFORCED
```

ENFORCED oder NOT ENFORCED: Gibt an, ob die Integritätsbedingung vom Datenbankmanager während normaler Operationen, wie z. B. Einfüge-, Aktualisierungs- oder Löschoperationen, erzwungen wird.

- ENFORCED kann nicht für eine funktionale Abhängigkeit angegeben werden (SQLSTATE-Wert 42621).
- NOT ENFORCED sollte nur angegeben werden, wenn die Tabellendaten der Integritätsbedingung entsprechen, ohne dass dies vom Datenbankmanager erzwungen wurde. Abfrageergebnisse können unvorhersehbar sein, wenn die Daten nicht tatsächlich der Integritätsbedingung entsprechen.

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Anzeigen der Definitionen von Integritätsbedingungen für eine Tabelle

Definitionen von Integritätsbedingungen für eine Tabelle befinden sich in den Katalogsichten SYSCAT.INDEXES und SYSCAT.REFERENCES.

Die Spalte UNIQUERULE der Sicht SYSCAT.INDEXES gibt das Merkmal des Index an. Wenn der Wert dieser Spalte 'P' ist, handelt es sich bei dem Index um einen Primärschlüsselindex. Ist der Wert 'U', handelt es sich um einen eindeutigen Index (jedoch nicht für einen Primärschlüssel).

Die Katalogsicht SYSCAT.REFERENCES enthält Informationen zu referenziellen Integritätsbedingungen (über Fremdschlüssel).

Löschen von Integritätsbedingungen

Eine Prüfung auf Integritätsbedingung in einer Tabelle (Table Check Constraint) kann explizit mithilfe einer Anweisung ALTER TABLE oder implizit als Ergebnis einer Anweisung DROP TABLE gelöscht werden.

Zum Löschen von Integritätsbedingungen verwenden Sie die Anweisung ALTER TABLE mit der Klausel DROP oder DROP CONSTRAINT. Dadurch können Sie die

Tabellen, die die betreffenden Spalten enthalten, binden (BIND) und weiterhin auf sie zugreifen. Die Namen aller eindeutigen Integritätsbedingungen für eine Tabelle befinden sich in der Systemkatalogsicht SYSCAT.INDEXES.

Löschen eindeutiger Integritätsbedingungen

Sie können eine eindeutige Integritätsbedingung explizit mit der Anweisung ALTER TABLE löschen.

Die Klausel DROP UNIQUE der Anweisung ALTER TABLE löscht die Definition der eindeutigen Integritätsbedingung mit dem angegebenen Namen sowie alle referenziellen Integritätsbedingungen, die von dieser eindeutigen Integritätsbedingung abhängig sind. Der angegebene Name muss eine vorhandene eindeutige Integritätsbedingung bezeichnen.

```
ALTER TABLE <tabellenname>
  DROP UNIQUE <name_der_eindeutigen_integritätsbedingung>
```

Durch das Löschen dieser eindeutigen Integritätsbedingung werden alle Pakete und im Cache zwischengespeicherte dynamische SQL-Anweisungen ungültig gemacht, die diese Integritätsbedingung verwendet haben.

Löschen von Integritätsbedingungen über Primärschlüssel

Mit der Klausel DROP PRIMARY KEY der Anweisung ALTER TABLE können Sie Integritätsbedingungen über Primärschlüssel löschen.

Die Klausel DROP PRIMARY KEY der Anweisung ALTER TABLE löscht die Definition des Primärschlüssels sowie alle referenziellen Integritätsbedingungen, die von diesem Primärschlüssel abhängig sind. Die Tabelle muss einen Primärschlüssel besitzen. Geben Sie die folgende Anweisung in die Befehlszeile ein, um einen Primärschlüssel zu löschen:

```
ALTER TABLE <tabellenname>
  DROP PRIMARY KEY
```

Löschen von Prüfungen auf Integritätsbedingungen (Tabelle)

Durch das Löschen einer Prüfung auf Integritätsbedingung werden alle Pakete und im Cache zwischengespeicherte dynamische Anweisungen mit INSERT- oder UPDATE-Abhängigkeiten für die Tabelle ungültig gemacht. Die Namen aller Prüfungen auf Integritätsbedingungen in einer Tabelle können mithilfe der Katalogsicht SYSCAT.CHECKS festgestellt werden. Vor dem Löschen einer Prüfung auf Integritätsbedingungen in einer Tabelle, die einen vom System generierten Namen hat, können Sie den Namen aus der Katalogsicht SYSCAT.CHECKS abfragen.

Mit der folgenden Anweisung wird die Prüfung auf Integritätsbedingung 'name_der_prüfung_auf_integritätsbedingung' gelöscht. Der angegebene Name muss eine vorhandene Prüfung auf Integritätsbedingung bezeichnen, die für die Tabelle definiert ist. Geben Sie die folgende Anweisung in die Befehlszeile ein, um eine Prüfung auf Integritätsbedingung in der Tabelle zu löschen:

```
ALTER TABLE <tabellenname>
  DROP <name_der_prüfung_auf_integritätsbedingung>
```

Löschen von (referenziellen) Integritätsbedingungen über Fremdschlüssel

Mit der Klausel DROP CONSTRAINT der Anweisung ALTER TABLE können Sie Integritätsbedingungen über Fremdschlüssel löschen.

Die Klausel DROP CONSTRAINT der Anweisung ALTER TABLE löscht die Integritätsbedingung mit dem angegeben Namen (*integritätsbedingungsname*). Der angegebene Name *integritätsbedingungsname* muss eine vorhandene Integritätsbedingung über Fremdschlüssel, eine Integritätsbedingung über Primärschlüssel oder eine eindeutige Integritätsbedingung bezeichnen,

die für die Tabelle definiert ist. Geben Sie die folgende Anweisung in die Befehlszeile ein, um Fremdschlüssel zu löschen:

```
ALTER TABLE <tabellenname>
  DROP FOREIGN KEY <fremdschlüsselname>
```

Im folgenden Beispiel werden die Klauseln DROP PRIMARY KEY und DROP FOREIGN KEY in der Anweisung ALTER TABLE verwendet, um Primär- und Fremdschlüssel zu löschen:

```
ALTER TABLE EMP_ACT
  DROP PRIMARY KEY
  DROP FOREIGN KEY ACT_EMP_REF
  DROP FOREIGN KEY ACT_PROJ_REF
ALTER TABLE PROJECT
  DROP PRIMARY KEY
```

Wenn eine Integritätsbedingung über Fremdschlüssel gelöscht wird, werden Pakete oder im Cache zwischengespeicherte dynamische Anweisungen, die folgende Arten von Anweisungen enthalten, eventuell als ungültig markiert:

- Anweisungen, die die Tabelle mit dem Fremdschlüssel aktualisieren oder dort Zeilen einfügen
- Anweisungen, die die übergeordnete Tabelle aktualisieren oder löschen

Kapitel 13. Indizes

Ein *Index* ist eine Gruppe aus einem oder mehreren Schlüsseln, wobei jeder Schlüssel auf eine Zeile in einer Tabelle zeigt. Das *SQL-Optimierungsprogramm* wählt automatisch den effizientesten Weg für den Zugriff auf Daten in Tabellen aus. Das Optimierungsprogramm bezieht Indizes bei der Ermittlung des schnellsten Pfads zu den Daten mit in Betracht.

Anmerkung: Nicht alle Indizes verweisen auf Zeilen in einer Tabelle. MDC-Blockindizes verweisen auf Speicherbereiche (bzw. Blöcke) von Daten. XML-Indizes für XML-Daten verwenden bestimmte XML-Musterausdrücke zur Indexierung von Pfaden und Werten in XML-Dokumenten, die innerhalb einer Spalte gespeichert sind. Der Datentyp einer solchen Spalte muss XML sein. Sowohl MDC-Blockindizes als auch XML-Indizes sind systemgenerierte Indizes.

Indizes werden vom Datenbankmanager wie folgt verwendet:

- Zur Verbesserung der Leistung. In den meisten Fällen kann mithilfe eines Indexes schneller auf Daten zugegriffen werden. Zwar kann ein Index nicht für eine Sicht erstellt werden, aber ein für die Tabelle, auf der eine Sicht basiert, erstellter Index kann gelegentlich die Leistung von Operationen für diese Sicht verbessern.
- Zur Sicherstellung der Eindeutigkeit. Eine Tabelle mit einem eindeutigen Index darf keine Zeilen mit identischen Schlüsseln haben.

Wenn Daten zu einer Tabelle hinzugefügt werden, werden sie einfach an das Ende der Tabelle angefügt, sofern nicht andere Aktionen für die Tabelle oder die hinzuzufügenden Daten durchgeführt wurden. Für die Daten gibt es keine Reihenfolge. Wenn nach einer bestimmten Datenzeile gesucht wird, muss von der ersten bis zur letzten Zeile jede Zeile der Tabelle überprüft werden. Indizes werden als Methode zum Zugreifen auf die Daten in der Tabelle in einer Reihenfolge verwendet, die möglicherweise andernfalls nicht verfügbar ist.

Ein Spaltenwert in einer Datenzeile kann zum Kennzeichnen der gesamten Zeile verwendet werden. Mindestens eine Spalte kann zum Kennzeichnen der Zeile erforderlich sein. Solche Spalten werden auch *Schlüssel* genannt. Eine Spalte kann in mehr als einem Schlüssel verwendet werden.

Ein Index wird nach den Werten in einem Schlüssel sortiert. Schlüssel können eindeutig oder nicht eindeutig sein. Jede Tabelle muss über mindestens einen eindeutigen Schlüssel verfügen; allerdings können die Tabellen aber auch über weitere nicht eindeutige Schlüssel verfügen. Jeder Index verfügt über genau einen Schlüssel. Beispiel: Sie können die Mitarbeiter-ID-Nummer (eindeutig) als Schlüssel für einen Index und die Abteilungsnummer (nicht eindeutig) als Schlüssel für einen anderen Index verwenden.

Beispiel

Tabelle A in Abb. 24 auf Seite 332 verfügt über einen Index, der auf den Personalnummern in der Tabelle basiert. Dieser Schlüsselwert dient als Zeiger auf die Zeilen in der Tabelle. Beispiel: Die Personalnummer 19 zeigt auf den Mitarbeiter KMP. Ein Index ermöglicht einen effizienten Zugriff auf Zeilen einer Tabelle, indem er einen Pfad zu den Daten mithilfe von Zeigern erstellt.

Es können eindeutige Indizes erstellt werden, um die Eindeutigkeit des Indexschlüssels sicherzustellen. Ein *Indexschlüssel* ist eine Spalte oder eine geordnete Gruppe von Spalten, auf denen ein Index definiert wird. Mithilfe eines eindeutigen Index wird gewährleistet, dass der Wert jedes Indexschlüssels in der indizierten Spalte bzw. den indizierten Spalten eindeutig ist.

Abb. 24 illustriert die Beziehung zwischen einem Index und einer Tabelle.

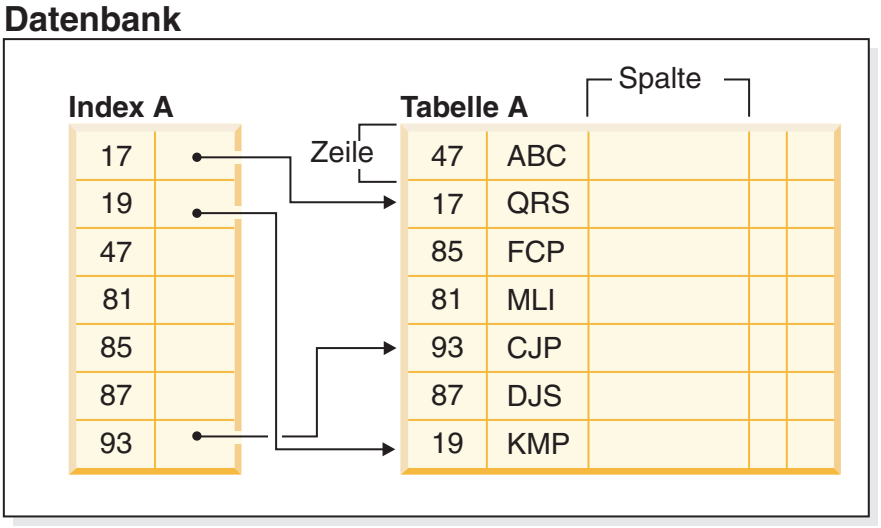


Abbildung 24. Beziehung zwischen einem Index und einer Tabelle

Abb. 25 auf Seite 333 veranschaulicht die Beziehungen zwischen einigen Datenbankobjekten. Sie zeigt außerdem, dass Tabellen, Indizes und lange Daten (LOB-Daten) in Tabellenbereichen gespeichert werden.

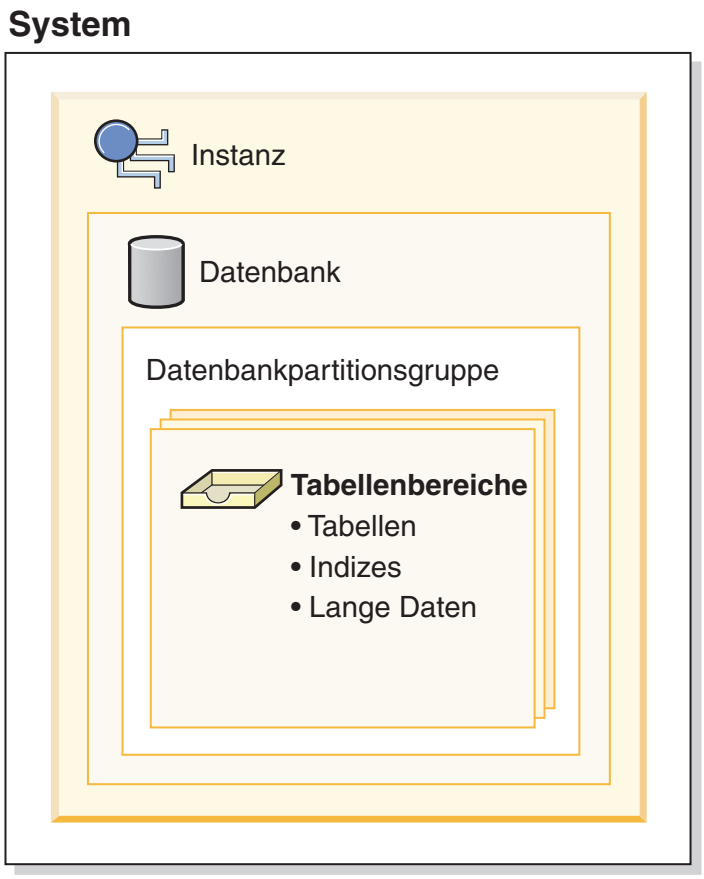


Abbildung 25. Beziehung zwischen ausgewählten Datenbankobjekten

Typen von Indizes

Es gibt fünf Typen von Indizes: eindeutige und nicht eindeutige Indizes, Clusterindizes und Nicht-Clusterindizes sowie vom System generierte Blockindizes für MDC-Tabellen.

Eindeutige und nicht eindeutige Indizes

Bei eindeutigen Indizes handelt es sich um Indizes, die zur Pflege der Datenintegrität beitragen, indem sie sicherstellen, dass keine zwei Datenzeilen in einer Tabelle über identische Schlüsselwerte verfügen.

Bei dem Versuch, einen eindeutigen Index für eine Tabelle zu erstellen, die bereits Daten enthält, werden Werte in der/den Spalten(n), aus der/denen der Index besteht, auf ihre Eindeutigkeit hin überprüft. Wenn die Tabelle Zeilen mit doppelten Schlüsselwerten enthält, schlägt die Indexerstellung fehl. Wenn ein eindeutiger Index für eine Tabelle definiert wurde, wird die Eindeutigkeit sichergestellt, wenn Schlüssel im Index hinzugefügt oder geändert werden. (Dies gilt zum Beispiel für Operationen wie INSERT, UPDATE, LOAD, IMPORT und SET INTEGRITY u. a.) Zusätzlich zur Gewährleistung der Eindeutigkeit von Datenwerten kann ein eindeutiger Index auch zur Verbesserung der Leistung beim Datenabruf während der Abfrageverarbeitung verwendet werden.

Auf der anderen Seite werden nicht eindeutige Indizes nicht für die Umsetzung von Integritätsbedingungen für die Tabellen verwendet, denen sie zugeordnet sind. Stattdessen werden nicht eindeutige Indizes ausschließlich zur Verbesserung der Abfrageleistung durch Beibehalten einer sortierten Reihenfolge von Datenwerten verwendet, die häufig benutzt werden.

Cluster- und Nicht-Clusterindizes

Indexarchitekturen werden als Cluster- und Nicht-Clusterindexarchitekturen klassifiziert. Clusterindizes sind Indizes, bei denen die Reihenfolge der Zeilen in den Datenseiten der Reihenfolge der Zeilen im Index entspricht. Dies der Grund, aus dem es nur einen Clusterindex für eine bestimmte Tabelle geben kann, während die Tabelle viele Nicht-Clusterindizes haben kann. Bei einigen Verwaltungssystemen für relationale Datenbanken entspricht der Blattknoten (Leaf Node) des Clusterindex den eigentlichen Daten und nicht einem Zeiger auf Daten, die sich an anderer Stelle befinden.

Clusterindizes und Nicht-Clusterindizes können nur Schlüssel und Satz-IDs (RIDs) in der Indexstruktur enthalten. Die Satz-IDs zeigen immer auf Zeilen in den Datenseiten. Der einzige Unterschied zwischen einem Clusterindex und einem Nicht-Clusterindex besteht darin, dass der Datenbankmanager versucht, die Daten in den Datenseiten in genau der Reihenfolge zu halten, in der die entsprechenden Schlüssel in den Indexseiten angeordnet sind. Daher versucht der Datenbankmanager, Zeilen mit ähnlichen Schlüsseln in dieselben Seiten einzufügen. Wenn die Tabelle reorganisiert wird, werden ihre Zeilen in der Reihenfolge der Inderschlüssel in die Datenseiten eingefügt.

Die Reorganisation einer Tabelle in Bezug auf einen ausgewählten Index dient zur erneuten Erstellung des Datenclusterings. Ein Clusterindex eignet sich am Besten für Spalten, die über Bereichsvergleichselemente verfügen, da dieser einen verbesserten sequenziellen Zugriff auf die Daten der Tabelle ermöglicht. Dadurch werden weniger Seitenabrufoperationen benötigt, weil sich ähnliche Werte in derselben Datenseite befinden.

Im Allgemeinen kann nur einer der Indizes in einer Tabelle einen hohen Grad an Clusterbildung aufweisen.

Verbessern der Leistung mithilfe von Clusterindizes

Mithilfe von Clusterindizes kann die Leistung der meisten Abfrageoperationen verbessert werden, da sie einen lineareren Zugriffspfad zu den Daten bereitstellen, die in Seiten gespeichert wurden. Außerdem ist der Vorablesezugriff in der Regel effizienter, wenn Clusterindizes verwendet werden, da Zeilen mit ähnlichen Indexschlüsselwerten zusammen gespeichert werden.

Allerdings können Clusterindizes nicht im Rahmen der Tabellendefinition angegeben werden, die mit der Anweisung CREATE TABLE verwendet wird. Stattdessen werden Clusterindizes nur durch die Ausführung der Anweisung CREATE INDEX mit der Option CLUSTER erstellt. Dann muss die Anweisung ALTER TABLE zum Hinzufügen eines Primärschlüssels verwendet werden, der dem für die Tabelle erstellten Clusterindex entspricht. Dieser Clusterindex wird dann als Primärschlüsselindex der Tabelle verwendet.

Anmerkung: Das Einstellen von PCTFREE in der Tabelle mit der Anweisung ALTER TABLE auf einen geeigneten Wert kann helfen, das Clustering der Tabelle aufrechtzuerhalten, indem ausreichend freier Speicherbereich zum Einfügen von

Zeilen in den Seiten mit ähnlichen Werten freigehalten wird. Weitere Informationen finden Sie in der Beschreibung der Anweisung ALTER TABLE sowie in „Senken des Bedarfs an Tabellen- und Indexreorganisationen" in *Optimieren der Datenbankleistung*.

Im Allgemeinen bleibt die Clusterbildung effektiver erhalten, wenn der Clusterindex eindeutig ist.

Unterschiede zwischen Integritätsbedingungen über Primärschlüssel bzw. eindeutige Schlüssel und eindeutigen Indizes

Es ist wichtig, zu verstehen, dass es keinen bedeutenden Unterschied zwischen einer Integritätsbedingung über Primärschlüssel bzw. eindeutige Schlüssel und einem eindeutigen Index gibt. Der Datenbankmanager verwendet eine Kombination aus einem eindeutigen Index und der Integritätsbedingung NOT NULL, um das Konzept relationaler Datenbanken von Integritätsbedingungen über Primärschlüssel bzw. eindeutige Schlüssel zu implementieren. Aus diesem Grund setzen eindeutige Indizes Integritätsbedingungen über Primärschlüssel nicht selbst um, denn sie lassen Nullwerte zu. (Zwar stellen Nullwerte unbekannte Werte dar, bei einer Indexierung wird ein Nullwert jedoch so behandelt, als würde er anderen Nullwerten entsprechen.)

Aus diesem Grund ist nur ein einziger Nullwert zulässig, wenn ein eindeutiger Index aus nur einer Spalte besteht. Ein weiterer Nullwert würde gegen die eindeutige Integritätsbedingung verstoßen. Bei einem eindeutigen Index aus mehreren Spalten kann dementsprechend eine bestimmte Kombination aus Werten und Nullen nur einmal verwendet werden.

Bidirektionale Indizes

Standardmäßig lassen bidirektionale Indizes Suchoperationen in beide Richtung, d. h. vorwärts und rückwärts, zu. Mit der Klausel ALLOW REVERSE SCANS in der Anweisung CREATE INDEX ist eine vorwärts- und rückwärtsgerichtete Indexsuche möglich, d. h., der Index kann in der Reihenfolge, die bei der Erstellung des Index definiert wurde, und in umgekehrter Richtung durchsucht werden. Diese Option bietet folgende Möglichkeiten:

- Sie können die Funktionen MIN und MAX besser nutzen.
- Sie können vorherige Schlüssel abrufen.
- Der Datenbankmanager muss keine temporäre Tabelle mehr für die Rückwärtssuche erstellen.
- Redundante Indizes in umgekehrter Reihenfolge werden eliminiert.

Wenn DISALLOW REVERSE SCANS angegeben ist, kann der Index nicht rückwärts durchsucht werden. (Physisch wird er jedoch genau so erstellt, wie ein Index mit ALLOW REVERSE SCANS.)

Entwerfen von Indizes

In der Regel dienen Indizes zur Beschleunigung des Zugriffs auf eine Tabelle. Sie können aber auch Zwecke des logischen Datenentwurfs erfüllen.

Zum Beispiel lässt ein eindeutiger Index nicht zu, dass in die Spalten ein Wert mehrfach eingegeben wird, wodurch gewährleistet wird, dass nicht zwei Zeilen in einer Tabelle identisch sind. Indizes können auch erstellt werden, um die Werte in einer Spalte aufsteigend oder absteigend anzuordnen.

Anmerkung: Bei der Erstellung von Indizes sollten Sie beachten, dass sie möglicherweise die Leseleistung verbessern können, sich im Gegenzug aber negativ auf die Schreibleistung auswirken können. Der Grund hierfür ist, dass für jede Zeile, die der Datenbankmanager in eine Tabelle schreibt, auch die betroffenen Indizes aktualisiert werden müssen. Deshalb sollten Sie Indizes nur dann erstellen, wenn sie einen eindeutigen Vorteil für die Gesamtleistung bringen.

Bei der Erstellung von Indizes müssen Sie außerdem die Struktur der Tabellen und den Typ von Abfragen berücksichtigen, die am häufigsten für sie durchgeführt werden. So sind beispielsweise Spalten, die in der Klausel WHERE einer häufig abgesetzten Abfrage auftreten, geeignete Kandidaten für Indizes. Bei weniger häufig ausgeführten Abfragen kann der Aufwand, der für einen Index für die Leistung in den Anweisungen INSERT und UPDATE entsteht, die Vorteile zunichte machen.

In ähnlicher Weise können Spalten, die in einer GROUP BY-Klausel einer häufigen Abfrage auftreten, die Erstellung eines Index vorteilhaft nutzen, insbesondere dann, wenn die Anzahl der zum Gruppieren der Zeilen verwendeten Werte im Gegensatz zur Anzahl der gruppierten Zeilen gering ist.

Richtlinien und Aspekte für den Entwurf von Indizes

- Ein *Index* wird durch Spalten der Tabelle definiert. Er kann vom Ersteller einer Tabelle oder von einem Benutzer, der weiß, dass für bestimmte Spalten ein direkter Zugriff erforderlich ist, definiert werden. Ein Primärindexschlüssel wird automatisch anhand des Primärschlüssels erstellt, sofern kein benutzerdefinierter Index bereits existiert.
- Ein *Indexschlüssel* ist eine Spalte oder eine Gruppe von Spalten, mit denen ein Index definiert wird. Die Zweckmäßigkeit eines Index hängt von seinem Schlüssel ab. Obwohl die Reihenfolge der Spalten, die einen Indexschlüssel bilden, bei der Erstellung des Indexschlüssels keine Rolle spielt, kann sie für das Optimierungsprogramm bei der Entscheidung von Bedeutung sein, ob ein Index verwendet werden soll oder nicht.
- Für eine bestimmte Tabelle kann eine beliebige Anzahl von Indizes definiert werden, und diese Indizes können sich positiv auf die Leistung von Abfragen auswirken. Der Indexmanager muss die Indizes bei Aktualisierungs-, Lösch- und Einfügeoperationen beibehalten. Daher kann die Erstellung einer großen Anzahl von Indizes für eine Tabelle, die häufig aktualisiert wird, die Verarbeitung von Anforderungen verlangsamen. Durch umfangreiche Indexschlüssel kann es ebenfalls zu einer Verlangsamung der Verarbeitungsgeschwindigkeit für Anforderungen kommen. Die Verwendung von Indizes ist also nur dann sinnvoll, wenn sich klare Vorteile für den häufigen Zugriff ergeben.
- Spaltendaten, die nicht Teil des eindeutigen Indexschlüssels sind, jedoch im Index gespeichert oder gepflegt werden sollen, werden als INCLUDE-Spalten bezeichnet. INCLUDE-Spalten können nur für eindeutige Indizes angegeben werden. Bei der Erstellung eines Index mit INCLUDE-Spalten werden nur die eindeutigen Schlüsselspalten sortiert und im Hinblick auf Eindeutigkeit geprüft. Die Verwendung von INCLUDE-Spalten kann reinen Indexzugriff für den Datenabruf ermöglichen, was zu einer Verbesserung der Leistung führt.
- Wenn die Tabelle, für die ein Index erstellt wird, leer ist, wird der Index zwar erstellt, jedoch werden erst Indexeinträge erstellt, wenn die Tabelle geladen oder Zeilen eingefügt werden. Ist die Tabelle nicht leer, erstellt der Datenbankmanager die Indexeinträge während der Verarbeitung der Anweisung CREATE INDEX.

- Bei einem *Clusterindex* versucht der Datenbankmanager, die neuen Zeilen für die Tabelle physisch nahe bei vorhandenen Zeilen mit ähnlichen Schlüsselwerten (wie durch den Index definiert) einzufügen.
- Wenn Sie einen *Primärschlüsselindex* als Clusterindex verwenden wollen, sollte in der Anweisung CREATE TABLE kein Primärschlüssel angegeben werden. Wenn der Primärschlüssel einmal erstellt ist, kann der zugehörige Index nicht geändert werden. Setzen Sie stattdessen die Anweisung CREATE TABLE ohne Primärschlüsselklausel (PRIMARY KEY) ab. Führen Sie anschließend die Anweisung CREATE INDEX aus, in der Sie die Clustering-Attribute angeben. Verwenden Sie schließlich die Anweisung ALTER TABLE, um einen Primärschlüssel hinzuzufügen, der dem gerade erstellten Index entspricht. Dieser Index wird dann als Primärschlüsselindex verwendet.
- Indizes belegen Plattenspeicherplatz. Die Menge an Plattenspeicherplatz variiert in Abhängigkeit von der Länge der Schlüsselspalten und der Anzahl der indexierten Zeilen. Die Größe des Index nimmt zu, je mehr Daten in die Tabelle eingefügt werden. Aus diesem Grund müssen Sie den Umfang der Daten, die indexiert werden, bei der Planung der Größe der Datenbank berücksichtigen. Im Folgenden sind einige der Aspekte für die Dimensionierung des Index aufgeführt:
  - Integritätsbedingungen über Primärschlüssel und eindeutige Schlüssel führen immer zur Erstellung eines systemgenerierten eindeutigen Index.
  - Die Erstellung einer MDC-Tabelle führt ebenfalls zur Generierung eines Blockindex durch das System.
  - XML-Spalten verursachen immer eine Erstellung systemgenerierter Indizes.
  - Es ist in der Regel von Vorteil, Indizes für Spalten mit Integritätsbedingungen über Fremdschlüssel zu erstellen.

  Anmerkung: Die maximale Anzahl der Spalten in einem Index beträgt 64. Allerdings beträgt die maximale Anzahl an Spalten in einem Index beim Indexieren einer typisierten Tabelle 63. Die maximale Länge eines Indexschlüssels einschließlich des gesamten Systemaufwands beträgt *indexpagesize*/4. Die maximal zulässige Anzahl an Indizes in einer Tabelle beträgt 32.767. Die maximale Länge eines Indexschlüssels darf die für die Seitengröße geltende Längenbegrenzung für Indexschlüssel nicht überschreiten. Informationen zu gespeicherten Längen finden Sie im Abschnitt zur „Anweisung CREATE TABLE". Informationen zu Schlüssellängenbegrenzungen finden Sie im Abschnitt „SQL- und XQuery-Begrenzungen".

  Anmerkung:

Tools für das Entwerfen von Indizes

Sobald Sie Ihre Tabellen erstellt haben, müssen Sie erörtern, wie schnell der Datenbankmanager Daten aus den Tabellen abrufen kann. Sie können den Designadvisor oder den Befehl db2advis zur Unterstützung beim Entwerfen Ihrer Indizes verwenden.

Das Erstellen nützlicher Indizes für Ihre Tabellen kann zur deutlichen Verbesserung der Abfrageleistung beitragen. Ähnlich wie bei Buchindizes können mithilfe von Tabellenindizes bestimmte Informationen mit einem minimalen Suchaufwand schnell gefunden werden. Die Verwendung eines Index zum Abrufen bestimmter Zeilen aus einer Tabelle kann die Anzahl aufwandsintensiver Ein-/Ausgabeoperationen reduzieren, die der Datenbankmanager durchführen muss. Der Grund hierfür ist, dass es ein Index dem Datenbankmanager ermöglicht, durch

Lesen einer relativ geringen Anzahl von Datenseiten nach einer Zeile zu suchen; er muss keine aufwandsintensive Suche für sämtliche Datenseiten betreiben, bis alle Übereinstimmungen gefunden sind.

Der Designadvisor von DB2 ist ein Tool, das Ihnen helfen kann, die Auslastungsleistung Ihres Systems erheblich zu verbessern. Die Aufgabe, zu entscheiden, welche Indizes, MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle), Clusteringdimensionen oder Datenbankpartitionen für eine komplexe Auslastung zu erstellen sind, kann sich als äußerst schwierig gestalten. Der Designadvisor ermittelt alle Objekte, die zur Verbesserung der Leistung Ihrer Auslastung erforderlich sind. Für einen gegebenen Satz von SQL-Anweisungen in einer Auslastung generiert der Designadvisor Empfehlungen für folgende Objekte und Maßnahmen:

- Neue Indizes
- Neue gespeicherte MQTs
- Umwandlung in MDC-Tabellen (mit mehrdimensionalem Clustering)
- Umverteilen von Tabellen
- Löschen von Indizes und MQTs, die durch die angegebene Auslastung nicht genutzt werden (über das GUI-Tool)

Über den Designadvisor können Sie einige oder alle dieser Empfehlungen unverzüglich implementieren oder ihre Implementierung für einen späteren Zeitpunkt terminieren.

Über die grafische Benutzerschnittstelle (GUI) des Designadvisors bzw. das Befehlszeilentool können Sie den Designadvisor zur Vereinfachung der folgenden Aufgaben einsetzen:

- Planen oder Einrichten einer neuen Datenbank
- Optimieren der Auslastungsleistung

Speicherbedarf für Indizes

Beim Entwerfen von Indizes müssen Sie sich mit dem anfallenden Speicherplatzbedarf vertraut machen.

Für jeden Index kann der erforderliche Speicherbereich wie folgt abgeschätzt werden:

```
(durchschnittliche Indexschlüsselgröße + Indexschlüsselaufwand) * Anzahl der Zeilen * 2
```

Dabei gilt Folgendes:

- Die „durchschnittliche Indexschlüsselgröße" ist die Bytezahl jeder Spalte im Indexschlüssel. (Verwenden Sie bei der Abschätzung der durchschnittlichen Spaltengröße für VARCHAR- und VARGRAPHIC-Spalten einen Durchschnittswert der aktuellen Datengröße plus zwei Byte. Verwenden Sie nicht die deklarierte Maximalgröße.)
- Der „Indexschlüsselaufwand" hängt vom Typ der Tabelle ab, für die der Index erstellt wird. Für große Tabellen (mit oder ohne XML-Indizes) gilt der Wert 11, sofern die Tabelle nicht partitioniert ist. Für partitionierte Tabellen ist dieser Wert 13. Für reguläre Tabellen beträgt dieser Wert 9 ohne XML-Indizes und 11 mit XML-Indizes. Für alle regulären Tabellen, die partitioniert sind, gilt der Wert 11.
- Der Faktor „2" ist für den Systemaufwand, z. B. für Nichtblattseiten und freien Speicherbereich.

Anmerkung:

1. Fügen Sie für jede Spalte, die Nullwerte zulässt, ein zusätzliches Byte für den Nullanzeiger hinzu.
2. Für Blockindizes, die intern für MDC-Tabellen (MDC - Multidimensional Clustering) erstellt werden, ist die „Anzahl der Zeilen" durch die „Anzahl der Blöcke" zu ersetzen.

Für jeden Index für eine XML-Spalte kann der erforderliche Speicherbereich wie folgt abgeschätzt werden:

```
(durchschnittliche Indexschlüsselgröße + Indexschlüsselaufwand) * Anzahl
der indexierten Knoten * 2
```

Dabei gilt Folgendes:

- Die „durchschnittliche Indexschlüsselgröße" ist die Summe der Schlüsselkomponenten, die den Index bilden. Der XML-Index setzt sich aus verschiedenen XML-Schlüsselkomponenten und einem Wert (SQL-Datentyp: sql-data-type) zusammen:

  ```
  Fester Aufwand + variabler Aufwand + Bytezahl des SQL-Datentyps
  ```

  Dabei gilt Folgendes:
  - Der „feste Aufwand" beträgt 14 Byte.
  - Der „variable Aufwand" ist die durchschnittliche Tiefe des indexierten Knotens plus 4 Byte.
  - Die Bytezahl des Werts für den SQL-Datentyp (sql-data-type) folgt den gleichen Regeln wie SQL.
- Die „Anzahl indexierter Knoten" ist die Anzahl der einzufügenden Dokumente multipliziert mit der Anzahl von Knoten in einem Musterdokument, die dem XML-Musterausdruck (XMLPATTERN) in der Indexdefinition entsprechen.

Indizes, die vor Version 8 erstellt wurden (Indizes des Typs 1), unterscheiden sich von denen in Version 8 bzw. einer der folgenden Versionen erstellten (Indizes des Typs 2). Zur Ermittlung des Typs von Index, der für eine Tabelle vorhanden ist, verwenden Sie die Tabellenfunktion ADMIN_GET_TAB_INFO. Zur Umwandlung von Indizes des Typs 1 in Indizes des Typs 2 verwenden Sie den Befehl REORG INDEXES .... CONVERT.

Stellen Sie bei der Verwendung des Befehls REORG INDEXES sicher, dass Sie über genügend freien Speicherbereich in dem Tabellenbereich verfügen, in dem die Indizes gespeichert werden. Die Größe des freien Speicherbereichs muss gleich der aktuellen Größe des Index sein. Zusätzlicher Speicherplatz wird eventuell benötigt, wenn Sie die Indizes mit der Option ALLOW WRITE ACCESS reorganisieren. Der zusätzliche Speicherbedarf fällt für die Protokolle der Aktivitäten an, die sich während der Reorganisation der Indizes auf die Indizes auswirken.

Bei der Indexerstellung ist temporärer Speicherplatz erforderlich. Der maximal während der Indexerstellung erforderliche temporäre Speicherplatz kann folgendermaßen abgeschätzt werden:

```
(durchschnittliche Indexschlüsselgröße + Indexschlüsselaufwand) * Anzahl
der Zeilen * 3,2
```

oder

```
(durchschnittliche Indexschlüsselgröße + Indexschlüsselaufwand) * Anzahl
der indexierten Knoten * 3,2
```

Dabei wird der Faktor „3,2" für den Indexaufwand sowie für den Speicherbereich einkalkuliert, der für Sortierungen während der Indexerstellung erforderlich ist.

Anmerkung: Für nicht eindeutige Indizes werden zum Speichern doppelter Schlüsseleinträge nur fünf Byte benötigt. Die oben gezeigte Berechnung geht von eindeutigen Indizes aus. Der zum Speichern eines Index erforderliche Speicherbereich kann durch die oben gezeigte Formel eventuell zu groß abgeschätzt werden.

Wenn die Anzahl der Indexknoten ein Datenvolumen von 64 KB überschreitet, wird beim Einfügen ein temporärer Speicherbereich benötigt. Die Größe des temporären Speicherbereichs lässt sich wie folgt abschätzen:

```
(durchschnittliche Indexschlüsselgröße) * Anzahl der indexierten Knoten * 1,2
```

Die beiden folgenden Formeln können zum Abschätzen der Anzahl von Schlüsseln pro Blattseite eines Index verwendet werden. (Die zweite Formel liefert einen etwas genaueren Schätzwert.) Die Genauigkeit dieser Schätzwerte hängt im Wesentlichen davon ab, wie gut die verwendeten Durchschnittswerte die tatsächlichen Daten widerspiegeln.

Anmerkung: Für SMS-Tabellenbereiche beträgt der minimale Speicherbedarf für Blattseiten 12 KB. Für DMS-Tabellenbereiche entspricht der minimale Speicherbedarf einem durch den Wert von EXTENTSIZE definierten Speicherbereich.

- Die Durchschnittszahl von Schlüsseln pro Blattseite kann mit folgender Formel grob abgeschätzt werden:

  $$\frac{(0{,}9 * (U - (M*2))) * (D + 1)}{K + 7 + (5 * D)}$$

  Dabei gilt Folgendes:
  - U, der verwendbare Speicherbereich auf einer Seite, entspricht ungefähr der Seitengröße minus 100. Bei einer Seitengröße von 4096 ist U gleich 3996.
  - M = U / (9 + *minimale Schlüsselgröße*)
  - D = durchschnittliche Anzahl mehrfacher Werte pro Schlüsselwert
  - K = *durchschnittliche Schlüsselgröße*

  Beachten Sie, dass die Werte für *minimale Schlüsselgröße* und *durchschnittliche Schlüsselgröße* ein Byte extra für jeden Teil des Schlüssels haben müssen, der einen Nullwert enthalten kann, und zusätzliche zwei Byte für die Länge jedes Teils des Schlüssels mit variabler Länge.

  Wenn INCLUDE-Spalten vorhanden sind, müssen sie in den Werten für *minimale Schlüsselgröße* und *durchschnittliche Schlüsselgröße* berücksichtigt werden.

  Die „minimale Schlüsselgröße" ist die Summe der Schlüsselkomponenten, die den Index bilden:

  ```
  Fester Aufwand + variabler Aufwand + Bytezahl des SQL-Datentyps
  ```

  Dabei gilt Folgendes:
  - Der „feste Aufwand" beträgt 13 Byte.
  - Der „variable Aufwand" ist die minimale Tiefe des indexierten Knotens plus 4 Byte.
  - Die Bytezahl des Werts für den SQL-Datentyp (sql-data-type) folgt den gleichen Regeln wie SQL.

Der Wert `0,9` kann durch einen beliebigen, mit (100 - pctfree)/100 berechneten Wert ersetzt werden, wenn während der Indexerstellung ein anderer Prozentwert für freien Speicherbereich (pctfree) angegeben wird als der Standardwert 10.

- Die Durchschnittszahl von Schlüsseln pro Blattseite kann mit folgender Formel etwas genauer abgeschätzt werden:

  ```
  L = Blattseitenzahl = X / (Durchschnittszahl von Schlüsseln pro Blattseite)
  ```

  Dabei ist X die Gesamtzahl der Zeilen in der Tabelle.

  Für den Index für eine XML-Spalte, ist X die Gesamtzahl der indexierten Knoten in der Spalte.

  Einen Schätzwert für die Originalgröße eines Indexes können Sie wie folgt berechnen:

  ```
  (L + 2L/(Durchschnittszahl von Schlüsseln pro Blattseite)) * Seitengröße
  ```

  Für DMS-Tabellenbereiche addieren Sie die Größen aller Indizes einer Tabelle und runden auf ein Vielfaches des Werts für EXTENTSIZE für den Tabellenbereich auf, in dem sich der Index befindet.

  Sie sollten weiteren Speicherbereich für das Anwachsen des Index durch INSERT/UPDATE-Vorgänge bereitstellen, die zur Teilung von Seiten führen können.

  Durch die folgende Berechnung erhalten Sie einen genaueren Schätzwert für die Originalgröße des Indexes sowie einen Schätzwert für die Anzahl der Indexstufen. (Dies ist möglicherweise von besonderem Interesse, wenn in der Indexdefinition INCLUDE-Spalten verwendet werden.) Die Durchschnittszahl von Schlüsseln pro Nichtblattseite kann mit folgender Formel grob abgeschätzt werden:

  ```
  (0,9 * (U - (M*2))) * (D + 1)
  ----------------------------
        K + 13 + (9 * D)
  ```

  Dabei gilt Folgendes:

  – U, der verwendbare Speicherbereich auf einer Seite, entspricht ungefähr der Seitengröße minus 100. Bei einer Seitengröße von 4096 ist U gleich 3996.
  – D ist die durchschnittliche Anzahl mehrfacher Werte pro Schlüsselwert auf Nichtblattseiten (dieser Wert ist deutlich kleiner als für Blattseiten. Sie können ihn zur Vereinfachung der Berechnung auf 0 setzen).
  – M = U / (9 + *minimale Schlüsselgröße* für Nichtblattseiten)
  – K = *durchschnittliche Schlüsselgröße* für Nichtblattseiten

  Die *minimale Schlüsselgröße* und die *durchschnittliche Schlüsselgröße* für Nichtblattseiten stimmen mit den Werten für Blattseiten überein, sofern keine INCLUDE-Spalten vorkommen. INCLUDE-Spalten werden auf Nichtblattseiten nicht gespeichert.

  Sie sollten `0,9` nur durch `(100 - pctfree)/100` ersetzen, wenn dieser Wert größer als `0,9` ist, weil auf Nichtblattseiten bei der Indexerstellung maximal 10 % Speicherbereich frei bleibt.

  Die Zahl der Nichtblattseiten kann mit folgender Formel abgeschätzt werden:

  ```
  if L > 1 then {P++; Z++}
  While (Y > 1)
  {
     P = P + Y
     Y = Y / N
    Z++
  }
  ```

Dabei gilt Folgendes:

– P ist die Anzahl von Seiten (zunächst 0).
– L ist die Anzahl der Blattseiten.
– N ist die Anzahl von Schlüsseln pro Nichtblattseite.
– Y = L / N
– Z ist die Anzahl der Ebenen in der Indexbaumstruktur (zunächst 1).

Als Gesamtzahl der Seiten ergibt sich:

```
T = (L + P + 2) * 1,0002
```

Die zusätzlichen 0,02 % sind für Systemaufwand (einschließlich Speicherzuordnungsseiten).

Der für die Indexerstellung erforderliche Speicherbereich lässt sich folgendermaßen abschätzen:

```
T * Seitengröße
```

Erstellen von Indizes

Indizes können erstellt werden, um die Werte in einer Spalte aufsteigend oder absteigend anzuordnen. Sie können die Anweisung CREATE INDEX, den DB2-Designadvisor oder den Designadvisorbefehl db2advis verwenden, um die Indizes zu erstellen.

Zum Erstellen eines Index mit der Anweisung CREATE INDEX geben Sie in die Befehlszeile folgende Anweisung ein:

```
CREATE UNIQUE INDEX EMP_IX
  ON EMPLOYEE(EMPNO)
  INCLUDE(FIRSTNAME, JOB)
```

Die Klausel INCLUDE, die nur für eindeutige Indizes verwendbar ist, gibt zusätzliche Spalten an, die an die Gruppe der Indexschlüsselspalten angehängt werden. Alle Spalten, die mit dieser Klausel in den Index eingeschlossen werden, werden nicht zur Umsetzung der Eindeutigkeit verwendet. Solche INCLUDE-Spalten können die Leistung einiger Abfragen durch die Möglichkeit eines reinen Indexzugriffs verbessern. Diese Option bietet folgende Vorteile:

- Sie hilft bei der Vermeidung ansonsten notwendiger Zugriffe auf die Datenseiten für weitere Abfragen.
- Sie hilft bei der Vermeidung redundanter Indizes.

Wenn die Anweisung SELECT EMPNO, FIRSTNAME, JOB FROM EMPLOYEE an der Tabelle, für die dieser Index definiert ist, ausgeführt wird, können alle angeforderten Daten aus dem Index abgerufen werden, ohne dass die Datenseiten gelesen werden müssen. Dies verbessert die Leistung.

Anmerkung: Bei Indizes, die in Version 8 oder einer späteren Version erstellt werden (Indizes des Typs 2), werden Schlüssel lediglich als gelöscht markiert, wenn eine Zeile gelöscht oder aktualisiert wird. Solche Schlüssel werden als pseudogelöscht bezeichnet. Sie werden von einer Seite erst dann physisch entfernt, wenn eine Bereinigung durchgeführt wird, nachdem die Löschung oder Aktualisierung mit COMMIT festgeschrieben wurde. Eine solche Bereinigung kann zum Beispiel durch eine nachfolgende Transaktion durchgeführt werden, welche die Seite ändert, auf der ein Schlüssel als gelöscht markiert ist. Die Bereinigung pseudogelöschter Schlüssel kann explizit durch die Option CLEANUP ONLY ALL des Dienstprogramms REORG INDEXES ausgelöst werden.

Indizes für Tabellen in einer Umgebungen mit partitionierten Datenbanken werden mithilfe der gleichen Anweisung CREATE INDEX erstellt. Die Daten in den Indizes werden auf der Basis des Verteilungsschlüssels der Tabelle verteilt. Dazu wird eine B+-Baumstruktur in jeder Datenbankpartition in der Datenbankpartitionsgruppe erstellt. Jede B+-Baumstruktur indexiert den Teil der Tabelle, der zur eigenen Datenbankpartition gehört. Spalten in einem eindeutigen Index, der für eine Mehrpartitionsdatenbank definiert ist, müssen eine Obermenge der Spalten im Verteilungsschlüssel sein.

Anmerkung: In Version 9.5 blockiert die Ausführung der Anweisung CREATE INDEX auf Solaris-Plattformen, wenn eine Roheinheit verwendet wird. Sun wird dieses Problem beheben und eine Programmkorrektur in einem Kernel-Patch bereitstellen.

Ändern von Indizes

Wenn Sie einen Index ändern möchten, müssen Sie den Index zunächst löschen und anschließend erneut erstellen. Eine Anweisung ALTER INDEX ist nicht verfügbar.

Es ist zum Beispiel nicht möglich, der Liste der Schlüsselspalten eine Spalte hinzuzufügen, ohne die vorherige Definition zu löschen und einen neuen Index zu erstellen. Sie können jedoch einen Kommentar mithilfe der Anweisung COMMENT hinzufügen, um den Zweck des Index zu beschreiben.

Umbenennen von Indizes

Mithilfe der Anweisung RENAME können Sie einen vorhandenen Index umbenennen.

Zum Umbenennen eines vorhandenen Index geben Sie die folgende Anweisung in die Befehlszeile ein:

```
RENAME INDEX <quellenindexname> TO <zielindexname>
```

- Die Angabe <quellenindexname> ist der Name des vorhandenen Index, der umbenannt werden soll. Der Name, einschließlich des Schemanamens, muss einen Index angeben, der bereits in der Datenbank vorhanden ist. Dabei darf es sich nicht um den Namen eines Index für eine deklarierte globale temporäre Tabelle handeln. Der Schemaname darf nicht SYSIBM, SYSCAT, SYSFUN oder SYSSTAT sein.
- Die Angabe <zielindexname> enthält den neuen Namen für den Index ohne Schemanamen. Der Schemaname des Quellenobjekts wird zur Qualifikation des neuen Namens für das Objekt verwendet. Der qualifizierte Name darf keinen Index bezeichnen, der bereits in der Datenbank vorhanden ist.

Beim Umbenennen eines Index darf der Quellenindex kein systemgenerierter Index sein. Wenn die Anweisung erfolgreich ausgeführt wird, werden die Systemkatalogtabellen mit dem neuen Indexnamen aktualisiert.

Erneutes Erstellen von Indizes

Bestimmte Datenbankoperationen, wie zum Beispiel eine aktualisierende Recovery (Rollforward), die eine Indexerstellungsoperation (CREATE INDEX) umfasst, die nicht vollständig protokolliert wurde, können dazu führen, dass ein Indexobjekt ungültig wird, weil der Index während der aktualisierenden Recovery nicht erstellt wird. Das Indexobjekt kann durch erneutes Erstellen des enthaltenen Index wiederhergestellt werden.

Wenn der Datenbankmanager erkennt, dass ein Index nicht mehr gültig ist, versucht er automatisch, diesen Index erneut zu erstellen. Wann die erneute Erstellung stattfindet, wird durch den Parameter **indexrec** in der Konfigurationsdatei der Datenbank oder des Datenbankmanagers gesteuert. Für diesen Parameter sind fünf Einstellungen möglich:

- SYSTEM
- RESTART
- RESTART_NO_REDO
- ACCESS
- ACCESS_NO_REDO

Die Werte RESTART_NO_REDO und ACCESS_NO_REDO haben eine ähnliche Bedeutung wie RESTART und ACCESS.

Die NO_REDO-Optionen legen fest, dass, auch wenn der Index während der ursprünglichen Operation (z. B. CREATE INDEX) vollständig protokolliert wurde, der Index bei der aktualisierende Recovery nicht erneut erstellt wird. Stattdessen wird er entweder beim Neustart oder beim ersten Zugriff erstellt. Weitere Informationen finden Sie in der Beschreibung des Parameters **indexrec**.

Wenn der Zeitpunkt des Datenbankneustarts kein Problem darstellt, ist es besser, ungültige Indizes im Rahmen des Prozesses zur Wiederherstellung der Datenbank in einem konsistenten Status erneut zu erstellen. Bei dieser Vorgehensweise dauert das erneute Starten einer Datenbank aufgrund des erneuten Erstellungsprozesses für den Index länger. Jedoch wird der normale Verarbeitungsbetrieb nicht mehr gestört, wenn sich die Datenbank wieder in einem konsistenten Status befindet.

Wenn Indizes andererseits beim nächsten Zugriff erneut erstellt werden, ist die Zeit, die für den Neustart einer Datenbank benötigt wird, zwar kürzer, jedoch kann es anschließend infolge eines Index, der erneut erstellt wird, zu einer unerwarteten Verschlechterung der Antwortzeit kommen. Zum Beispiel müssen Benutzer, die auf eine Tabelle zugreifen, die einen ungültigen Index hat, warten, bis der Index erneut erstellt wurde. Darüber hinaus ist es möglich, dass unerwartete Sperren aktiviert und bis lange nach der Neuerstellung eines ungültigen Index beibehalten werden, insbesondere wenn die Transaktion, die die Neuerstellung des Index verursacht hat, nie beendet wird (d. h. die durch sie vorgenommenen Änderungen festschreibt (COMMIT) oder rückgängig macht (ROLLBACK)).

Löschen von Indizes

Sie können keine Klausel einer Indexdefinition ändern. Sie müssen den Index löschen und erneut erstellen. (Das Löschen eines Index führt nicht dazu, dass andere Objekte gelöscht werden. Allerdings werden möglicherweise einige Pakete ungültig gemacht.) Verwenden Sie die Anweisung DROP zum Löschen von Indizes.

Ein Index für den Primärschlüssel oder eindeutigen Schlüssel kann nicht explizit gelöscht werden. Zum Löschen eines solchen Index gibt es folgende Methoden:

- Wenn der Primärindex bzw. die eindeutige Integritätsbedingung automatisch für den Primärschlüssel oder eindeutigen Schlüssel erstellt wurde, wird der Index durch Löschen des Primärschlüssels oder eindeutigen Schlüssels gelöscht. Das Löschen des Schlüssels erfolgt mithilfe der Anweisung ALTER TABLE.
- Wenn der Primärindex oder die eindeutige Integritätsbedingung benutzerdefiniert ist, muss der Primärschlüssel oder eindeutige Schlüssel zuerst mit der Anweisung ALTER TABLE gelöscht werden. Nach dem Löschen des Primärschlüssels oder des eindeutigen Schlüssels wird der Index nicht länger als Primärindex oder eindeutiger Index betrachtet und kann daher explizit gelöscht werden.

Geben Sie zum Löschen eines Index die folgende Anweisung in die Befehlszeile ein:

```
DROP INDEX <indexname>
```

Mit der folgenden Anweisung wird der Index PH gelöscht:

```
DROP INDEX PH
```

Pakete und zwischengespeicherte dynamische SQL- und XQuery-Anweisungen, die von den gelöschten Indizes abhängig sind, werden als ungültig markiert. Das Anwendungsprogramm ist von Änderungen durch das Hinzufügen oder Löschen von Indizes nicht betroffen.

Kapitel 14. Trigger

Ein *Trigger* definiert eine Reihe von Aktionen, die in Reaktion auf eine Einfüge-, Aktualisierungs- oder Löschoperation für eine angegebene Tabelle ausgeführt werden. Wenn eine solche SQL-Operation ausgeführt wird, wird der Trigger als *aktiviert* bezeichnet. Trigger sind optional und werden mithilfe der Anweisung CREATE TRIGGER definiert.

Trigger können zusammen mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen in Tabellen zur Implementierung von Datenintegritätsregeln eingesetzt werden. Darüber hinaus können Trigger auch genutzt werden, um Aktualisierungen an anderen Tabellen zu bewirken, automatisch Werte für eingefügte oder aktualisierte Zeilen zu generieren bzw. umzuwandeln oder Funktion für solche Operationen wie das Absetzen von Alerts aufzurufen.

Trigger bilden einen nützlichen Mechanismus zur Definition und Implementierung von *Übergangsregeln* für den Geschäftsbetrieb, bei denen es sich um Regeln handelt, die für verschiedene Status der Daten gelten (z. B. ein Gehalt, dass nicht um mehr als zehn Prozent angehoben werden kann).

Mithilfe von Triggern lässt sich die Logik, durch die Geschäftsregeln implementiert werden, außerhalb der Datenbank verwalten. Das heißt, dass nicht die Anwendungen für die Implementierung dieser Regeln zuständig sind. Eine zentralisierte Logik, die für alle Tabellen implementiert wird, ermöglicht eine einfachere Pflege, da keine Änderungen an Anwendungsprogrammen erforderlich werden, wenn sich die Logik ändert.

Zur Erstellung eines Triggers sind folgende Angaben erforderlich:

- Die *Subjekttabelle* gibt die Tabelle an, für die der Trigger definiert wird.
- Das *Trigger-Ereignis* definiert eine bestimmte SQL-Operation, die die Subjekttabelle modifiziert. Bei dem Ereignis kann es sich um eine Einfüge-, Aktualisierungs- oder Löschoperation handeln.
- Die *Aktivierungszeit des Triggers* gibt an, ob der Trigger vor oder nach dem Eintreten des Triggerereignisses zu aktivieren ist.

Die Anweisung, welche die Aktivierung eines Triggers bewirkt, enthält eine *Gruppe der betroffenen Zeilen*. Dies sind die Zeilen der Subjekttabelle, die eingefügt, aktualisiert oder gelöscht werden. Die *Triggergranularität* gibt an, ob die Aktionen des Triggers einmal für die Anweisung oder einmal für jede der betroffenen Zeilen ausgeführt werden.

Die *ausgelöste Aktion* besteht aus einer optionalen Suchbedingung und einer Gruppe von Anweisungen, die ausgeführt werden, wenn der Trigger aktiviert wird. Die Anweisungen werden nur ausgeführt, wenn die Suchbedingung ein wahres Ergebnis liefert. Wenn die Aktivierungszeit des Triggers vor dem Trigger-Ereignis liegt, können die ausgelösten Aktionen Anweisungen umfassen, mit denen Auswahlen getroffen, Übergangsvariablen festgelegt oder SQL-Status signalisiert werden können. Liegt die Aktivierungszeit des Triggers nach dem Trigger-Ereignis, können die ausgelösten Aktionen Anweisungen umfassen, die auswählen (SELECT), aktualisieren (UPDATE), löschen (DELETE) oder SQL-Status signalisieren.

Die ausgelöste Aktion kann sich mithilfe von *Übergangsvariablen* auf die Werte in der Gruppe der betroffenen Zeilen beziehen. Übergangsvariablen verwenden die Namen der Spalten in der Subjekttabelle, die durch einen angegebenen Namen qualifiziert werden, der zu erkennen gibt, ob es sich um einen Verweis auf den alten Wert (vor der Aktualisierung) oder den neuen Wert (nach der Aktualisierung) handelt. Der neue Wert kann mithilfe der Anweisung 'SET Variable' in BEFORE-, INSERT und UPDATE-Triggern ebenfalls geändert werden.

Eine weitere Methode zur Bezugnahme auf die Werte in der Gruppe der betroffenen Zeilen besteht in der Verwendung von *Übergangstabellen*. Übergangstabellen können ebenfalls die Namen von Spalten in der Subjekttabelle verwenden, stellen jedoch einen Namen bereit, über den die gesamte Gruppe der betroffenen Zeilen wie eine Tabelle behandelt werden kann. Übergangstabellen können nur in AFTER-Triggern (d. h., nicht mit BEFORE- und INSTEAD OF-Triggern) verwendet werden, wobei für alte und neue Werte getrennte Übergangstabellen definiert werden können.

Es können mehrere Trigger angegeben werden, um Tabellen, Ereignisse (INSERT, UPDATE, DELETE, INSTEAD OF) oder Aktivierungszeiten (BEFORE, AFTER) zu kombinieren. Wenn mehr als ein Trigger für eine bestimmte Tabelle, ein bestimmtes Ereignis und eine bestimmte Aktivierungszeit vorhanden ist, entspricht die Reihenfolge, in der die Trigger aktiviert werden, der Reihenfolge, in der sie erstellt werden. Das heißt, der zuletzt erstellte Trigger wird auch zuletzt aktiviert.

Die Aktivierung eines Triggers kann unter Umständen zu einem *Hintereinanderschalten von Triggern* führen. Diese Bezeichnung bezieht sich auf das Ergebnis der Aktivierung eines Triggers, der Anweisungen ausführt, die wiederum die Aktivierung anderer Trigger oder sogar erneut desselben Triggers bewirken. Die ausgelösten Aktionen können außerdem Aktualisierungen bewirken, die sich aus der Anwendung von Regeln der referenziellen Integrität für Löschungen ergeben, die wiederum die Aktivierung weiterer Trigger nach sich ziehen können. Durch die Hintereinanderschaltung von Triggern kann eine Kette von Triggern und Löschregeln der referenziellen Integrität aktiviert werden, die einen erheblichen Umfang an Änderungen an der Datenbank als Folge einer einzigen INSERT-, UPDATE- oder DELETE-Anweisung bewirken.

Wenn mehrere Trigger Einfüge-, Aktualisierungs- oder Löschaktionen für dasselbe Objekt enthalten, werden zur Lösung von Zugriffskonflikten Konfliktlösungsmeachnismen wie z. B. temporäre Tabellen verwendet. Dies kann sich merklich auf die Leistung auswirken, insbesondere in Umgebungen mit partitionierten Datenbanken.

Typen von Triggern

Ein *Trigger* definiert eine Reihe von Aktionen, die in Reaktion auf eine Einfüge-, Aktualisierungs- oder Löschoperation für eine angegebene Tabelle ausgeführt werden. Wenn eine solche SQL-Operation ausgeführt wird, wird der Trigger als *aktiviert* bezeichnet. Trigger sind optional und werden mithilfe der Anweisung CREATE TRIGGER definiert.

Trigger können zusammen mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen in Tabellen zur Implementierung von Datenintegritätsregeln eingesetzt werden. Darüber hinaus können Trigger auch genutzt werden, um Aktualisierungen an anderen Tabellen zu bewirken, automatisch Werte für eingefügte oder aktualisierte Zeilen zu generieren bzw. umzuwandeln oder Funktion für solche Operationen wie das Absetzen von Alerts aufzurufen.

Folgende Typen von Triggern werden unterstützt:

BEFORE-Trigger
Diese werden vor einer Aktualisierungs- oder Einfügeoperation ausgeführt. Werte, die aktualisiert oder eingefügt wurden, können geändert werden, bevor die Datenbank tatsächlich geändert wird. Sie können Trigger, die vor einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, auf verschiedene Weise verwenden:

- Zum Prüfen oder Ändern von Werten, bevor sie tatsächlich aktualisiert oder in die Datenbank eingefügt werden. Diese Art der Verwendung ist nützlich, wenn Sie Daten, so, wie sie der Benutzer sieht, in ein beliebiges internes Datenbankformat umsetzen müssen.
- Zum Ausführen von anderen Operationen als Datenbankoperationen, die in benutzerdefinierten Funktionen codiert sind.

BEFORE DELETE-Trigger
Diese werden vor einer Löschoperation ausgeführt. Sie dienen zum Überprüfen von Werten (und, falls erforderlich, zum Ausgeben von Fehlern).

AFTER-Trigger
Diese werden nach einer Aktualisierungs-, Einfüge- oder Löschoperation ausgeführt. Sie können Trigger verwenden, die nach einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, auf verschiedene Weise verwenden:

- Zum Aktualisieren von Daten in anderen Tabellen. Diese Funktion ist für die Pflege von Beziehungen zwischen Daten oder für die Aufbewahrung von Prüfprotokollinformationen nützlich.
- Zum Abgleichen gegen andere Daten in der Tabelle oder in anderen Tabellen. Diese Funktion ist für die Sicherstellung der Datenintegrität nützlich, wenn referenzielle Integritätsbedingungen nicht geeignet sind oder wenn Prüfungen auf Integritätsbedingung in Tabellen nur auf die Prüfung der aktuellen Tabelle beschränkt werden.
- Zum Ausführen von anderen Operationen als Datenbankoperationen, die in benutzerdefinierten Funktionen codiert sind. Diese Funktion ist nützlich, wenn Alerts abgesetzt werden; sie ist auch für die Aktualisierung von Informationen außerhalb der Datenbank hilfreich.

INSTEAD OF-Trigger
Zum Beschreiben der Vorgehensweise beim Durchführen von Einfüge-, Aktualisierung- und Löschoperationen für Sichten, die zu komplex sind, als dass sie diese Operationen nativ unterstützen könnten. Mit diesen Triggern können Anwendungen eine Sicht als alleinige Schnittstelle für alle SQL-Operationen (INSERT, DELETE, UPDATE und SELECT) verwenden.

BEFORE-Trigger

Durch die Verwendung von Triggern, die vor einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, können Werte, die aktualisiert oder eingefügt wurden, geändert werden, bevor die Datenbank tatsächlich geändert wird. Falls dies gewünscht wird, können sie zur Umsetzung von Eingaben aus dem Anwendungsformat (Benutzersicht der Daten) in ein internes Datenbankformat verwendet werden.

Diese BEFORE-Trigger können auch zur Aktivierung anderer Operationen als Datenbankoperationen durch benutzerdefinierte Funktionen verwendet werden.

BEFORE-Trigger werden vor einer Löschoperation ausgeführt. Sie überprüfen die Werte und geben einen Fehler aus, falls erforderlich.

Beispiele

Im folgenden Beispiel wird ein BEFORE-Trigger zum Setzen eines komplexen Standardwerts definiert:

```
CREATE TRIGGER trigger1
  BEFORE UPDATE ON table1
  REFERENCING NEW AS N
  WHEN (N.expected_delivery_date IS NULL)
  SET N.expected_delivery_date = N.order_date + 5 days;
```

Im folgenden Beispiel wird ein BEFORE-Trigger mit einer tabellenübergreifenden Integritätsbedingung definiert, bei der es sich nicht um eine referenzielle Integritätsbedingung handelt:

```
CREATE TRIGGER trigger2
  BEFORE UPDATE ON table2
  REFERENCING NEW AS N
  WHEN (n.salary > (SELECT maxsalary FROM salaryguide WHERE rank = n.position))
  SIGNAL SQLSTATE '78000' SET MESSAGE_TEXT = 'Salary out of range');
```

AFTER-Trigger

Trigger, die nach einer Aktualisierungs-, Einfüge- oder Löschoperation ausgeführt werden, können auf verschiedene Weise verwendet werden.

- Trigger können Daten in ein und derselben Tabelle oder in anderen Tabellen aktualisieren, einfügen oder löschen. Dies ist nützlich, um Beziehungen zwischen Daten zu pflegen oder um Prüfprotokollinformationen aufzubewahren.
- Trigger können in der übrigen Tabelle oder in anderen Tabellen Daten auf Datenwerte hin überprüfen. Dies ist nützlich, wenn Sie aufgrund von Verweisen auf Daten anderer Zeilen dieser oder anderer Tabellen keine referenziellen Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen verwenden können.
- Trigger können benutzerdefinierte Funktionen zur Aktivierung von anderen Operationen als Datenbankoperationen verwenden. Dies ist beispielsweise beim Absetzen von Alerts oder beim Aktualisieren von Informationen außerhalb der Datenbank nützlich.

Beispiel

Im folgenden Beispiel wird ein AFTER-Trigger dargestellt, mit dem die Anzahl der Mitarbeiter vergrößert wird, wenn ein neuer Mitarbeiter eingestellt wird.

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

ININSTEAD OF-Trigger

INSTEAD OF-Trigger beschreiben die Vorgehensweise beim Durchführen von Einfüge-, Aktualisierung- und Löschoperationen für komplexe Sichten. Mit INSTEAD OF-Triggern können Anwendungen eine Sicht als alleinige Schnittstelle für alle SQL-Operationen (INSERT, DELETE, UPDATE und SELECT) verwenden.

In der Regel enthalten INSTEAD OF-Trigger die Umkehrung der in einem Hauptteil der Sicht angewendeten Logik. Beispiel: Betrachten Sie eine Sicht, bei der Spal-

ten der Quellentabelle entschlüsselt werden. Der INSTEAD OF-Trigger dieser Sicht verschlüsselt Daten und fügt sie anschließend in die Quellentabelle ein; so wird die symmetrische Operation durchgeführt.

Mit einem INSTEAD OF-Trigger wird die angeforderte Änderungsoperation für die Sicht durch die Triggerlogik ersetzt; diese führt die Operation im Auftrag der Sicht durch. Aus der Sicht der Anwendung geschieht dies transparent; sie erkennt, dass alle Operationen für die Sicht durchgeführt werden. Es ist nur ein einziger INSTEAD OF-Trigger für jede Art von Operation in einer angegebenen Themensicht zulässig.

Bei der Sicht selbst muss es sich um eine Sicht ohne Typ oder einen Aliasnamen handeln, der in eine Sicht ohne Typ aufgelöst wird. Außerdem darf es keine Sicht sein, die mit WITH CHECK OPTION (symmetrische Sicht) definiert ist; es darf auch keine Sicht sein, für die direkt oder indirekt eine symmetrische Sicht definiert wurde.

Beispiel

Im folgenden Beispiel werden drei INSTEAD OF-Trigger dargestellt, die Logik für INSERT-, UPDATE- und DELETE-Operationen für die definierte Sicht (EMPV) bereitstellen. Die Sicht EMPV enthält einen Join in ihrer Klausel FROM; deshalb ist eine native Unterstützung von Änderungsoperationen nicht möglich.

```
CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO,
                 HIREDATE, DEPTNAME)
AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO,
          HIREDATE, DEPTNAME
          FROM EMPLOYEE, DEPARTMENT WHERE
               EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO

CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
REFERENCING NEW AS NEWEMP FOR EACH ROW
INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME,
                      WORKDEPT, PHONENO, HIREDATE)
        VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
               COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                         WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                        RAISE_ERROR('70001', 'Unknown dept name')),
               PHONENO, HIREDATE)

CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
REFERENCING NEW AS NEWEMP OLD AS OLDEMP
  FOR EACH ROW
BEGIN ATOMIC
 VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
             ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
 UPDATE EMPLOYEE AS E
   SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE)
     = (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
               COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                         WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                        RAISE_ERROR ('70001', 'Unknown dept name')),
               NEWEMP.PHONENO, NEWEMP.HIREDATE)
 WHERE NEWEMP.EMPNO = E.EMPNO;
END

CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
REFERENCING OLD AS OLDEMP FOR EACH ROW
DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

Entwerfen von Triggern

Wenn Sie einen Trigger erstellen, müssen Sie ihn einer Tabelle zuordnen. Handelt es sich um einen INSTEAD OF-Trigger, müssen Sie ihn einer Sicht zuordnen. Diese Tabelle bzw. Sicht wird als *Zieltabelle* des Triggers bezeichnet. Der Begriff *Änderungsoperation* bezieht sich auf jede Änderung am Status der Zieltabelle.

Eine *Änderungsoperation* wird durch folgende Elemente eingeleitet:

- Anweisung INSERT
- Anweisung UPDATE bzw. referenzielle Integritätsbedingung, die eine UPDATE-Operation ausführt
- Anweisung DELETE bzw. referenzielle Integritätsbedingung, die eine DELETE-Operation ausführt
- Anweisung MERGE

Sie müssen jeden Trigger einem dieser drei Typen von Änderungsoperation zuordnen. Diese Zuordnung wird als *Trigger-Ereignis* für diesen bestimmten Trigger bezeichnet.

Sie müssen außerdem die Aktion definieren, die als *ausgelöste Aktion* bezeichnet wird und vom Trigger ausgeführt wird, wenn das Trigger-Ereignis eintritt. Die ausgelöste Aktion besteht aus einer oder mehreren Anweisungen, die entweder vor oder nach der Ausführung des Trigger-Ereignisses durch den Datenbankmanager ausgeführt werden können. Wenn ein Trigger-Ereignis eintritt, bestimmt der Datenbankmanager die Gruppe von Zeilen in der Subjekttabelle, die von der Änderungsoperation betroffen ist, und führt den Trigger aus.

Richtlinien zur Erstellung von Triggern:

Bei der Erstellung eines Triggers müssen die folgenden Attribute bzw. das folgende Verhalten deklariert werden:

- Der Name des Triggers
- Der Name der Subjekttabelle
- Die Aktivierungszeit des Triggers (vor (BEFORE) oder nach (AFTER) Ausführung der Änderungsoperation)
- Das Trigger-Ereignis (INSERT, DELETE oder UPDATE)
- Der alte Wert der Übergangsvariablen, falls vorhanden
- Der neue Wert der Übergangsvariablen, falls vorhanden
- Der alte Wert der Übergangstabelle, falls vorhanden
- Der neue Wert der Übergangstabelle, falls vorhanden
- Die Granularität (FOR EACH STATEMENT oder FOR EACH ROW)
- Die ausgelöste Aktion des Triggers (einschließlich einer Bedingung für die ausgelöste Aktion und ausgelöster Anweisung(en))
- Wenn das Trigger-Ereignis eine UPDATE-Operation ist: Eine Trigger-Spaltenliste, falls der Trigger nur aktiviert werden soll, wenn bestimmte Spalten in der Anweisung UPDATE angegeben werden

Entwerfen mehrerer Trigger:

Wenn Trigger mithilfe der Anweisung CREATE TRIGGER definiert werden, wird ihre Erstellungszeit in der Datenbank in Form einer Zeitmarke registriert. Der Wert dieser Zeitmarke wird nachfolgend dazu verwendet, die Reihenfolge der Aktivierung von Triggern zu bestimmen, wenn mehrere Trigger vorhanden sind, die zur gleichen Zeit ausgeführt werden sollen. Die Zeitmarke wird zum Beispiel verwendet, wenn mehr als ein Trigger

für dieselbe Subjekttabelle mit demselben Ereignis und derselben Aktivierungszeit vorhanden ist. Die Zeitmarke wird auch verwendet, wenn mindestens ein AFTER- oder INSTEAD OF-Trigger vorhanden ist, der durch das Trigger-Ereignis und durch Aktionen von referenziellen Integritätsbedingungen, die von der ausgelösten Aktion direkt oder indirekt verursacht werden, aktiviert wird (d. h. rekursiv durch andere referenzielle Integritätsbedingungen).

Betrachten Sie die folgenden beiden Trigger:

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  BEGIN ATOMIC
    UPDATE COMPANY_STATS
    SET NBEMP = NBEMP + 1;
  END

CREATE TRIGGER NEW_HIRED_DEPT
   AFTER INSERT ON EMPLOYEE
   REFERENCING NEW AS EMP
   FOR EACH ROW
     BEGIN ATOMIC
       UPDATE DEPTS
       SET NBEMP = NBEMP + 1
       WHERE DEPT_ID = EMP.DEPT_ID;
     END
```

Die oben gezeigten Trigger werden aktiviert, wenn Sie eine INSERT-Operation an der Tabelle EMPLOYEE ausführen. In diesem Fall definiert die Zeitmarke ihrer Erstellung, welcher der beiden Trigger zuerst aktiviert wird.

Die Aktivierung der Trigger erfolgt in aufsteigender Reihenfolge der Zeitmarkenwerte. Das heißt, ein Trigger, der einer Datenbank neu hinzugefügt wurde, wird nach allen anderen, zuvor definierten Triggern ausgeführt.

Alte Trigger werden vor neuen Triggern aktiviert, um sicherzustellen, dass neue Trigger als *inkrementelle* Zusätze zu Änderungen verwendet werden können, die die Datenbank betreffen. Wenn eine ausgelöste Anweisung von Trigger T1 zum Beispiel eine neue Zeile in Tabelle T einfügt, kann eine ausgelöste Anweisung von Trigger T2, der nach T1 ausgeführt wird, dazu verwendet werden, dieselbe Zeile in T mit bestimmten Werten zu aktualisieren. Da die Aktivierungsreihenfolge von Triggern vorhersagbar ist, können Sie mehrere Trigger für eine Tabelle definieren und trotzdem sicher sein, dass die neueren eine Tabelle bearbeiten, die bereits von den älteren geändert wurde.

Interaktionen von Triggern mit referenziellen Integritätsbedingungen:
Ein Trigger-Ereignis kann infolge von Änderungen eintreten, die auf die Umsetzung von referenziellen Integritätsbedingungen zurückzuführen sind. Betrachten Sie ein Beispiel mit den beiden Tabellen DEPT und EMP: Wenn Lösch- oder Aktualisierungsoperationen an DEPT die Weitergabe von Löschungen bzw. Aktualisierungen an EMP über referenzielle Integritätsbedingungen verursachen, werden für EMP bei DELETE- oder UPDATE-Operationen definierte Trigger infolge der referenziellen Integritätsbedingungen aktiviert, die für DEPT definiert sind. Die Trigger für EMP werden entweder vor oder nach der Löschung (bei Angabe von ON DELETE CASCADE) oder der Aktualisierung von Zeilen in EMP (bei Angabe von ON DELETE SET NULL) abhängig von ihrer Aktivierungszeit ausgeführt.

Angeben der Auslösebedingungen eines Triggers (auslösende Anweisung oder Auslöseereignis)

Jeder Trigger ist einem Ereignis zugeordnet. Trigger werden aktiviert, wenn das ihnen zugeordnete Ereignis in der Datenbank eintritt. Dieses Trigger-Ereignis (Auslöseereignis) tritt ein, wenn die angegebene Aktion, das heißt eine Anweisung UPDATE, INSERT oder DELETE (einschließlich solcher, die durch Aktionen von referenziellen Integritätsbedingungen verursacht werden) an der Zieltabelle ausgeführt wird.

Beispiel:

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

Die oben gezeigte Anweisung definiert den Trigger NEW_HIRE, der aktiviert wird, wenn Sie eine INSERT-Operation an der Tabelle EMPLOYEE ausführen.

Sie ordnen jedes Trigger-Ereignis und infolgedessen jeden Trigger genau einer Zieltabelle und genau einer Änderungsoperation zu. Folgende Änderungsoperationen sind verfügbar:

INSERT-Operation
Eine INSERT-Operation kann nur durch eine Anweisung INSERT oder durch die INSERT-Operation einer Anweisung MERGE verursacht werden. Daher werden entsprechende Trigger nicht aktiviert, wenn Daten mithilfe von Dienstprogrammen geladen werden, die kein INSERT verwenden, wie zum Beispiel der Befehl LOAD.

DELETE-Operation
Eine DELETE-Operation kann durch eine Anweisung DELETE, durch die DELETE-Operation einer Anweisung MERGE oder durch die referenzielle Integritätsregel ON DELETE CASCADE verursacht werden.

UPDATE-Operation
Eine UPDATE-Operation kann durch eine Anweisung UPDATE, durch die UPDATE-Operation einer Anweisung MERGE oder durch die referenzielle Integritätsregel ON DELETE SET NULL verursacht werden.

Wenn das Trigger-Ereignis eine UPDATE-Operation ist, kann das Ereignis mit bestimmten Spalten der Zieltabelle verknüpft werden. In diesem Fall wird der Trigger nur aktiviert, wenn die UPDATE-Operation versucht, eine der angegebenen Spalten zu aktualisieren. Dies bietet weitere Verfeinerungsmöglichkeiten für die Angabe des Ereignisses, das den Trigger aktiviert.

Zum Beispiel wird der folgende Trigger REORDER nur aktiviert, wenn eine UPDATE-Operation an den Spalten ON_HAND oder MAX_STOCKED der Tabelle PARTS ausgeführt wird:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
  FOR EACH ROW
  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED)
  BEGIN ATOMIC
  VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                            N_ROW.ON_HAND,
                            N_ROW.PARTNO));
  END
```

Wenn ein Trigger aktiviert wird, wird er entsprechend seiner Granularität wie folgt ausgeführt:

FOR EACH ROW
: Er wird so oft ausgeführt, wie Zeilen in der Menge der betroffenen Zeilen enthalten sind. Wenn Sie sich auf die bestimmten, von der ausgelösten Aktion betroffenen Zeilen beziehen müssen, verwenden Sie die Granularität FOR EACH ROW. Ein Beispiel für diesen Fall ist der Vergleich der neuen und alten Werte einer aktualisierten Zeile in einem mit AFTER UPDATE definierten Trigger.

FOR EACH STATEMENT
: Der Trigger wird einmal für das gesamte Trigger-Ereignis ausgeführt.

Wenn die Menge der betroffenen Zeilen leer ist (d. h. im Fall einer gezielten UPDATE- oder DELETE-Operation, bei der die WHERE-Klausel keine Zeilen ergeben hat), wird der Trigger mit der Definition FOR EACH ROW nicht ausgeführt. Ein Trigger mit der Definition FOR EACH STATEMENT wird jedoch auch in diesem Fall einmal ausgeführt.

Zum Beispiel kann eine Verfolgung der Anzahl von Mitarbeitern mit der Granularität FOR EACH ROW realisiert werden:

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

Denselben Effekt können Sie auch mit einer Aktualisierung erzielen, indem Sie die Granularität FOR EACH STATEMENT verwenden:

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  REFERENCING NEW_TABLE AS NEWEMPS
  FOR EACH STATEMENT
  UPDATE COMPANY_STATS
  SET NBEMP = NBEMP + (SELECT COUNT(*) FROM NEWEMPS)
```

Anmerkung:

- Die Granularität FOR EACH STATEMENT wird von Vortriggern (BEFORE-Triggern) nicht unterstützt.
- Die maximale Verschachtelungsebene für Trigger beträgt 16. Das heißt, die maximale Anzahl hintereinandergeschalteter Triggeraktivierungen beträgt 16. Eine Triggeraktivierung bezieht sich auf die Aktivierung eines Triggers aufgrund eines auslösenden Ereignisses, wie z. B. einer Einfügung, einer Aktualisierung oder einer Löschung von Daten in einer Tabellenspalte, oder im Allgemeinen auf eine Tabelle.

Angeben des Aktivierungszeitpunkts eines Triggers (Klauseln BEFORE, AFTER und INSTEAD OF)

Die *Aktivierungszeit des Triggers* gibt an, wann der Trigger in Relation zum Trigger-Ereignis aktiviert werden soll.

Es gibt drei Aktivierungszeiten, die Sie angegeben können: BEFORE, AFTER oder INSTEAD OF:

- Wenn die Aktivierungszeit mit BEFORE (Vortrigger) definiert wird, werden die ausgelösten Aktionen für jede Zeile in der Menge der betroffenen Zeilen aktiviert, bevor das Trigger-Ereignis ausgeführt wird. Das bedeutet, dass die Subjekttabelle erst geändert wird, nachdem der Vortrigger (BEFORE-Trigger) die Ausführung seiner Aktion für jede Zeile abgeschlossen hat. Beachten Sie, dass Vortrigger die Granularität FOR EACH ROW haben müssen.
- Wenn die Aktivierungszeit mit AFTER (Nachtrigger) definiert wird, werden die ausgelösten Aktionen entsprechend der angegebenen Granularität des Triggers für jede Zeile in der Menge der betroffenen Zeilen oder für die Anweisung aktiviert. Dies geschieht, nachdem das Trigger-Ereignis ausgeführt wurde und der Datenbankmanager alle Integritätsbedingungen, die von dem Trigger-Ereignis betroffen sein könnten, einschließlich der Aktionen durch referenzielle Integritätsbedingungen, überprüft hat. Beachten Sie, dass für Nachtrigger Granularität FOR EACH ROW oder FOR EACH STATEMENT angegeben werden kann.

  Die Aktivierungszeit des folgenden Triggers ist zum Beispiel nach der INSERT-Operation an der Tabelle EMPLOYEE:

  ```
  CREATE TRIGGER NEW_HIRE
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
  ```

- Wenn die Aktivierungszeit mit INSTEAD OF definiert wird, werden die ausgelösten Aktionen für jede Zeile in der Menge der betroffenen Zeilen anstatt der Ausführung des Trigger-Ereignisses aktiviert. INSTEAD OF-Trigger müssen die Granularität FOR EACH ROW haben, und als Subjekttabelle muss eine Sicht angegeben werden. Keine anderen Trigger können eine Sicht als Subjekttabelle verwenden.

Das folgende Diagramm veranschaulicht das Ausführungsmodell von Vortriggern und Nachtriggern:

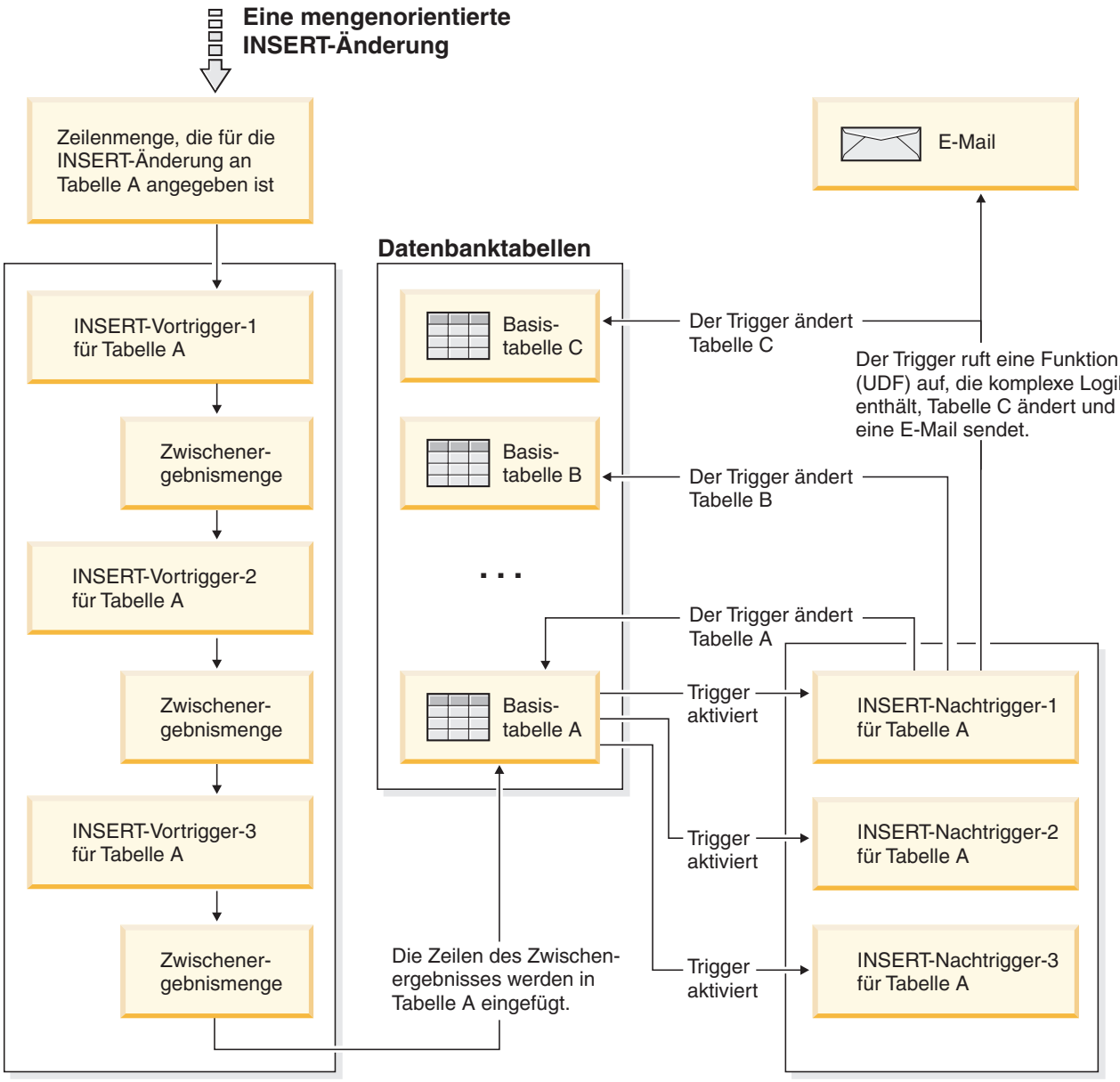


Abbildung 26. Ausführungsmodell für Trigger

Für eine Tabelle mit Vortriggern und Nachtriggern sowie mit einem Änderungsereignis, das diesen Triggern zugeordnet ist, werden alle Vortrigger zuerst aktiviert. Der erste Vortrigger für ein bestimmtes Ereignis operiert an der Menge der Zeilen, die Ziel der Operation sind, und nimmt alle Aktualisierungsänderungen an der Menge vor, die von seiner Logik vorgeschrieben werden. Die Ausgabe dieses Vortriggers wird vom nächsten Vortrigger als Eingabe empfangen. Wenn alle Vortrigger, die von dem Ereignis aktiviert wurden, ausgeführt wurden, wird die Zwischenergebnismenge, das heißt das Ergebnis der Änderungen durch die Vortrigger an den Zielzeilen der Trigger-Ereignisoperation, auf die Tabelle angewendet. Nachfolgend werden die einzelnen, dem Ereignis zugeordneten Nachtrigger aktiviert. Die Nachtrigger können dieselbe Tabelle oder eine andere Tabelle ändern oder auch eine datenbankexterne Aktion ausführen.

Die verschiedenen Aktivierungszeiten von Triggern entsprechen den verschiedenen Zwecken von Triggern. Im Grunde sind Vortrigger eine Erweiterung des Subsys-

tems der Integritätsbedingungen des Datenbankmanagementsystems. Daher werden sie in der Regel zu folgenden Zwecken eingesetzt:

- Zur Überprüfung von Eingabedaten
- Zur automatischen Generierung von Werten für neu eingefügte Zeilen
- Zum Lesen aus anderen Tabellen zur Auflösung von Querverweisen

Vortrigger werden nicht zur weiteren Modifikation der Datenbank verwendet, weil sie aktiviert werden, bevor das Trigger-Ereignis auf die Datenbank angewendet wird. Infolgedessen werden sie aktiviert, bevor Integritätsbedingungen überprüft werden.

Im Gegensatz dazu können Sie Nachtrigger als Modul der Anwendungslogik betrachten, das in der Datenbank bei jedem Auftreten eines bestimmten Ereignisses ausgeführt wird. Als Teil einer Anwendung sehen Nachtrigger die Datenbank immer in einem konsistenten Status. Beachten Sie, dass diese Trigger nach den Überprüfungen der Integritätsbedingungen ausgeführt werden. Infolgedessen können Sie sie in erster Linie zur Ausführung von Operationen verwenden, die auch von einer Anwendung ausgeführt werden können. Beispiel:

- Sie können Folgeänderungsoperationen in der Datenbank ausführen.
- Sie können Aktionen außerhalb der Datenbank ausführen, zum Beispiel um Alerts zu unterstützen. Beachten Sie, dass Aktionen, die außerhalb der Datenbank ausgeführt werden, nicht rückgängig gemacht werden, falls der Trigger rückgängig gemacht wird.

Im Unterschied dazu können Sie einen INSTEAD OF-Trigger als Beschreibung der inversen Operation der Sicht, für die er definiert ist, betrachten. Wenn zum Beispiel die SELECT-Liste in der Sicht einen Ausdruck über eine Tabelle enthält, enthält die INSERT-Anweisung im Hauptteil des entsprechenden INSTEAD OF-Triggers für die INSERT-Operation den umgekehrten Ausdruck.

Aufgrund der unterschiedlichen Merkmale von BEFORE-, AFTER- und INSTEAD OF-Triggern kann jeweils eine andere Gruppe von SQL-Operationen zur Definition der ausgelösten Aktionen von BEFORE-, AFTER- und INSTEAD OF-Triggern verwendet werden. So sind zum Beispiel UPDATE-Operationen in Vortriggern nicht zulässig, da es keine Garantie gibt, dass keine Integritätsbedingungen durch die ausgelöste Aktion verletzt werden. Analog werden in BEFORE-, AFTER- und INSTEAD OF-Triggern verschiedene Triggergranularitäten unterstützt.

Die ausgelöste SQL-Anweisung aller Trigger kann eine dynamische zusammengesetzte Anweisung sein. Für Vortrigger gelten jedoch einige Einschränkungen, da sie die folgenden SQL-Anweisungen nicht enthalten dürfen:

- UPDATE
- DELETE
- INSERT
- MERGE

Definieren von Bedingungen für den Aktivierungszeitpunkt einer Triggeraktion (Klausel WHEN)

Die Aktivierung eines Triggers führt zur Ausführung der zugeordneten ausgelösten Aktion. Jeder Trigger hat genau eine ausgelöste Aktion, die wiederum zwei Komponenten hat: eine optionale *Bedingung zum Auslösen der Aktion* bzw. WHEN-Klausel und eine Folge aus einer oder mehreren *ausgelösten Anweisungen*.

Die *Bedingung zum Auslösen der Aktion* ist eine optionale Klausel der ausgelösten Aktion, die eine Suchbedingung angibt, die als wahr ausgewertet werden muss, damit die Anweisungen in der ausgelösten Aktion ausgeführt werden. Wenn die WHEN-Klausel nicht angegeben wird, werden die Anweisungen in der ausgelösten Aktion in jedem Fall ausgeführt.

Die Bedingung zum Auslösen der Aktion wird einmal pro Zeile ausgewertet, wenn der Trigger mit der Granularität FOR EACH ROW definiert ist, und einmal pro Anweisung, wenn der Trigger mit der Granularität FOR EACH STATEMENT definiert ist.

Diese Klausel bietet weitere Steuerungsmöglichkeiten für die Feinabstimmung der Aktionen, die durch einen Trigger aktiviert werden. Ein Beispiel für die Nützlichkeit der WHEN-Klausel ist die Umsetzung einer von den Daten abhängigen Regel, bei der die ausgelöste Aktion nur aktiviert wird, wenn der eingehende Wert innerhalb oder außerhalb eines bestimmten Bereichs liegt.

Die Aktivierung eines Triggers führt zur Ausführung der zugeordneten ausgelösten Aktion. Jeder Trigger hat genau eine ausgelöste Aktion, die wiederum zwei Komponenten hat: eine optionale Bedingung zum Auslösen der Aktion bzw. WHEN-Klausel und eine Folge aus einer oder mehreren ausgelösten Anweisungen.

Die Bedingung zum Auslösen der Aktion definiert, ob die ausgelösten Anweisungen für die Zeile bzw. für die Anweisung, für die die ausgelöste Aktion ausgeführt wird, ausgeführt werden oder nicht. Die ausgelösten Anweisungen definieren die Aktionen, die von dem Trigger in der Datenbank ausgeführt werden, wenn das dem Trigger zugeordnete Ereignis eintritt.

Die folgende Triggeraktion gibt zum Beispiel an, dass die ausgelösten Anweisungen nur für Zeilen aktiviert werden sollen, in denen der Wert der Spalte `ON_HAND` kleiner als zehn Prozent des Werts der Spalte `MAX_STOCKED` ist. In diesem Fall wird durch die ausgelösten Anweisungen die Funktion `ISSUE_SHIP_REQUEST` aufgerufen.

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
  FOR EACH ROW

  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED)
  BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                              N_ROW.ON_HAND,
                              N_ROW.PARTNO));
  END
```

Die ausgelösten Anweisungen führen die tatsächlichen Aktionen aus, die durch die Aktivierung eines Triggers verursacht werden. Nicht jede SQL-Operation ist in jeder Art von Trigger sinnvoll. Je nachdem, ob die Aktivierungszeit des Triggers mit BEFORE oder AFTER definiert ist, eignen sich verschiedene Arten von Operationen als ausgelöste Anweisungen.

Wenn eine der ausgelösten Anweisungen einen negativen Rückkehrcode zurückgibt, werden in den meisten Fällen die auslösende Anweisung sowie alle Triggeraktionen und Aktionen durch referenzielle Integritätsbedingungen rückgängig gemacht (ROLLBACK). Der Name des Triggers, der SQLCODE-Wert und der SQLSTATE-Wert sowie viele der Token aus der fehlgeschlagenen ausgelösten Anweisung werden in der Fehlernachricht zurückgegeben.

Unterstützte SQL PL-Anweisungen in Triggern

Die ausgelöste SQL-Anweisung aller Trigger kann eine zusammengesetzte, dynamische Anweisung sein.

Das heißt, ausgelöste SQL-Anweisungen können eines oder mehrere der folgenden Elemente enthalten:

- Anweisung CALL
- Anweisung DECLARE variable
- Anweisung SET variable
- WHILE-Schleife
- FOR-Schleife
- Anweisung IF
- Anweisung SIGNAL
- Anweisung ITERATE
- Anweisung LEAVE
- Anweisung GET DIGNOSTIC
- Fullselect

Jedoch können nur Nachtrigger (AFTER-Trigger) und INSTEAD OF-Trigger eine oder mehrere der folgenden SQL-Anweisungen enthalten:

- Anweisung UPDATE
- Anweisung DELETE
- Anweisung INSERT
- Anweisung MERGE

Zugreifen auf alte und neue Spaltenwerte in Triggern durch Übergangsvariablen

Wenn Sie einen Trigger mit der Definition FOR EACH ROW implementieren, kann es erforderlich sein, auf den Wert der Spalten der Zeile in der Menge der betroffenen Zeilen, für die der Trigger gerade ausgeführt wird, zu verweisen. Beachten Sie, dass Sie zum Verweisen auf Spalten in Tabellen in der Datenbank (einschließlich der Subjekttabelle) reguläre SELECT-Anweisungen verwenden können.

Ein mit der Granularität FOR EACH ROW definierter Trigger kann auf Spalten der Zeile verweisen, für die er gerade ausgeführt wird, indem er zwei Übergangsvariablen verwendet, die in der Klausel REFERENCING einer Anweisung CREATE TRIGGER angegeben werden. Es gibt zwei Arten von Übergangsvariablen, die durch OLD und NEW sowie einen Korrelationsnamen angegeben werden. Sie besitzen folgende Semantik:

OLD AS korrelationsname
: Gibt einen Korrelationsnamen an, der den ursprünglichen Status der Zeile erfasst, das heißt, bevor die ausgelöste Aktion auf die Datenbank angewendet wird.

NEW AS korrelationsname
: Gibt einen Korrelationsnamen an, der den Wert erfasst, der zum Aktualisieren der Zeile in der Datenbank verwendet wird bzw. wurde, wenn die ausgelöste Aktion auf die Datenbank angewendet wird.

Betrachten Sie das folgende Beispiel:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
  FOR EACH ROW
  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED
  AND N_ROW.ORDER_PENDING = 'N')
  BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                              N_ROW.ON_HAND,
                              N_ROW.PARTNO));
    UPDATE PARTS SET PARTS.ORDER_PENDING = 'Y'
    WHERE PARTS.PARTNO = N_ROW.PARTNO;
  END
```

An der oben gezeigten Definition der Übergangsvariablen OLD und NEW wird klar, dass nicht jede Übergangsvariable für jeden Trigger definiert werden kann. Übergangsvariablen können abhängig von der Art des Trigger-Ereignisses definiert werden:

UPDATE
Ein UPDATE-Trigger kann beide Übergangsvariablen (OLD und NEW) verwenden.

INSERT
Ein INSERT-Trigger kann sich nur auf eine Übergangsvariable NEW beziehen, da die betroffene Zeile vor der Aktivierung der INSERT-Operation in der Datenbank nicht vorhanden ist. Das heißt, dass es keinen ursprünglichen Status der Zeile gibt, der alte Werte definieren würde, bevor die ausgelöste Aktion auf die Datenbank angewendet wird.

DELETE
Ein DELETE-Trigger kann sich nur auf eine Übergangsvariable OLD beziehen, weil in der DELETE-Operation keine neuen Werte angegeben werden.

Anmerkung: Übergangsvariablen können nur für Trigger angegeben werden, die mit der Granularität FOR EACH ROW definiert werden. In einem Trigger mit der Granularität FOR EACH STATEMENT reicht ein Verweis auf eine Übergangsvariable nicht aus, um anzugeben, auf welche der Zeilen in der Menge der betroffenen Zeilen sich die Übergangsvariable bezieht. Verweisen Sie stattdessen auf die Menge der alten und neuen Zeilen, indem Sie die Klauseln OLD TABLE und NEW TABLE der Anweisung CREATE TRIGGER verwenden. Weitere Informationen zu diesen Klauseln finden Sie in der Beschreibung der Anweisung CREATE TRIGGER.

Verweisen auf alte und neue Tabellenergebnismengen mit Übergangstabellen

In Triggern, die mit FOR EACH ROW bzw. FOR EACH STATEMENT definiert sind, kann es erforderlich sein, auf die gesamte Menge der betroffenen Zeilen zu verweisen. Dies ist zum Beispiel der Fall, wenn der Triggerhauptteil Spaltenberechnungen auf die Menge der betroffenen Zeilen anwenden muss (z. B. MAX, MIN oder AVG für einige Spaltenwerte).

Ein Trigger kann auf die Menge der betroffenen Zeilen Bezug nehmen, indem er zwei Übergangstabellen verwendet, die in der Klausel REFERENCING einer Anweisung CREATE TRIGGER angegeben werden können. Ebenso wie bei Übergangsvariablen gibt es auch bei Übergangstabellen zwei Arten, die durch OLD_TABLE und NEW_TABLE sowie durch einen Tabellennamen mit der folgenden Semantik angegeben werden:

OLD_TABLE AS tabellenname
Gibt den Namen der Tabelle an, die den ursprünglichen Status der Menge der betroffenen Zeilen erfasst (d. h. bevor die auslösende SQL-Operation auf die Datenbank angewendet wird).

NEW_TABLE AS tabellenname
Gibt den Namen der Tabelle an, die den Wert erfasst, der zur Aktualisierung der Zeilen in der Datenbank verwendet wird, wenn die ausgelöste Aktion auf die Datenbank angewendet wird.

Beispiel:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW_TABLE AS N_TABLE
  NEW AS N_ROW
  FOR EACH ROW
  WHEN ((SELECT AVG (ON_HAND) FROM N_TABLE) > 35)
  BEGIN ATOMIC
    VALUES(INFORM_SUPERVISOR(N_ROW.PARTNO,
                             N_ROW.MAX_STOCKED,
                             N_ROW.ON_HAND));
  END
```

Beachten Sie, dass NEW_TABLE immer die vollständige Menge der aktualisierten Zeilen enthält, auch für einen Trigger mit der Granularität FOR EACH ROW. Wenn ein Trigger Operationen an der Tabelle ausführt, für die der Trigger definiert ist, enthält NEW_TABLE die geänderten Zeilen aus der Anweisung, die den Trigger aktiviert hat. Die Übergangstabelle NEW_TABLE enthält jedoch nicht die geänderten Zeilen, die durch Anweisungen innerhalb des Triggers bewirkt wurden, da dies wiederum eine separate Aktivierung des Triggers zur Folge hätte.

Die Übergangstabellen sind schreibgeschützt. Für Übergangstabellen gelten die gleichen Regeln, die auch bei der Definition der Art von Übergangsvariablen gelten, die für die verschiedenen Trigger-Ereignisse definiert werden können:

UPDATE
Ein UPDATE-Trigger kann beide Übergangstabellen (OLD_TABLE und NEW_TABLE) verwenden.

INSERT
Ein INSERT-Trigger kann sich nur auf eine Übergangstabelle NEW_TABLE beziehen, da die betroffenen Zeilen vor der Aktivierung der INSERT-Operation in der Datenbank nicht vorhanden sind. Das heißt, dass es keinen ursprünglichen Status der Zeilen gibt, der alte Werte definiert, bevor die ausgelöste Aktion auf die Datenbank angewendet wird.

DELETE
Ein DELETE-Trigger kann sich nur auf eine Übergangstabelle OLD_TABLE beziehen, weil in der DELETE-Operation keine neuen Werte angegeben werden.

Anmerkung: Es wichtig zu beachten, dass Übergangstabellen für beide Granularitäten von AFTER-Triggern angegeben werden können: FOR EACH ROW und FOR EACH STATEMENT.

Der Geltungsbereich der Angabe `OLD_TABLE` und `NEW_TABLE tabellenname` ist der Triggerhauptteil. In diesem Geltungsbereich hat dieser Tabellenname Vorrang vor den Namen jeder anderen Tabelle mit demselben *tabellennamen* ohne Qualifikationsmerkmal, die möglicherweise im Schema vorhanden ist. Das bedeu-

tet, wenn für OLD_TABLE bzw. NEW_TABLE tabellenname zum Beispiel 'X' angegeben wird, bezieht sich ein Verweis auf X (d. h. ein X ohne Qualifikationsmerkmal) in der FROM-Klausel einer SELECT-Anweisung immer auf die Übergangstabelle, auch wenn im Schema des Trigger-Erstellers eine Tabelle mit dem Namen 'X' vorhanden ist. In diesem Fall muss der Benutzer den vollständig qualifizierten Namen verwenden, um auf die Tabelle 'X' im Schema zu verweisen.

Erstellen von Triggern

Ein Trigger definiert eine Reihe von Aktionen, die in Verbindung mit bzw. ausgelöst durch eine Anweisung INSERT, UPDATE oder DELETE für eine angegebene Tabelle bzw. eine typisierte Tabelle ausgeführt werden.

Verwenden Sie Trigger zu folgenden Zwecken:

- Überprüfen der Gültigkeit von Eingabedaten
- Generieren eines Werts für eine neu eingefügte Zeile
- Lesen aus anderen Tabellen zur Auflösung von Querverweisen
- Schreiben in andere Tabellen zur Führung von Prüfprotokollen

Trigger können zur Unterstützung allgemeiner Formen von Datenintegrität und Geschäftsregeln verwendet werden. Zum Beispiel kann ein Trigger den Kreditrahmen eines Kunden überprüfen, bevor eine Bestellung entgegengenommen oder eine Tabelle mit Übersichtsdaten aktualisiert wird.

Vorteile:

- Schnellere Anwendungsentwicklung: Da ein Trigger in der Datenbank gespeichert wird, müssen die durch den Trigger ausgeführten Aktionen nicht mehr in jeder Anwendung codiert werden.
- Einfachere Wartung: Wenn ein Trigger definiert ist, wird er automatisch aufgerufen, wenn auf die Tabelle, für die er erstellt wurde, zugegriffen wird.
- Globale Implementierung von Geschäftsregeln: Wenn sich die geschäftsinternen Abläufe oder Regeln ändern, müssen lediglich die Trigger und nicht sämtliche Anwendungsprogramme geändert werden.

Einschränkungen:

- Trigger können nicht mit Kurznamen verwendet werden.
- Wenn es sich bei dem Trigger um einen Vortrigger (BEFORE) handelt, darf der von der ausgelösten Aktion angegebene Spaltenname keine generierte Spalte außer einer Identitätsspalte bezeichnen. Dies bedeutet, dass der generierte Identitätswert für Vortrigger sichtbar ist.

Beim Erstellen eines ganzheitlichen Triggers (ATOMIC) muss das Zeichen für das Anweisungsende sorgfältig behandelt werden. Der Befehlszeilenprozessor erkennt standardmäßig ein Semikolon („;") als Markierung für das Anweisungsende. Sie sollten das Zeichen für das Anweisungsende manuell im Script für die Erstellung des ganzheitlichen Triggers editieren, um ein anderes Zeichen als das Semikolon („;") zu verwenden. Das Semikolon („;") könnte zum Beispiel durch ein anderes Sonderzeichen wie das Nummernzeichen („#") ersetzt werden. Sie können der DDL-Anweisung CREATE TRIGGER auch Folgendes voranstellen:

```
--#SET TERMINATOR @
```

Wenn das Abschlusszeichen im Befehlszeilenprozessor während der Verarbeitung geändert werden soll, wird es durch die folgende Syntax zurückgesetzt:

```
--#SET TERMINATOR
```

Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen Trigger zu erstellen:

```
db2 -td <begrenzer> -vf <script>
```

Dabei ist <begrenzer> das alternative Zeichen für das Anweisungsende und <script> das modifizierte Script mit dem neuen <begrenzer>.

Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen Trigger zu erstellen:

```
CREATE TRIGGER <name>
   <aktion> ON <tabellenname>
   <operation>
   <ausgelöste_aktion>
```

Mit der folgenden Anweisung wird ein Trigger erstellt, der die Anzahl der Mitarbeiter automatisch erhöht, wenn eine neue Person angestellt wird, indem zum Wert in der Spalte für die Anzahl der Mitarbeiter (NBEMP) in der Tabelle COMPANY_STATS jedes Mal der Wert 1 addiert wird, wenn der Tabelle EMPLOYEE eine Zeile hinzugefügt wird:

```
CREATE TRIGGER NEW_HIRED
   AFTER INSERT ON EMPLOYEE
   FOR EACH ROW
   UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

Der Hauptteil einer Triggerdefinition kann eine oder mehrere der folgenden Anweisungen enthalten: Anweisung INSERT, gezielte Anweisung UPDATE, gezielte Anweisung DELETE, Fullselect, Anweisung SET variable und Anweisung SIGNAL SQLSTATE. Der Trigger kann vor oder nach der Anweisung INSERT, UPDATE bzw. DELETE, auf die er bezogen ist, aktiviert werden.

Ändern und Löschen von Triggern

Trigger können nicht geändert werden. Sie müssen gelöscht und anschließend den neuen erforderlichen Definitionen entsprechend erstellt werden.

Triggerabhängigkeiten

- Alle Abhängigkeiten eines Triggers von einem anderen Objekt werden in der Katalogsicht SYSCAT.TRIGDEP aufgezeichnet. Ein Trigger kann von vielen Objekten abhängig sein.
- Wenn ein Objekt, von dem ein Trigger abhängig ist, gelöscht wird, wird der Trigger unbrauchbar. Die Definition verbleibt jedoch in der Katalogsicht. Um einen solchen Trigger wieder brauchbar zu machen, müssen Sie die entsprechende Definition aus der Systemkatalogsicht abrufen und eine neue Anweisung CREATE TRIGGER ausführen.
- Wenn ein Trigger gelöscht wird (DROP), wird die entsprechende Beschreibung aus der Systemkatalogsicht SYSCAT.TRIGGERS gelöscht, und alle zugehörigen Einträge für Abhängigkeiten werden aus der Systemkatalogsicht SYSCAT.TRIGDEP gelöscht. Alle Pakete mit Abhängigkeiten über Anweisungen UPDATE, INSERT oder DELETE für den Trigger, werden ungültig gemacht.
- Wenn die Sicht von dem Trigger abhängig ist und funktionsunfähig gemacht wird, wird der Trigger ebenfalls als unbrauchbar markiert. Alle Pakete, die von Triggern abhängig sind, die als unbrauchbar markiert wurden, werden ungültig gemacht.

Ein Triggerobjekt kann mit der Anweisung DROP TRIGGER gelöscht werden. Dieses Verfahren hat jedoch zur Folge, dass abhängige Pakete als ungültig markiert werden, wie im Folgenden beschrieben:

- Wenn ein UPDATE-Trigger, für den keine explizite Spaltenliste definiert ist, gelöscht wird, werden Pakete, die eine Aktualisierung an der Zieltabelle ausführen, ungültig gemacht.
- Wenn ein UPDATE-Trigger, für den eine Spaltenliste definiert ist, gelöscht wird, werden Pakete, die eine Aktualisierung an der Zieltabelle ausführen, nur dann ungültig gemacht, wenn das Paket auch eine Aktualisierung für mindestens eine Spalte in der Liste der Spaltennamen der Anweisung CREATE TRIGGER enthält.
- Wenn ein INSERT-Trigger gelöscht wird, werden Pakete, die eine Einfügung für die Zieltabelle enthalten, ungültig gemacht.
- Wenn ein DELETE-Trigger gelöscht wird, werden Pakete, die eine Löschung für die Zieltabelle enthalten, ungültig gemacht.

Ein Paket bleibt ungültig, bis das Anwendungsprogramm explizit gebunden(Bind) bzw. erneut gebunden (Rebind) wird oder bis das Anwendungsprogramm ausgeführt wird und der Datenbankmanager einen automatischen Rebind durchführt.

Beispiele zu Triggern und zur Verwendung von Triggern

Beispiele für die Interaktion zwischen Triggern und referenziellen Integritätsbedingungen

Aktualisierungsoperationen (UPDATE) können zu Interaktionen von Triggern mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen führen.

Abb. 23 auf Seite 320 und die zugehörige Beschreibung sind für die Verarbeitung repräsentativ, die für eine Anweisung ausgeführt wird, die Daten in der Datenbank aktualisiert.

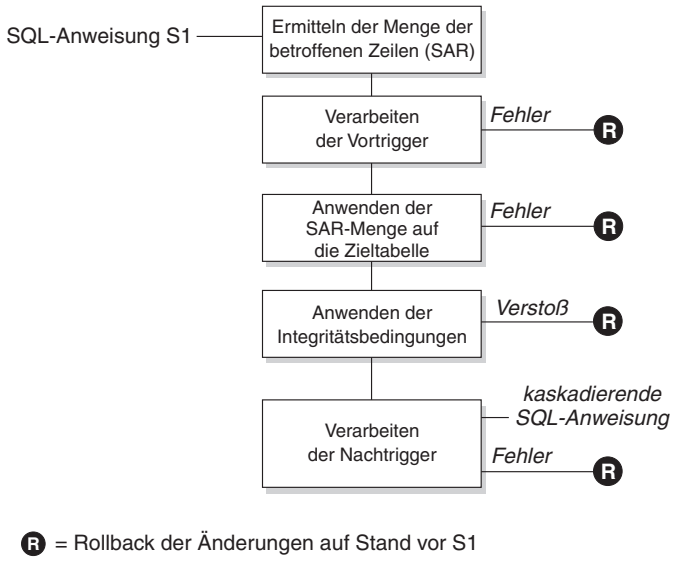


Abbildung 27. Verarbeitung einer Anweisung mit definierten Triggern und Integritätsbedingungen

Abb. 23 auf Seite 320 zeigt den allgemeinen Verarbeitungsablauf für eine Anweisung, die eine Tabelle aktualisiert. Sie geht von einer Situation aus, bei der die Tabelle Vortrigger, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Nachtrigger enthält, die hintereinander (kaskadierend) verarbeitet werden. Im Folgenden werden die Kästchen und anderen Elemente in Abb. 23 auf Seite 320 beschrieben.

- Anweisung S_1

  Dies ist die DELETE-, INSERT- oder UPDATE-Anweisung, die den Prozess erfordert. Die Anweisung S_1 gibt eine Tabelle (bzw. eine aktualisierbare Sicht für eine Tabelle) an, die in dieser Beschreibung als *Subjekttabelle* bezeichnet wird.

- Ermitteln der Menge der betroffenen Zeilen

  Dieser Schritt ist der Ausgangspunkt für einen Prozess, der für die Löschregeln CASCADE und SET NULL referenzieller Integritätsbedingungen sowie für kaskadierende Anweisungen aus Nachtriggern wiederholt ausgeführt wird.

  Zweck dieses Schritts ist die Ermittlung der *Menge der betroffenen Zeilen* (Set of Affected Rows - SAR) für die Anweisung. Die in dieser Menge enthaltenen Zeilen hängen von der Anweisung ab:

  – Für DELETE: Alle Zeilen, die die Suchbedingung der Anweisung erfüllen (oder die aktuelle Zeile bei einer positionierten DELETE-Operation)
  – Für INSERT: Die Zeilen, die durch die Klausel VALUES oder den Fullselect angegeben werden
  – Für UPDATE: Alle Zeilen, die die Suchbedingung erfüllten (oder die aktuelle Zeile bei einer positionierten UPDATE-Operation)

  Wenn die Menge der betroffenen Zeilen leer ist, sind keine Vortrigger, keine Änderungen zur Anwendung auf die Subjekttabelle und keine Integritätsbedingungen für die Anweisung zu verarbeiten.

- Verarbeiten der Vortrigger

  Alle Vortrigger (BEFORE) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet. Jeder Vortrigger verarbeitet die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

  Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

  Wenn keine Vortrigger definiert sind oder die Menge der betroffenen Zeilen leer ist, wird dieser Schritt übersprungen.

- Anwenden der Menge der betroffenen Zeilen auf die Subjekttabelle

  Die tatsächliche DELETE-, INSERT- oder UPDATE-Operation wird unter Verwendung der Menge der betroffenen Zeilen auf die Subjekttabelle (Zieltabelle) in der Datenbank angewendet.

  Beim Anwenden der Menge der betroffenen Zeilen kann ein Fehler auftreten (z. B. wenn versucht wird, eine Zeile mit einem bereits vorhandenen Schlüssel einzufügen, obwohl ein eindeutiger Index vorhanden ist). In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Anwenden der Integritätsbedingungen

  Die der Subjekttabelle zugeordneten Integritätsbedingungen werden angewendet, wenn die Menge der betroffenen Zeilen nicht leer ist. Dazu gehören eindeutige Integritätsbedingungen, eindeutige Indizes, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Prüfungen der Klausel

WITH CHECK OPTION für Sichten. Referenzielle Integritätsbedingungen mit den Löschregeln CASCADE oder SET NULL können die Aktivierung weiterer Trigger bewirken.

Ein Verstoß gegen eine Integritätsbedingung oder die Klausel WITH CHECK OPTION führt zu einem Fehler. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Verarbeiten der Nachtrigger

  Alle von S_1 aktivierten Nachtrigger (AFTER) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet.

  Trigger mit der Definition FOR EACH STATEMENT verarbeiten die ausgelöste Aktion genau einmal, auch wenn die Menge der betroffenen Zeilen leer ist. Trigger mit der Definition FOR EACH ROW verarbeiten die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

  Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

  Die ausgelöste Aktion eines Triggers kann ausgelöste Anweisungen enthalten, bei denen es sich um DELETE-, INSERT- oder UPDATE-Anweisungen handelt. Zum Zweck dieser Beschreibung wird eine solche Anweisung als *kaskadierende Anweisung* betrachtet.

  Eine kaskadierende Anweisung ist eine DELETE-, INSERT- oder UPDATE-Anweisung, die als Teil der ausgelösten Aktion eines Nachtriggers verarbeitet wird. Eine solche Anweisung wird auf einer nachfolgenden Ebene der Triggerverarbeitung gestartet. Dies lässt sich in etwa so beschreiben, dass die ausgelöste Anweisung als neue Anweisung S_1 zugeordnet wird und alle hier beschriebenen Schritte wiederum rekursiv ausgeführt werden.

  Wenn alle ausgelösten Anweisungen aus allen Nachtriggern, die von jeder Anweisung S_1 aktiviert werden, vollständig verarbeitet wurden, wird die Verarbeitung der ursprünglichen Anweisung S_1 abgeschlossen.

- R = Rollback der Änderungen auf Stand vor S_1

  Jeder Fehler (einschließlich Verstößen gegen Integritätsbedingungen) während der Verarbeitung führt zu einem Rollback aller Änderungen, die direkt oder indirekt infolge der ursprünglichen Anweisung S_1 ausgeführt wurden. Die Datenbank wird in den Zustand zurückversetzt, den sie unmittelbar vor der Ausführung der ursprünglichen Anweisung S_1 hatte.

Beispiele für das Definieren von Aktionen mit Triggern

Nehmen Sie an, ein Generalmanager möchte die Namen von Kunden, die drei oder mehr Beschwerden in den letzten 72 Stunden gesendet haben, in einer separaten Tabelle speichern. Der Generalmanager möchte außerdem informiert werden, wenn ein Kundenname in diese Tabelle mehr als einmal eingefügt wird.

Zur Definition solcher Aktionen gehen Sie zum Beispiel wie folgt vor:

- Definieren Sie eine Tabelle für unzufriedene Kunden (UNHAPPY_CUSTOMERS):

```
CREATE TABLE UNHAPPY_CUSTOMERS (
  NAME           VARCHAR (30),
  EMAIL_ADDRESS  VARCHAR (200),
  INSERTION_DATE DATE)
```

- Definieren Sie einen Trigger, der automatisch eine Zeile in UNHAPPY_CUSTOMERS einfügt, wenn 3 oder mehr Nachrichten innerhalb der letzten 3 Tage emp-

fangen wurden (unter der Annahme, dass eine Kundentabelle CUSTOMERS vorhanden ist, die eine Spalte NAME und eine Spalte E_MAIL_ADDRESS enthält):

```
CREATE TRIGGER STORE_UNHAPPY_CUST
  AFTER INSERT ON ELECTRONIC_MAIL
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (3 <= (SELECT COUNT(*)
              FROM ELECTRONIC_MAIL
              WHERE SENDER = N.SENDER
                AND SENDING_DATE(MESSAGE) > CURRENT DATE - 3 DAYS)
       )
  BEGIN ATOMIC
    INSERT INTO UNHAPPY_CUSTOMERS
    VALUES ((SELECT NAME
    FROM CUSTOMERS
    WHERE EMAIL_ADDRESS = N.SENDER), N.SENDER, CURRENT DATE);
  END
```

- Definieren Sie einen Trigger, der eine Nachricht an den Generalmanager sendet, wenn derselbe Kunde mehr als einmal in die Tabelle UNHAPPY_CUSTOMERS eingefügt wird (unter der Annahme, dass eine Funktion SEND_NOTE vorhanden ist, die zwei Zeichenfolgen als Eingabe akzeptiert):

```
CREATE TRIGGER INFORM_GEN_MGR
  AFTER INSERT ON UNHAPPY_CUSTOMERS
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (1 <(SELECT COUNT(*)
            FROM UNHAPPY_CUSTOMERS
            WHERE EMAIL_ADDRESS = N.EMAIL_ADDRESS)
       )
  BEGIN ATOMIC
    VALUES(SEND_NOTE('Check customer:' CONCAT N.NAME,
                     'bigboss@vnet.ibm.com'));
  END
```

Beispiel für das Definieren von Geschäftsregeln mit Triggern

Nehmen Sie an, ein Unternehmen hat die Richtlinie, dass alle E-Mails, die sich auf Kundenbeschwerden beziehen, den Marketing-Manager Nelson in der Kopieverteilerliste zur Kenntnisnahme (CC-Liste) enthalten müssen.

Da es sich um eine Regel handelt, mag es sinnvoll erscheinen, sie als Integritätsbedingung wie im folgenden Beispiel auszudrücken (unter der Annahme, dass eine benutzerdefinierte Funktion (UDF) CC_LIST zur Überprüfung vorhanden ist):

```
ALTER TABLE ELECTRONIC_MAIL ADD
  CHECK (SUBJECT <> 'Customer complaint' OR
         CONTAINS (CC_LIST(MESSAGE), 'nelson@vnet.ibm.com') = 1)
```

Eine solche Integritätsbedingung verhindert jedoch die Einfügung von E-Mails, die sich auf Kundenbeschwerden beziehen und den Marketing-Manager nicht in der CC-Liste haben. Dies ist sicherlich im Sinne der Geschäftsregel des Unternehmens. Deren Absicht ist vielmehr, alle E-Mails an den Marketing-Manager weiterzuleiten, die sich auf Kundenbeschwerden beziehen und die nicht an den Marketing-Manager kopiert wurden. Eine solche Geschäftsregel kann nur durch einen Trigger realisiert werden, weil sie die Ausführung von Aktionen erfordert, die nicht durch deklarative Integritätsbedingungen ausgedrückt werden können. Der Trigger setzt das Vorhandensein einer Funktion SEND_NOTE voraus, die Parameter der Typen E_MAIL und 'Zeichenfolge' akzeptiert.

```
CREATE TRIGGER INFORM_MANAGER
  AFTER INSERT ON ELECTRONIC_MAIL
  REFERENCING NEW AS N
  FOR EACH ROW
```

```
WHEN (N.SUBJECT = 'Customer complaint' AND
  CONTAINS (CC_LIST(MESSAGE), 'nelson@vnet.ibm.com') = 0)
BEGIN ATOMIC
  VALUES(SEND_NOTE(N.MESSAGE, 'nelson@vnet.ibm.com'));
END
```

Beispiel für das Verhindern von Operationen an Tabellen mithilfe von Triggern

Nehmen Sie an, es soll verhindert werden, dass nicht zustellbare E-Mails in einer Tabelle mit dem Namen ELECTRONIC_MAIL gespeichert werden. Zu diesem Zweck muss die Ausführung bestimmter SQL-Anweisungen INSERT vermieden werden.

Es gibt zwei Möglichkeiten, dies zu erreichen:

- Sie können einen Vortrigger (BEFORE) definieren, der einen Fehler zurückgibt, wenn der Betreff (SUBJECT) einer E-Mail die Zeichenfolge *undelivered mail* ('nicht zugestellte E-Mail') enthält:

  ```
  CREATE TRIGGER BLOCK_INSERT
    NO CASCADE BEFORE INSERT ON ELECTRONIC_MAIL
    REFERENCING NEW AS N
    FOR EACH ROW
    WHEN (SUBJECT(N.MESSAGE) = 'undelivered mail')
    BEGIN ATOMIC
      SIGNAL SQLSTATE '85101'
        SET MESSAGE_TEXT = ('Attempt to insert undelivered mail');
       END
  ```

- Sie können eine Prüfung auf Integritätsbedingung definieren, die erzwingt, dass sich Werte der neuen Spalte SUBJECT von der Zeichenfolge *undelivered mail* unterscheiden:

  ```
  ALTER TABLE ELECTRONIC_MAIL
    ADD CONSTRAINT NO_UNDELIVERED
    CHECK (SUBJECT <> 'undelivered mail')
  ```

Kapitel 15. Sequenzen

Eine *Sequenz* ist ein Datenbankobjekt, das die automatische Generierung von Werten wie zum Beispiel Schecknummern ermöglicht. Sequenzen eignen sich ideal für die Aufgabe der Generierung eindeutiger Schlüsselwerte. Anwendungen können Sequenzen verwenden, um mögliche Probleme in Bezug auf den gemeinsamen Zugriff und die Leistung infolge von Spaltenwerten zu vermeiden, die für die Protokollierung von numerischen Werten verwendet werden. Der Vorteil von Sequenzen im Vergleich zu außerhalb der Datenbank erstellten numerischen Werten besteht darin, dass der Datenbankserver die generierten numerischen Werte protokollieren kann. Durch einen Systemabsturz mit anschließendem Wiederanlauf werden keine doppelten Zahlenwerte generiert.

Die generierten Sequenznummern haben folgende Merkmale:

- Die Werte können einen beliebigen exakten numerischen Datentyp (Exact Numeric) ohne Kommastellen aufweisen. Solche Datentypen sind: SMALLINT, BIGINT, INTEGER und DECIMAL.
- Aufeinander folgende Werte können sich um ein beliebiges angegebenes ganzzahliges Inkrement unterscheiden. Der Standardwert für das Inkrement ist 1.
- Der Wert des Zählers ist wiederherstellbar. Der Wert des Zählers wird mithilfe der Protokolle wiederhergestellt, wenn eine Recovery erforderlich ist.
- Werte können in einem Cache zwischengespeichert werden, wenn dies die Leistung verbessert. Durch das Vorabzuordnen und Speichern von Werten im Cache wird die synchrone E/A für das Protokoll verringert, wenn Werte für die Sequenz generiert werden. Im Fall eines Systemfehlers werden alle zwischengespeicherten Werte, die noch nicht verwendet wurden, als verloren eingestuft. Der für CACHE angegebene Wert ist die maximale Anzahl der Sequenzwerte, die verloren gehen könnten.

Für Sequenzen können zwei Ausdrücke verwendet werden:

- **Ausdruck NEXT VALUE:** Dieser Ausdruck gibt den nächsten Wert für die angegebene Sequenz zurück. Eine neue Sequenznummer wird generiert, wenn ein Ausdruck NEXT VALUE den Namen der Sequenz angibt. Wenn es jedoch in einer Abfrage mehrere Instanzen eines Ausdrucks NEXT VALUE gibt, die denselben Sequenznamen angeben, wird der Zähler für die Sequenz nur einmal für jede Zeile des Ergebnisses erhöht und alle Instanzen des Ausdrucks NEXT VALUE geben für jede Zeile des Ergebnisses den gleichen Wert zurück.
- **Ausdruck PREVIOUS VALUE:** Dieser Ausdruck gibt den zuletzt generierten Wert für die angegebene Sequenz für eine vorherige Anweisung innerhalb des aktuellen Anwendungsprozesses zurück. Dies bedeutet, dass der Ausdruck PREVIOUS VALUE für eine Verbindung auch dann einen konstanten Wert beibehält, wenn eine andere Verbindung den Ausdruck NEXT VALUE aufruft.

Vollständige und detaillierte Informationen und Beispiele zu diesen Ausdrücken finden Sie in der „Referenz zu Sequenzen" in *SQL Reference, Volume 1*.

Entwerfen von Sequenzen

Beim Entwerfen von Sequenzen müssen Sie die Unterschiede berücksichtigen, die zwischen Identitätsspalten und Sequenzen bestehen und feststellen, welche dieser Komponenten sich für Ihre Umgebung besser eignet. Wenn Sie sich für den Einsatz von Sequenzen entscheiden, müssen Sie sich mit den verfügbaren Optionen und Parametern vertraut machen.

Lesen Sie vor dem Entwerfen von Sequenzen die Informationen unter „Sequenzen im Vergleich zu Identitätsspalten" auf Seite 374.

Sequenzen sind einfach zu definieren und zu erstellen und verfügen darüber hinaus über eine Vielzahl zusätzlicher Optionen, die Ihnen ein höheres Maß an Flexibilität beim Generieren der Werte bieten:

- Auswahl aus einer Vielzahl von Datentypen (SMALLINT, INTEGER, BIGINT, DECIMAL)
- Ändern von Anfangswerten (START WITH)
- Ändern des Sequenzinkrements einschließlich der Angabe der Erhöhungs- oder Reduzierungswerte (INCREMENT BY)
- Definieren von Mindest- und Maximalwerten für den Beginn und das Ende der Sequenznummernvergabe (MINVALUE/MAXVALUE)
- Unterstützung des Umlaufs von Werten, sodass Sequenzen wieder von vorne begonnen werden können, oder Verhinderung dieser Umlauffunktion (CYCLE/NO CYCLE)
- Unterstützung des Cachings von Sequenzwerten zur Verbesserung der Leistung oder Verhinderung des Cachings (CACHE/NO CACHE)

Selbst nach dem Generieren der Sequenz können zahlreiche dieser Werte geändert werden. Sie können beispielsweise einen anderen Anfangswert definieren, der vom jeweiligen Wochentag abhängt. Ein weiteres praktisches Beispiel zur Verwendung von Sequenzen stellt das Generieren und die Verarbeitung von Bankschecks dar. Die Reihenfolge von Bankschecknummern ist äußerst wichtig, und es ergeben sich ernste Konsequenzen, wenn eine Serie von Sequenznummern verloren geht oder wenn es zu Fehlern kommt.

Zur Verbesserung der Leistung sollten Sie auch mit der Funktionsweise der Option CACHE vertraut sein und diese ggf. verwenden. Mit dieser Option wird für den Datenbankmanager angegeben, wie viele Sequenzwerte vom System generiert werden sollen, bevor auf den Katalog zurückgegriffen wird, um eine weitere Gruppe von Sequenzen zu generieren. Wenn keine andere Angabe erfolgt, wird für CACHE der Standardwert 20 verwendet. Bei Verwendung des Standardwerts generiert der Datenbankmanager automatisch 20 sequenzielle Werte im Speicher (1, 2, ..., 20), wenn der erste Sequenzwert angefordert wird. Sobald eine neue Sequenznummer erforderlich ist, wird der nächste Wert aus diesem Hauptspeichercache abgerufen. Wenn die in diesem Cache gespeicherten Werte verbraucht sind, generiert der Datenbankmanager die nächsten 20 Werte (21, 22, ..., 40).

Durch die Implementierung des Sequenznummerncachings muss der Datenbankmanager nicht laufend die Katalogtabellen abfragen, um den nächsten Wert abzurufen. Hierdurch wird der Systemaufwand für das Abrufen von Sequenznummern reduziert, es können sich jedoch möglicherweise Lücken in den Sequenzen ergeben, wenn ein Systemfehler auftritt oder wenn das System heruntergefahren wird. Wenn Sie beispielsweise für den Sequenzcache den Wert 100 definieren möchten, dann speichert der Datenbankmanager 100 Werte dieser Nummern im Cache und

richtet auch den Systemkatalog so ein, dass angezeigt wird, dass die nächste Wertesequenz bei 201 beginnen soll. Wenn die Datenbank heruntergefahren wird, dann beginnt die nächste Gruppe von Sequenznummern bei 201. Die zwischen 101 und 200 generierten Nummern innerhalb der Gruppe von Sequenzen gehen verloren, wenn sie nicht verwendet wurden. Wenn das Auftreten von Lücken in den generierten Werten in Ihrer Anwendung nicht akzeptabel ist, müssen Sie für den Cachingwert die Einstellung NO CACHE definieren, obwohl dies einen höheren Systemaufwand zur Folge hat.

Weitere Informationen zu allen verfügbaren Optionen und den zugehörigen Werten finden Sie in den Informationen zur Anweisung CREATE SEQUENCE.

Verwalten des Sequenzverhaltens

Sie können das Verhalten von Sequenzen anpassen, sodass dieses optimal auf die Anforderungen Ihrer Anwendung zugeschnitten ist. Sie können die Attribute einer Sequenz ändern, indem Sie zum Erstellen einer neuen Sequenz die Anweisung CREATE SEQUENCE und für eine bereits vorhandene Sequenz die Anweisung ALTER SEQUENCE eingeben.

Im Folgenden sind verschiedene Attribute einer Sequenz aufgeführt, die angegeben werden können:

Datentyp
Die Klausel AS der Anweisung CREATE SEQUENCE gibt den numerischen Datentyp der Sequenz an. Der Datentyp legt die möglichen Mindest- und Maximalwerte der Sequenz fest. Die Mindest- und Maximalwerte für einen Datentyp sind in SQL- und XML-Begrenzungen aufgeführt. Sie können den Datentyp einer Sequenz nicht ändern. Stattdessen müssen Sie die Sequenz löschen, indem Sie die Anweisung DROP SEQUENCE eingeben und anschließend die Anweisung CREATE SEQUENCE mit dem neuen Datentyp eingeben.

Anfangswert
Die Klausel START WITH der Anweisung CREATE SEQUENCE gibt den Anfangswert der Sequenz an. Die Klausel RESTART WITH der Anweisung ALTER SEQUENCE setzt den Wert der Sequenz auf einen angegebenen Wert zurück.

Mindestwert
Die Klausel MINVALUE definiert den Mindestwert einer Sequenz.

Maximalwert
Die Klausel MAXVALUE definiert den Maximalwert einer Sequenz.

Inkrementwert
Die Klausel INCREMENT BY definiert den Wert, den alle NEXT VALUE-Ausdrücke zum aktuellen Wert der Sequenz hinzufügen. Um den Wert der Sequenz zu verringern, müssen Sie einen negativen Wert angeben.

Sequenzzyklus
Die Klausel CYCLE bewirkt, dass der Wert einer Sequenz bei Erreichen des Maximal- oder des Mindestwertes den zugehörigen Mindest- bzw. Maximalwert generiert, wenn der Ausdruck NEXT VALUE das nächste Mal verwendet wird.

Anmerkung: Die Klausel CYCLE sollte nur dann verwendet werden, wenn keine eindeutigen Zahlenwerte benötigt werden oder wenn gewährleistet

werden kann, dass ältere Sequenzwerte nicht mehr verwendet werden, wenn die Sequenz einen Zyklus durchläuft.

Um beispielsweise eine Sequenz mit dem Namen `id_werte` zu erstellen, die mit einem Mindestwert von 0 beginnt, für die ein Maximalwert von 1000 definiert ist und die den vorhandenen Wert bei jeder Ausführung des Ausdrucks NEXT VALUE um 2 erhöht und wieder zum Mindestwert zurückkehrt, wenn der Maximalwert erreicht ist, müssen Sie folgende Anweisung eingeben:

```
CREATE SEQUENCE id_werte
  START WITH 0
  INCREMENT BY 2
  MAXVALUE 1000
  CYCLE
```

Anwendungsleistung und Sequenzen

Ähnlich wie die Verwendung von Identitätsspalten führt auch die Verwendung von Sequenzen zum Generieren von Werten gegenüber anderen Methoden im Allgemeinen zu einer Verbesserung der Anwendungsleistung. Als Alternative zu Sequenzen können Sie auch eine einspaltige Tabelle erstellen, in der der aktuelle Wert gespeichert wird, und diesen Wert entweder mithilfe eines Triggers oder unter der Steuerung der Anwendung erhöhen. In einer verteilten Umgebung, in der Anwendungen gleichzeitig auf die einspaltige Tabelle zugreifen, können die Sperren, die gesetzt werden müssen, um den seriellen Zugriff zu erzwingen, zu erheblichen Leistungseinbußen führen.

Durch Sequenzen können die Probleme vermieden werden, die sich bei Verwendung von Sperren in Bezug auf einspaltige Tabellen ergeben. Zur Verbesserung der Antwortzeiten können die Sequenzwerte im Cache zwischengespeichert werden. Um die Leistung von Anwendungen, die mit Sequenzen arbeiten, zu maximieren, sollten Sie sicherstellen, dass in Ihren Sequenzcaches eine angemessene Menge von Sequenzwerten gespeichert werden kann. Die Klausel CACHE der Anweisungen CREATE SEQUENCE und ALTER SEQUENCE gibt die maximale Anzahl von Sequenzwerten an, die vom Datenbankmanager generiert und im Speicher abgelegt werden können.

Wenn die Sequenz Werte in einer bestimmten Reihenfolge generieren muss, ohne dass diese Reihenfolge Lücken aufgrund von Systemausfällen oder einer Inaktivierung der Datenbank aufweist, müssen Sie in der Anweisung CREATE SEQUENCE die Klauseln ORDER und NO CACHE verwenden. Die Klausel NO CACHE garantiert, dass in den generierten Werten keine Lücken enthalten sind. Dies führt zu einer gewissen Reduzierung der Anwendungsleistung, da die Sequenz bei der Generierung eines neuen Werts immer einen Eintrag in das Datenbankprotokoll schreiben muss. Beachten Sie hierbei, dass Lücken trotzdem auftreten können, wenn eine Transaktion mit einem Rollback zurückgesetzt wird und den angeforderten Sequenzwert nicht tatsächlich benutzt.

Sequenzen im Vergleich zu Identitätsspalten

Obwohl Sequenzen und Identitätsspalten in DB2-Anwendungen demselben Zweck dienen, besteht zwischen diesen beiden Elementen ein wichtiger Unterschied. Eine Identitätsspalte generiert automatisch Werte für eine Spalte in einer einzelnen Tabelle und verwendet hierzu das Dienstprogramm LOAD. Eine Sequenz generiert auf Anforderung sequenzielle Werte, die in einer beliebigen SQL-Anweisung benutzt werden können. Hierzu wird die Anweisung CREATE SEQUENCE verwendet.

Identitätsspalten

Identitätsspalten ermöglichen dem Datenbankmanager das automatische Generieren eines eindeutigen numerischen Wertes für jede Zeile, die der Tabelle hinzugefügt wird. Wenn Sie eine Tabelle erstellen, wobei Sie jede der Tabelle hinzugefügte Zeile eindeutig identifizieren müssen, können Sie der Tabellendefinition eine Identitätsspalte hinzufügen und dies in der Anweisung CREATE TABLE angeben:

```
CREATE TABLE <tabellenname>
  (<spaltenname 1> INT,
   <spaltenname 2>, DOUBLE,
   <spaltenname 3> INT NOT NULL GENERATED ALWAYS AS IDENTITY
     (START WITH <wert 1>, INCREMENT BY <wert 2>))
```

In diesem Beispiel gibt die dritte Spalte die Identitätsspalte an. Eines der Attribute, das Sie definieren können, ist der Wert, der in der Spalte verwendet wird, um jede Zeile beim Hinzufügen eindeutig zu definieren. Der Wert, der im Anschluss an die Klausel INCREMENT BY angegeben wird, um welchen Wert nachfolgende Werte des Identitätsspalteninhalts für jede zur Tabelle hinzugefügte Zeile erhöht werden sollen.

Nach ihrer Erstellung können die Identitätsmerkmale mit der Anweisung ALTER TABLE geändert oder entfernt werden. Sie können die Anweisung ALTER TABLE auch verwenden, um Identitätsmerkmale für andere Spalten hinzuzufügen.

Sequenzen

Sequenzen ermöglichen das automatische Generieren von Werten. Sequenzen eignen sich ideal für die Aufgabe der Generierung eindeutiger Schlüsselwerte. Anwendungen können Sequenzen benutzen, um mögliche Probleme in Bezug auf den gemeinsamen Zugriff und die Leistung infolge der Generierung eines eindeutigen Zählers mit anderen Methoden zu vermeiden. Im Gegensatz zu einer Identitätsspalte ist eine Sequenz weder an eine bestimmte Tabellenspalte noch an eine eindeutige Tabellenspalte gebunden, die gleichzeitig die einzige Zugriffsmöglichkeit darstellt.

Eine Sequenz kann erstellt und zu einem späteren Zeitpunkt geändert werden, sodass diese Werte generiert, indem Werte entweder ohne Einschränkung erhöht oder reduziert werden oder indem ein benutzerdefinierter Grenzwert verwendet wird, nach dessen Erreichen die Verarbeitung gestoppt wird. Alternativ hierzu kann auch ein benutzerdefinierter Grenzwert verwendet werden, nach dessen Erreichen die Verarbeitung in Form eines Zyklus wieder von vorne beginnt. Sequenzen werden nur in Einzelpartitionsdatenbanken unterstützt.

Das folgende Beispiel zeigt, wie eine Sequenz mit dem Namen `orderseq` erstellt wird:

```
CREATE SEQUENCE orderseq
        START WITH 1
        INCREMENT BY 1
        NOMAXVALUE
        NOCYCLE
  CACHE 50
```

In diesem Beispiel beginnt die Sequenz bei 1 und wird ohne Begrenzung immer um den Wert 1 erhöht. Es gibt keinen Grund, zum Anfang zurückzukehren und wieder mit 1 zu beginnen, da keine obere Begrenzung zugeordnet wurde. Der Parameter CACHE gibt die maximale Anzahl von Sequenzwerten an, die vom Datenbankmanager vorab zugeordnet und im Speicher gehalten werden.

Erstellen von Sequenzen

Geben Sie zur Erstellung von Sequenzen die Anweisung CREATE SEQUENCE ein. Im Gegensatz zu einem Identitätsspaltenattribut ist eine Sequenz weder an eine bestimmte Tabellenspalte noch an eine eindeutige Tabellenspalte gebunden, die gleichzeitig die einzige Zugriffsmöglichkeit darstellt.

Die Positionen, an denen Ausdrücke NEXT VALUE oder PREVIOUS VALUE verwendet werden können, unterliegen verschiedenen Einschränkungen. Eine Sequenz kann erstellt oder geändert werden, sodass mit einer der folgenden Methoden Werte generiert werden:

- Monoton erhöhen oder vermindern (d. h. um einen konstanten Betrag ändern) ohne Begrenzung
- Monoton erhöhen oder vermindern bis zu einer benutzerdefinierten Begrenzung oder einem benutzerdefinierten Stopp
- Monoton erhöhen oder vermindern bis zu einer benutzerdefinierten Begrenzung und regelmäßiges Zurücksetzen an den Anfang zum erneuten Start

Anmerkung: Beim Wiederherstellen von Datenbanken, die mit Sequenzen arbeiten, sollten Sie besonders vorsichtig vorgehen: Wenn bei Sequenzwerten, die außerhalb der Datenbank verwendet werden, z. B. Sequenznummern für Bankschecks, die Datenbank auf einen Status vor dem Auftreten des Datenbankfehlers zurückgesetzt wird, dann kann dies bei bestimmten Sequenzen zur Generierung doppelter Werte führen. Um mögliche Doppelwerte zu vermeiden, sollten Datenbanken mit Sequenzwerten, die außerhalb der Datenbank verwendet werden, nicht in einem Zustand wiederhergestellt werden, in dem sich diese zu einem früheren Zeitpunkt befunden haben.

Geben Sie zum Erstellen einer Sequenz mit dem Namen `order_seq`, für die bei allen Optionen die Standardwerte benutzt werden, in einem Anwendungsprogramm oder mithilfe der entsprechenden dynamischen SQL-Anweisungen die folgende Anweisung ein:

```
CREATE SEQUENCE order_seq
```

Diese Sequenz beginnt mit 1 und wird ohne Begrenzung immer um den Wert 1 erhöht.

Dieses Beispiel kann z. B. für die Darstellung der Verarbeitungsabläufe bei einer Reihe von Bankschecks verwendet werden, deren Nummern bei 101 beginnen und bis 200 reichen. Die erste Reihe hat Nummern zwischen 1 und 100. Die Sequenz beginnt bei 101 und wird jeweils um den Wert 1 erhöht, bis der obere Grenzwert von 200 erreicht ist. Außerdem wurde NOCYCLE angegeben, sodass keine doppelten Schecknummern erzeugt werden können. Die dem Parameter CACHE zugeordnete Anzahl gibt die maximale Anzahl der Sequenzwerte an, die der Datenbankmanager vorab zuordnet und speichert.

```
CREATE SEQUENCE order_seq
   START WITH 101
   INCREMENT BY 1
   MAXVALUE 200
   NOCYCLE
   CACHE 25
```

Weitere Informationen zu dieser und weiteren Optionen sowie zu den geltenden Berechtigungsvoraussetzungen finden Sie in den Informationen zur Anweisung CREATE SEQUENCE.

Generieren sequenzieller Werte

Das Generieren sequenzieller Werte stellt bei der Entwicklung von Datenbankanwendungen häufig ein Problem dar. Die beste Lösung dieses Problems besteht in der Verwendung von Sequenzen und Sequenzausdrücken in SQL. Jede *Sequenz* stellt ein eindeutig benanntes Datenbankobjekt dar, auf das ausschließlich über Sequenzausdrücke zugegriffen werden kann.

Es gibt zwei *Sequenzausdrücke*: Ausdruck PREVIOUS VALUE und Ausdruck NEXT VALUE. Der Ausdruck PREVIOUS VALUE gibt den Wert zurück, der für die angegebene Sequenz im Anwendungsprozess zuletzt generiert wurde. NEXT VALUE-Ausdrücke, die in derselben Anweisung wie der Ausdruck PREVIOUS VALUE aufgeführt sind, wirken sich nicht auf den vom Ausdruck PREVIOUS VALUE in dieser Anweisung generierten Wert aus. Der Sequenzausdruck NEXT VALUE erhöht den Wert der Sequenz und gibt den neuen Wert der Sequenz zurück.

Geben Sie zur Erstellung einer Sequenz die Anweisung CREATE SEQUENCE ein. Um beispielsweise eine Sequenz mit dem Namen `id_werte` zu erstellen und hierbei die Standardattribute zu verwenden, müssen Sie die folgende Anweisung eingeben:

```
CREATE SEQUENCE id_werte
```

Geben Sie zum Generieren des ersten Werts in der Anwendungssitzung für die Sequenz eine Anweisung VALUES mit dem Ausdruck NEXT VALUE ein:

```
VALUES NEXT VALUE FOR id_werte

1
-----------
          1

            1 Satz/Sätze ausgewählt.
```

Um den Wert einer Spalte mit dem nächsten Wert der Sequenz zu aktualisieren, müssen Sie den Ausdruck NEXT VALUE wie folgt in die Anweisung UPDATE aufnehmen:

```
UPDATE staff
  SET id = NEXT VALUE FOR id_werte
  WHERE id = 350
```

Um eine neue Zeile in eine Tabelle einzufügen und hierbei den nächsten Wert der Sequenz zu benutzen, müssen Sie den Ausdruck NEXT VALUE wie folgt in die Anweisung INSERT aufnehmen:

```
INSERT INTO staff (id, name, dept, job)
  VALUES (NEXT VALUE FOR id_werte, 'Kandil', 51, 'Mgr')
```

Ermitteln der Bedingungen zur Verwendung von Identitätsspalten oder Sequenzen

Zwischen Identitätsspalten und Sequenzen gibt es sowohl Gemeinsamkeiten als auch Unterschiede. Die Merkmale von Identitätsspalten und Sequenzen können bei der Analyse und beim Entwurf von Datenbanken und Anwendungen verwendet werden.

Abhängig vom Design Ihrer Datenbank und der Anwendungen, die diese Datenbank nutzen, können Sie anhand der folgenden Merkmale feststellen, wann Identitätsspalten und wann Sequenzen verwendet werden sollten.

Merkmale von Identitätsspalten

- Eine Identitätsspalte generiert automatisch Werte für eine einzelne Tabelle.
- Wenn eine Identitätsspalte als GENERATED ALWAYS definiert ist, werden die verwendeten Werte immer vom Datenbankmanager generiert. Es ist nicht zulässig, dass Anwendungen bei der Änderung des Inhalts der Tabelle eigene Werte bereitstellen.
- Nach dem Einfügen einer Zeile kann der generierte Identitätswert entweder mit der Funktion IDENTITY_VAL_LOCAL() oder durch Rückauswahl der Identitätsspalte aus der Einfügeoperation mithilfe der Anweisung SELECT FROM INSERT abgerufen werden.
- Zum Generieren von Identitätswerten (IDENTITY-Werten) kann das Dienstprogramm LOAD eingesetzt werden.

Merkmale von Sequenzen

- Sequenzen sind an keine bestimmte Tabelle gebunden.
- Sequenzen generieren sequenzielle Werte, die in einer beliebigen SQL- oder XQuery-Anweisung verwendet werden können.

  Da eine Sequenz von jeder Anwendung verwendet werden kann, gibt es zwei Ausdrücke, die das Abrufen des nächsten Wertes in der angegebenen Sequenz und des Wertes, der vor der gerade ausgeführten Anweisung generiert wurde, steuern. Der Ausdruck PREVIOUS VALUE gibt den zuletzt generierten Wert für die angegebene Sequenz für eine vorherige Anweisung innerhalb der aktuellen Sitzung zurück. Der Ausdruck NEXT VALUE gibt den nächsten Wert für die angegebene Sequenz zurück. Die Verwendung dieser Ausdrücke ermöglicht es, denselben Wert in verschiedenen SQL- und XQuery-Anweisungen innerhalb mehrerer Tabellen zu verwenden.

Ändern von Sequenzen

Ändern Sie die Attribute einer vorhandenen Sequenz mit der Anweisung ALTER SEQUENCE.

Die folgenden Attribute der Sequenz können geändert werden:

- Ändern des Inkrements zwischen künftigen Werten
- Einrichten neuer Minimal- oder Maximalwerte
- Ändern der Anzahl zwischengespeicherter Sequenznummern
- Ändern der Anweisung, ob die Sequenz einen Zyklus ausführen soll oder nicht
- Ändern der Anweisung, ob die Sequenznummern in der Anforderungsreihenfolge generiert werden müssen
- Neustarten der Sequenz

Es gibt zwei Aufgaben, die nicht Teil der Erstellung der Sequenz sind. Dies sind:

- **RESTART:** Setzt die Sequenz auf den bei der Erstellung implizit oder explizit als Anfangswert angegebenen Wert zurück.
- **RESTART WITH <numerische-konstante>:** Setzt die Sequenz auf den exakten Wert der numerischen Konstanten zurück. Die numerische Konstante kann ein beliebiger positiver oder negativer Wert, der von einem etwaigen Dezimalzeichen keine anderen Ziffern als Nullen aufweisen darf.

Nach dem Neustarten einer Sequenz oder dem Wechsel zu CYCLE können doppelte Sequenznummern auftreten. Von der Anweisung ALTER SEQUENCE sind nur zukünftige Sequenznummern betroffen.

Der Datentyp einer Sequenz kann nicht geändert werden. Sie müssen stattdessen die aktuelle Sequenz löschen und anschließend eine neue Sequenz unter Angabe des neuen Datentyps erstellen.

Alle im Cache gespeicherten Sequenzwerte, die vom Datenbankmanager nicht verwendet werden, gehen beim Ändern einer Sequenz verloren.

Anzeigen von Sequenzdefinitionen

Zum Anzeigen der Verweisinformationen, die einer Sequenz zugeordnet sind, oder zum Anzeigen der Sequenz selbst können Sie die Anweisung VALUES mit der Option PREVIOUS VALUE benutzen.

Zum Anzeigen des aktuellen Werts der Sequenz müssen Sie eine Anweisung VALUES mit dem Ausdruck PREVIOUS VALUE eingeben:

```
VALUES PREVIOUS VALUE FOR id_werte

1
-----------
          1

            1 Satz/Sätze ausgewählt.
```

Sie können den aktuellen Wert der Sequenz wiederholt abrufen. Der von der Sequenz zurückgegebene Wert ändert sich erst, wenn Sie den Ausdruck NEXT VALUE eingeben. Im folgenden Beispiel gibt der Ausdruck PREVIOUS VALUE den Wert 1 zurück, bis mit dem Ausdruck NEXT VALUE in der aktuellen Verbindung der Wert der Sequenz erhöht wird:

```
VALUES PREVIOUS VALUE FOR id_werte

1
-----------
          1

            1 Satz/Sätze ausgewählt.
VALUES PREVIOUS VALUE FOR id_werte

1
-----------
          1

            1 Satz/Sätze ausgewählt.
VALUES NEXT VALUE FOR id_werte

1
-----------
          2

            1 Satz/Sätze ausgewählt.
VALUES PREVIOUS VALUE FOR id_werte

1
-----------
          2

            1 Satz/Sätze ausgewählt.
```

Dies gilt auch dann, wenn eine andere Verbindung zur gleichen Zeit Sequenzwerte liest.

Löschen von Sequenzen

Verwenden Sie zum Löschen einer Sequenz die Anweisung DROP.

Beim Löschen von Sequenzen muss die Berechtigungs-ID der Anweisung über die Berechtigung SYSADM oder DBADM verfügen.

Eine bestimmte Sequenz kann wie folgt gelöscht werden:

```
DROP SEQUENCE <sequenzname>
```

Dabei ist <sequenzname> der Name der zu löschenden Sequenz, der den impliziten bzw. expliziten Schemanamen enthält, um eine vorhandene Sequenz genau anzugeben.

Vom System erstellte Sequenzen für Identitätsspalten können nicht mit der Anweisung DROP SEQUENCE gelöscht werden.

Wenn eine Sequenz gelöscht wird, werden alle Zugriffsrechte für die Sequenz ebenfalls gelöscht.

Beispiele zur Codierung von Sequenzen

Zahlreiche Anwendungen, die geschrieben werden, benötigen eine Sequenznummer, mit der Rechnungsnummern, Kundennummern und weitere Objekte protokolliert werden können, die um den Wert eins erhöht werden, sobald ein neues Element erforderlich ist. Der Datenbankmanager kann Werte in einer Tabelle mithilfe von Identitätsspalten automatisch erhöhen. Obwohl dieses Verfahren bei einzelnen Tabellen zufriedenstellend funktioniert, stellt es nicht die komfortabelste Möglichkeit zum Generieren eindeutiger Werte dar, die in mehreren Tabellen verwendet werden müssen.

Das *Sequenzobjekt* ermöglicht Ihnen die Erstellung eines Wertes, der unter Programmiersteuerung erhöht wird und in zahlreichen Tabellen verwendet werden kann. Das folgende Beispiel zeigt eine Sequenznummer, die für Kundennummern erstellt wird und für die der Datentyp Integer verwendet wird:

```
CREATE SEQUENCE kundennummer AS INTEGER
```

Standardmäßig beginnt die Sequenznummer bei eins und erhöht sich jeweils um den Wert eins. Sie weist den Datentyp INTEGER auf. Die Anwendung ruft mit der Funktion NEXT VALUE den nächsten Wert innerhalb der Sequenz ab. Diese Funktion generiert den nächsten Wert für die Sequenz, der dann für nachfolgende SQL-Anweisungen verwendet werden kann:

```
VALUES NEXT VALUE FOR kundennummer
```

Anstatt mit der Funktion VALUES die nächste Nummer zu generieren, kann der Programmierer diese Funktion auch in einer Anweisung INSERT verwenden. Wenn beispielsweise die erste Spalte der Kundentabelle die Kundennummer enthält, dann kann die folgende Anweisung INSERT definiert werden:

```
INSERT INTO kunden VALUES
  (NEXT VALUE FOR kundennummer, 'kommentar', ...)
```

Wenn die Sequenznummer für Einfügungen in andere Tabellen verwendet werden muss, dann kann die Funktion PREVIOUS VALUE benutzt werden, um den zuvor generierten Wert abzurufen. Wenn beispielsweise die soeben erstellte Kunden-

nummer für einen nachfolgenden Rechnungsdatensatz verwendet werden soll, dann umfasst der SQL-Code die Funktion PREVIOUS VALUE:

```
INSERT INTO rechnungen
  (34,PREVIOUS VALUE FOR kundennummer, 234.44, ...)
```

Die Funktion PREVIOUS VALUE kann innerhalb der Anwendung mehrfach verwendet werden und gibt nur den Wert zurück, der von dieser Anwendung zuletzt generiert wurde. Möglicherweise haben nachfolgende Transaktionen die Sequenz bereits auf einen anderen Wert erhöht, Sie erhalten jedoch immer die zuletzt generierte Zahl.

Sequenzverweis

sequenzverweis:

next-value-ausdruck:

previous-value-ausdruck:

NEXT VALUE FOR *sequenzname*
Ein Ausdruck NEXT VALUE generiert den nächsten Wert für die durch *sequenzname* angegebene Sequenz und gibt ihn zurück.

PREVIOUS VALUE FOR *sequenzname*
Ein Ausdruck PREVIOUS VALUE gibt den zuletzt generierten Wert für die angegebene Sequenz für eine vorherige Anweisung innerhalb des aktuellen Anwendungsprozesses zurück. Auf diesen Wert kann wiederholt verwiesen werden, indem Ausdrücke PREVIOUS VALUE verwendet werden, die den Namen der Sequenz angeben. Es können mehrere Instanzen von Ausdrücken PREVIOUS VALUE, die denselben Sequenznamen angeben, in einer einzigen Anweisung enthalten sein: Sie alle geben denselben Wert zurück. In einer Umgebung mit partitionierten Datenbanken ist es möglich, dass ein Ausdruck PREVIOUS VALUE nicht den zuletzt generierten Wert zurückgibt.

Ein Ausdruck PREVIOUS VALUE kann nur verwendet werden, wenn im aktuellen Anwendungsprozess entweder in der aktuellen oder in einer vorherigen Transaktion ein Ausdruck NEXT VALUE mit demselben Sequenznamen verwendet wurde (SQLSTATE-Wert 51035).

Anmerkungen

- Für eine Sequenz wird ein neuer Wert generiert, wenn ein Ausdruck NEXT VALUE den Namen dieser Sequenz angibt. Wenn es jedoch in einer Abfrage mehrere Instanzen eines Ausdrucks NEXT VALUE gibt, die denselben Sequenznamen angeben, wird der Zähler für die Sequenz nur einmal für jede Zeile des Ergebnisses erhöht und alle Instanzen des Ausdrucks NEXT VALUE geben für eine Zeile des Ergebnisses den gleichen Wert zurück.

- Derselbe Sequenzwert kann als eindeutiger Schlüsselwert in zwei separaten Tabellen verwendet werden, indem der Sequenzwert durch einen Ausdruck NEXT VALUE für die erste Zeile (dadurch wird der Sequenzwert generiert) und durch einen Ausdruck PREVIOUS VALUE für die anderen Zeilen (die Instanz von PREVIOUS VALUE bezieht sich auf den zuletzt in der aktuellen Sitzung generierten Sequenzwert) angegeben wird, wie im folgenden Beispiel gezeigt:

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- Ausdrücke mit NEXT VALUE und PREVIOUS VALUE können an folgenden Stellen angegeben werden:
  - Anweisung SELECT oder SELECT INTO (innerhalb der Klausel SELECT, sofern die Anweisung kein Schlüsselwort DISTINCT, keine Klausel GROUP BY oder ORDER BY und keines der Schlüsselwörter UNION, INTERSECT oder EXCEPT enthält)
  - Anweisung INSERT (in einer Klausel VALUES)
  - Anweisung INSERT (in der Klausel SELECT des Fullselects)
  - Anweisung UPDATE (in der Klausel SET, entweder gesuchte oder positionierte Anweisung UPDATE, mit der Ausnahme, dass der Ausdruck NEXT VALUE nicht in der Klausel SELECT des Fullselects eines Ausdrucks in der Klausel SET angegeben werden kann)
  - Anweisung SET variable (außer in der Klausel SELECT des Fullselects eines Ausdrucks; ein Ausdruck NEXT VALUE kann in einem Trigger angegeben werden, ein Ausdruck PREVIOUS VALUE hingegen nicht)
  - Anweisung VALUES INTO (in der Klausel SELECT des Fullselects eines Ausdrucks)
  - Anweisung CREATE PROCEDURE (im Routinenhauptteil einer SQL-Prozedur)
  - Anweisung CREATE TRIGGER in der ausgelösten Aktion (es kann ein Ausdruck NEXT VALUE angegeben werden, ein Ausdruck PREVIOUS VALUE hingegen nicht)
- Ausdrücke mit NEXT VALUE und PREVIOUS VALUE können an folgenden Stellen nicht angegeben werden (SQLSTATE-Wert 428F9):
  - Joinbedingung eines vollständigen Outer Joins
  - DEFAULT-Wert für eine Spalte in einer Anweisung CREATE oder ALTER TABLE
  - Definition für generierte Spalte in einer Anweisung CREATE oder ALTER TABLE
  - Definition für Übersichtstabelle in einer Anweisung CREATE TABLE oder ALTER TABLE
  - Bedingung einer Prüfung auf Integritätsbedingung (CHECK)
  - Anweisung CREATE TRIGGER (ein Ausdruck NEXT VALUE kann angegeben werden, jedoch kein Ausdruck PREVIOUS VALUE)
  - Anweisung CREATE VIEW
  - Anweisung CREATE METHOD
  - Anweisung CREATE FUNCTION
  - Argumentliste eines XMLQUERY-, XMLEXISTS- oder XMLTABLE-Ausdrucks
- Darüber hinaus kann ein Ausdruck NEXT VALUE nicht an folgenden Stellen angegeben werden (SQLSTATE-Wert 428F9):

  - CASE-Ausdruck
  - Parameterliste einer Spaltenfunktion
  - Unterabfrage in einem anderen Kontext als denen, die oben als zulässig angegeben sind
  - Anweisung SELECT, für die das äußere SELECT einen Operator DISTINCT enthält
  - Joinbedingung eines Joins
  - Anweisung SELECT, für die das äußere SELECT eine Klausel GROUP BY enthält
  - Anweisung SELECT, für die das äußere SELECT mit einer anderen Anweisung SELECT durch den Gruppenoperator UNION, INTERSECT oder EXCEPT kombiniert wird
  - Verschachtelter Tabellenausdruck
  - Parameterliste einer Tabellenfunktion
  - Klausel WHERE der äußersten Anweisung SELECT oder einer Anweisung DELETE bzw. UPDATE
  - Klausel ORDER BY der äußersten Anweisung SELECT
  - Klausel SELECT des Fullselects eines Ausdrucks, in der Klausel SET einer Anweisung UPDATE
  - Anweisung IF, WHILE, DO ... UNTIL oder CASE in einer SQL-Routine
- Wenn ein Wert für eine Sequenz generiert wird, ist dieser Wert verbraucht. Bei der nächsten Anforderung eines Werts wird ein neuer Wert generiert. Dies gilt auch für den Fall, dass die Anweisung, die den Ausdruck NEXT VALUE enthält, fehlschlägt oder durch Rollback rückgängig gemacht wird.

  Wenn eine Anweisung INSERT einen Ausdruck NEXT VALUE in der VALUES-Liste für die Spalte enthält und während der Ausführung dieser Anweisung INSERT ein Fehler auftritt (sei es Fehler beim Generieren des nächsten Sequenzwerts oder ein Fehler bei einem Wert für eine andere Spalte), wird ein Einfügefehler (SQLSTATE-Wert 23505) zurückgegeben, und der generierte Wert für die Sequenz wird als verbraucht betrachtet. In einigen Fällen kann ein erneutes Absetzen derselben Anweisung INSERT zum Erfolg führen.

  Betrachten Sie zum Beispiel einen Fehler, der infolge eines vorhandenen eindeutigen Index für die Spalte auftritt, für die ein Ausdruck NEXT VALUE verwendet wurde, weil der generierte Sequenzwert bereits im Index vorhanden ist. Es ist möglich, dass der nächste für die Sequenz generierte Wert ein Wert ist, der nicht im Index vorhanden ist, sodass die nachfolgende INSERT-Operation erfolgreich ausgeführt würde.
- Wenn beim Generieren eines Werts für eine Sequenz der Maximalwert für die Sequenz überschritten wird (bzw. bei einer absteigenden Sequenz der Minimalwert unterschritten wird) und Zyklen nicht zulässig sind, tritt ein Fehler auf (SQLSTATE-Wert 23522). In diesem Fall könnte der Benutzer die Sequenz ändern (ALTER), um den Bereich der akzeptablen Werte zu erweitern. Alternativ könnte er Zyklen für die Sequenz zulassen oder die Sequenz löschen (DROP) und eine neue Sequenz mit einem anderen Datentyp erstellen (CREATE), der einen größeren Bereich von Werten besitzt.

  Eine Sequenz kann zum Beispiel mit dem Datentyp SMALLINT definiert worden sein, sodass sie schließlich keine zuweisbaren Werte mehr generieren kann. Eine solche Sequenz können Sie löschen und erneut erstellen, um sie mit dem Datentyp INTEGER zu definieren.
- Ein Verweis auf einen Ausdruck NEXT VALUE in der Anweisung SELECT eines Cursors bezieht sich auf einen Wert, der für eine Zeile der Ergebnistabelle gene-

riert wird. Für einen Ausdruck NEXT VALUE wird für jede Zeile, die aus der Datenbank abgerufen wird, ein Sequenzwert generiert. Wenn auf dem Client eine Zeilenblockung stattfindet, können Werte auf dem Server vor der Verarbeitung der Anweisung FETCH generiert worden sein. Dies kann geschehen, wenn eine Blockung der Zeilen der Ergebnistabelle stattfindet. Wenn die Clientanwendung nicht explizit alle Zeilen abruft (FETCH), die in der Datenbank gespeichert sind, werden für die Anwendung nicht die Ergebnisse aller generierten Sequenzwerte sichtbar (nämlich die für die gespeicherten Zeilen nicht, die nicht zurückgegeben wurden).

- Ein Verweis auf einen Ausdruck PREVIOUS VALUE in der Anweisung SELECT eines Cursors bezieht sich auf einen Wert, der für die angegebene Sequenz vor dem Öffnen des Cursors generiert wurde. Das Schließen des Cursors kann sich jedoch auf die Werte auswirken, die durch PREVIOUS VALUE für die angegebene Sequenz in nachfolgenden Anweisungen oder sogar, falls der Cursor erneut geöffnet wird, in derselben Anweisung zurückgegeben werden. Dies wäre zum Beispiel der Fall, wenn die Anweisung SELECT des Cursors einen Verweis auf einen Ausdruck NEXT VALUE für denselben Sequenznamen enthielte.
- ***Kompatibilitäten***
  - Aus Gründen der Kompatibilität mit früheren Versionen von DB2 bestehen folgende Möglichkeiten:
    - Anstelle von NEXT VALUE und PREVIOUS VALUE können auch NEXTVAL und PREVVAL angegeben werden.
  - Aus Gründen der Kompatibilität mit IBM IDS bestehen folgende Möglichkeiten:
    - Anstelle von NEXT VALUE FOR *sequenzname* kann auch *sequenzname*.NEXTVAL angegeben werden.
    - Anstelle von PREVIOUS VALUE FOR *sequenzname* kann auch *sequenzname*.CURRVAL angegeben werden.

Beispiele

Nehmen Sie an, es gibt eine Tabelle mit dem Namen ″order″ und eine Sequenz mit dem Namen ″order_seq″ wird folgendermaßen erstellt:

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

Die folgenden Beispiele zeigen, wie eine Folgenummer mit der Sequenz ″order_seq″ durch einen Ausdruck NEXT VALUE erstellt werden kann:

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

oder

```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

oder

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

Kapitel 16. Sichten

Eine *Sicht* bietet eine effektive Methode zur Darstellung von Daten, ohne sie pflegen zu müssen. Eine Sicht ist keine wirkliche Tabelle und erfordert keine permanente Speicherung. Es wird eine „virtuelle Tabelle" erstellt und verwendet.

Eine *Sicht* bietet eine alternative Möglichkeit zur Darstellung der Daten in einer oder auch in mehreren Tabellen. Sie stellt eine benannte Spezifikation einer Ergebnistabelle dar. Die Spezifikation ist eine Anweisung SELECT, die immer dann ausgeführt wird, wenn in einer SQL-Anweisung auf die Sicht Bezug genommen wird. Eine Sicht verfügt ebenso wie eine Tabelle über Spalten und Zeilen. Alle Sichten können ebenso wie Tabellen zum Abrufen von Daten verwendet werden. Ob eine Sicht in einer Einfüge-, Aktualisierungs- oder Löschoperation verwendet werden kann, hängt von ihrer Definition ab.

Eine Sicht kann alle oder einige der Spalten oder Zeilen der Tabelle enthalten, auf der sie basiert. Zum Beispiel können Sie eine Tabelle mit Abteilungsdaten (DEPARTMENT) und eine Tabelle mit Mitarbeiterdaten (EMPLOYEE) in einer Sicht verknüpfen, sodass Sie alle Mitarbeiter in einer bestimmten Abteilung auflisten können.

Abb. 28 zeigt die Beziehung zwischen Tabellen und Sichten.

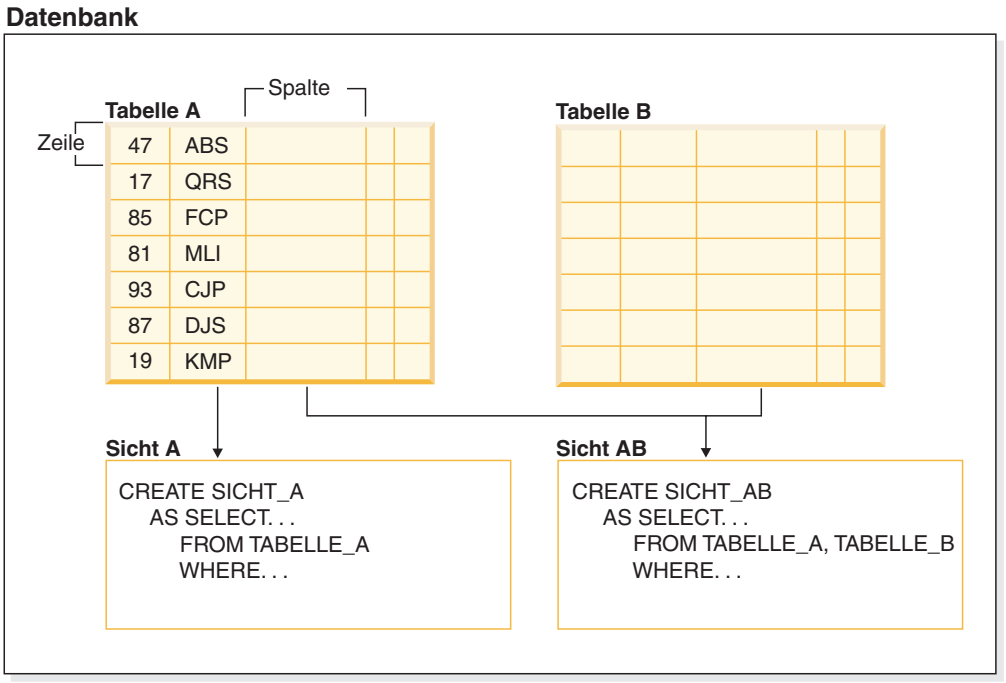


Abbildung 28. Beziehung zwischen Tabellen und Sichten

Sie können Sichten verwenden, um den Zugriff auf sensible Daten zu steuern, da mit ihnen ein bestimmter Datenbestand für unterschiedliche Benutzer in verschiedenen Darstellungen angezeigt werden kann. Beispiel: Mehrere Benutzer greifen auf eine Tabelle mit Mitarbeiterdaten zu. Ein Manager kann dann die Daten zu seinen Mitarbeitern, jedoch nicht die Daten der Mitarbeiter einer anderen Abteilung anzeigen. Ein Personalmitarbeiter kann die Einstellungstermine aller Mitarbeiter

anzeigen, jedoch nicht ihre Gehälter, ein Buchhaltungsmitarbeiter hingegen die Gehälter, jedoch nicht die Einstellungstermine. Jeder dieser Benutzer arbeitet mit einer Sicht, die von der Tabelle abgeleitet wurde. Jede Sicht erscheint für den Benutzer wie eine eigenständige Tabelle und verfügt über einen eigenen Namen.

Wenn die Spalte einer Sicht direkt aus der Spalte einer Basistabelle abgeleitet wird, dann übernimmt diese Sichtspalte alle Integritätsbedingungen, die auch für die entsprechende Spalte der Tabelle gelten. Wenn eine Sicht beispielsweise einen Fremdschlüssel der zugehörigen Tabelle enthält, dann gelten für Einfüge- und Aktualisierungsoperationen in dieser Sicht die gleichen referenziellen Integritätsbedingungen wie für die Tabelle. Wenn es sich bei der Tabelle einer Sicht um eine übergeordnete Tabelle handelt, dann gelten für Lösch- und Aktualisierungsoperationen in dieser Sicht ebenfalls die gleichen Regeln wie für Lösch- und Aktualisierungsoperationen in der Tabelle.

Eine Sicht kann den Datentyp aller Spalten aus der Ergebnistabelle ableiten oder die Typzuordnung auf der Basis der Attribute vornehmen, die für einen benutzerdefinierten strukturierten Typ gelten. Derartige Sichten werden als *typisierte Sicht* bezeichnet. Ähnlich wie bei einer typisierten Tabelle kann eine typisierte Sicht Teil einer Sichtenhierarchie sein. Eine *untergeordnete Sicht* übernimmt die Spalten der zugehörigen *übergeordneten Sicht*. Der Begriff *untergeordnete Sicht* bezieht sich auf eine typisierte Sicht und auf alle typisierten Sichten, die sich innerhalb der Sichtenhierarchie unter dieser befinden. Eine *korrekte untergeordnete Sicht* der Sicht V ist eine Sicht, die sich innerhalb der Hierarchie typisierter Sichten unterhalb der Sicht V befindet.

Eine Sicht kann funktionsunfähig werden. (Dies ist z. B. dann der Fall, wenn die zugehörige Tabelle gelöscht wird.) In einer derartigen Situation steht die Sicht nicht mehr für SQL-Operationen zur Verfügung.

Entwerfen von Sichten

Eine *Sicht* bietet eine alternative Möglichkeit zur Darstellung der Daten in einer oder auch in mehreren Tabellen. Sie stellt eine benannte Spezifikation einer Ergebnistabelle dar.

Die Spezifikation ist eine Anweisung SELECT, die immer dann ausgeführt wird, wenn in einer SQL-Anweisung auf die Sicht Bezug genommen wird. Eine Sicht verfügt ebenso wie eine Basistabelle über Spalten und Zeilen. Alle Sichten können ebenso wie die Tabellen zum Abrufen von Daten verwendet werden. Ob eine Sicht in einer Einfüge-, Aktualisierungs- oder Löschoperation verwendet werden kann, hängt von ihrer Definition ab.

Sichten werden abhängig von den zulässigen Operationen in verschiedene Klassen eingeteilt. Diese lauten wie folgt:

- Löschfähige Sichten
- Aktualisierungsfähige Sichten
- Einfügefähige Sichten
- Schreibgeschützte Sichten

Der Typ einer Sicht wird anhand ihrer Aktualisierungsmöglichkeiten festgelegt. Die Klassifizierung gibt die Art der SQL-Operation an, die für die Sicht ausgeführt werden darf.

Referenzielle Integritätsbedingungen und Prüfbedingungen werden unabhängig voneinander behandelt. Sie haben keine Auswirkungen auf die Klassifizierung der Sicht.

Beispielsweise ist es möglich, dass eine Zeile aufgrund einer referenziellen Integritätsbedingung nicht in eine Tabelle eingefügt werden kann. Wenn Sie eine Sicht auf der Basis dieser Tabelle erstellen, ist es auch nicht möglich, diese Zeile über die Sicht einzufügen. Wenn die Sicht jedoch alle Regeln für eine einfügefähige Sicht erfüllt, dann wird sie dennoch als einfügefähige Sicht betrachtet. Dies ist darauf zurückzuführen, dass die Einfügeeinschränkung für die Tabelle und nicht für die Sichtdefinition gilt.

Weitere Informationen zu diesem Thema finden Sie in den Informationen zur Anweisung CREATE VIEW.

Systemkatalogsichten

Der Datenbankmanager verwaltet eine Gruppe von Tabellen und Sichten, die Informationen zu den Daten enthalten, die seiner Steuerung unterliegen. Diese Tabellen und Sichten werden zusammen als *Systemkatalog* bezeichnet.

Der Systemkatalog enthält Informationen zu der logischen und physischen Struktur von Datenbankobjekten wie z. B. Tabellen, Sichten, Indizes, Paketen und Funktionen. Darüber hinaus enthält er statistische Informationen. Der Datenbankmanager stellt sicher, dass die Beschreibungen im Systemkatalog immer korrekt sind.

Die Sichten des Systemkatalogs weisen die gleichen Merkmale auf wie andere Datenbanksichten. Die in ihnen gespeicherten Daten können ebenfalls mithilfe von SQL-Anweisungen abgefragt werden. Eine Gruppe aktualisierbarer Systemkatalogsichten kann verwendet werden, um bestimmte Werte im Systemkatalog zu ändern.

Sichten mit Prüfoption

Eine Sicht, für die WITH CHECK OPTION definiert wurde, erzwingt die Verarbeitung aller Zeilen, die mit der Anweisung SELECT für diese Sicht geändert oder eingefügt wurden. Sichten mit der Prüfoption werden auch als *symmetrische Sichten* bezeichnet. Eine symmetrische Sicht, die ausschließlich die Mitarbeiter in Abteilung 10 zurückgibt, erlaubt z. B. keine Einfügeoperationen mit Mitarbeitern anderer Abteilungen. Diese Option stellt aus diesem Grund die Integrität der in der Datenbank geänderten Daten sicher und gibt einen Fehler zurück, wenn die geltende Bedingung während einer INSERT- oder UPDATE-Operation nicht beachtet wird.

Wenn in Ihrer Anwendung die gewünschten Regeln nicht in Form einer Prüfung auf Integritätsbedingungen in Tabellen definiert werden können oder die Regeln nicht für alle Verwendungsvorkommen der Daten gelten, dann gibt es eine alternative Vorgehensweise zur Integration der Regeln in die Anwendungslogik. Sie können eine Sicht der Tabelle erstellen, für die die Bedingungen, die für die Daten gelten, als Teil der angegebenen Klauseln WHERE und WITH CHECK OPTION definiert werden. Diese Sichtdefinition schränkt den Abruf der Daten auf die Gruppe ein, die für Ihre Anwendung zulässig ist. Darüber hinaus schränkt die Klausel WITH CHECK OPTION bei Sichten, die aktualisiert werden können, Aktualisierungs-, Einfüge- und Löschoperationen in den Zeilen ein, die für Ihre Anwendung gelten.

Die Klausel WITH CHECK OPTION darf in folgenden Sichten nicht angegeben werden:

- Sichten, für die die Option READ-ONLY (schreibgeschützte Sicht) definiert wurde.
- Sichten, die auf die Funktion NODENUMBER oder PARTITION, eine nicht deterministische Funktion (z. B. RAND) oder eine Funktion verweisen, für die externe Aktionen ausgeführt werden.
- Typisierte Sichten.

Beispiel 1

Im Folgenden ist ein Beispiel für eine Sichtdefinition aufgeführt, die die Klausel WITH CHECK OPTION verwendet. Diese Option ist erforderlich, um sicherzustellen, dass die Bedingung immer geprüft wird. Die Sicht stellt darüber hinaus sicher, dass für DEPT immer der Wert 10 benutzt wird. Dadurch werden die Eingabewerte für die Spalte DEPT eingeschränkt. Wenn eine Sicht verwendet wird, um einen neuen Wert einzufügen, wird die Verwendung der Klausel WITH CHECK OPTION immer erzwungen:

```
CREATE VIEW EMP_VIEW2
  (EMPNO, EMPNAME, DEPTNO, JOBTITLE, HIREDATE)
AS SELECT ID, NAME, DEPT, JOB, HIREDATE FROM EMPLOYEE
  WHERE DEPT=10
WITH CHECK OPTION;
```

Wird diese Sicht in einer Anweisung INSERT verwendet, wird die Zeile zurückgewiesen, wenn die Spalte DEPTNO nicht den Wert 10 aufweist. Sie sollten unbedingt beachten, dass während der Änderung keine Datenprüfung stattfindet, wenn die Klausel WITH CHECK OPTION nicht angegeben wurde.

Wenn diese Sicht in einer Anweisung SELECT verwendet wird, dann wird die Bedingung (Klausel WHERE) aufgerufen und die dabei generierte Tabelle enthält ausschließlich die Datenzeilen, die mit den angegebenen Werten übereinstimmen. Dies bedeutet, dass die Klausel WITH CHECK OPTION keine Auswirkungen auf das Ergebnis einer Anweisung SELECT hat.

Beispiel 2

Mithilfe einer Sicht können Sie eine Untermenge der Daten einer Tabelle für ein Anwendungsprogramm verfügbar machen und die Daten auf Gültigkeit prüfen, die eingefügt oder aktualisiert werden sollen. Eine Sicht kann Spaltennamen enthalten, die sich von den Namen der entsprechenden Spalten der Originaltabellen unterscheiden. Beispiel:

```
CREATE VIEW <name> (<spalte>, <spalte>, <spalte>)
SELECT <spaltenname> FROM <tabellenname>
      WITH CHECK OPTION
```

Beispiel 3

Durch die Verwendung von Sichten gestalten sich die Möglichkeiten der Anwendungsprogramme und Endbenutzerabfragen, auf Tabellendaten zuzugreifen, wesentlich flexibler.

Mit der folgenden SQL-Anweisung wird eine Sicht von der Tabelle EMPLOYEE erstellt, die alle Mitarbeiter der Abteilung A00 mit der Personalnummer und der Telefonnummer auflistet:

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
      AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
      WHERE WORKDEPT = 'A00'
      WITH CHECK OPTION
```

Die erste Zeile dieser Anweisung benennt die Sicht und definiert ihre Spalten. Der Name EMP_VIEW muss innerhalb seines Schemas in SYSCAT.TABLES eindeutig sein. Der Name der Sicht erscheint als Tabellenname, obwohl die Sicht keine Daten enthält. Die Sicht wird mit den Spalten DA00NAME, DA00NUM und PHONENO definiert, die den Spalten LASTNAME, EMPNO und PHONENO der Tabelle EMPLOYEE entsprechen. Die Spaltennamen werden nacheinander in der aufgeführten Reihenfolge jeweils den Spalten der mit der Anweisung SELECT angegebenen SELECT-Liste zugeordnet. Wenn Spaltennamen nicht angegeben werden, verwendet die Sicht dieselben Namen wie die Spalten der Ergebnistabelle der Anweisung SELECT.

Die zweite Zeile enthält eine Anweisung SELECT, die beschreibt, welche Werte aus der Datenbank ausgewählt werden sollen. Sie kann die Klauseln ALL, DISTINCT, FROM, WHERE, GROUP BY und HAVING enthalten. Der Name bzw. die Namen der Datenobjekte, aus denen die Spalten für die Sicht auszuwählen sind, müssen nach der Klausel FROM angegeben werden.

Beispiel 4

Die Klausel WITH CHECK OPTION gibt an, dass jede in der Sicht aktualisierte oder eingefügte Zeile an der Sichtdefinition überprüft und zurückgewiesen werden muss, wenn sie der Definition nicht entspricht. Dadurch wird die Datenintegrität erhöht, aber auch zusätzlicher Verarbeitungsaufwand verursacht. Wenn diese Klausel nicht angegeben wird, werden Einfügungen und Aktualisierungen nicht an der Sichtdefinition überprüft.

Mit der folgenden SQL-Anweisung wird dieselbe Sicht von der Tabelle EMPLOYEE erstellt, allerdings mit der Klausel SELECT AS:

```
CREATE VIEW EMP_VIEW
      SELECT LASTNAME AS DA00NAME,
                  EMPNO AS DA00NUM,
                  PHONENO
     FROM EMPLOYEE            WHERE WORKDEPT = 'A00'
      WITH CHECK OPTION
```

Hier kann die Tabelle EMPLOYEE beispielsweise Gehaltsdaten enthalten, die nicht unbedingt jeder Person zugänglich sein sollten. Die Telefonnummer des Mitarbeiters sollte dagegen allgemein verfügbar sein. In diesem Fall kann eine Sicht erstellt werden, die nur aus den Spalten für den Namen (LASTNAME) und für die Telefonnummer (PHONENO) besteht. Der Zugriff auf die Sicht könnte für die Berechtigung PUBLIC (allgemeiner Zugriff) erteilt werden, während der Zugriff auf die gesamte Tabelle EMPLOYEE auf die Personen beschränkt werden könnte, die die Berechtigung haben, Gehaltsdaten einzusehen.

Definitionen verschachtelter Sichten

Wenn eine Sicht auf einer anderen Sicht basiert, dann richtet sich die Anzahl der Vergleichselemente, die ausgewertet werden müssen, nach der Angabe von WITH CHECK OPTION.

Wenn eine Sicht ohne WITH CHECK OPTION definiert wird, dann wird die Definition der Sicht in keiner Einfüge- oder Aktualisierungsoperation für die Gültigkeitsprüfung der Daten verwendet. Wenn die Sicht jedoch direkt oder indi-

rekt von einer anderen Sicht abhängt, die mit WITH CHECK OPTION definiert wurde, dann wird die Definition dieser übergeordneten Sicht für die Prüfung aller Einfüge- und Aktualisierungsoperationen verwendet.

Wenn eine Sicht mit WITH CASCADED CHECK OPTION oder nur mit WITH CHECK OPTION (CASCADED ist der Standardwert von WITH CHECK OPTION) definiert wird, dann wird die Definition der Sicht für die Prüfung aller Einfüge- und Aktualisierungsoperationen benutzt. Darüber hinaus übernimmt die Sicht die Suchbedingungen aller aktualisierbaren Sichten, von denen die Sicht abhängig ist. Diese Bedingungen werden auch dann übernommen, wenn diese Sichten die Option WITH CHECK OPTION nicht umfassen. Dann werden die übernommenen Bedingungen multipliziert, um einer Integritätsbedingung zu entsprechen, die für alle Einfüge- oder Aktualisierungsoperationen in Bezug auf die Sicht oder alle anderen Sichten angewendet wird, die von dieser Sicht abhängig sind.

Wenn zum Beispiel die Sicht V2 auf der Sicht V1 basiert und die Prüfoption für V2 mit WITH CASCADED CHECK OPTION definiert ist, werden die Vergleichselemente beider Sichten ausgewertet, wenn für die Sicht V2 INSERT- und UPDATE-Anweisungen ausgeführt werden:

```
CREATE VIEW EMP_VIEW2 AS
  SELECT EMPNO, EMPNAME, DEPTNO FROM EMP
    WHERE DEPTNO = 10
WITH CHECK OPTION;
```

Das folgende Beispiel zeigt eine Anweisung CREATE VIEW, in der WITH CASCADED CHECK OPTION angegeben ist. Die Sicht EMP_VIEW3 wird auf der Basis der Sicht EMP_VIEW2 erstellt, die mit WITH CHECK OPTION erstellt wurde. Wenn Sie einen Datensatz in EMP_VIEW3 einfügen oder dort aktualisieren wollen, dann muss der Datensatz die Werte DEPTNO=10 und EMPNO=20 aufweisen.

```
CREATE VIEW EMP_VIEW3 AS
  SELECT EMPNO, EMPNAME, DEPTNO FROM EMP_VIEW2
    WHERE EMPNO > 20
WITH CASCADED CHECK OPTION;
```

Anmerkung: Die Verwendung der Bedingung DEPTNO=10 wird für Einfüge- oder Aktualisierungsoperationen in EMP_VIEW3 auch dann durchgesetzt, wenn EMP_VIEW2 die Angabe WITH CHECK OPTION nicht enthält.

Die Option WITH LOCAL CHECK OPTION kann auch bei der Erstellung einer Sicht angegeben werden. Wenn für eine Sicht LOCAL CHECK OPTION definiert wurde, dann wird die Definition der Sicht für die Prüfung aller Einfüge- und Aktualisierungsoperationen verwendet. Die Sicht übernimmt allerdings nicht die Suchbedingungen der aktualisierbaren Sichten, von denen diese abhängig ist.

Löschfähige Sichten

Abhängig von der Art und Weise, in der eine Sicht definiert ist, kann die zum Löschen verwendet werden. Für eine solche löschfähige Sicht kann die Anweisung DELETE erfolgreich ausgeführt werden.

Eine löschfähige Sicht muss die folgenden Regeln erfüllen:

- Jede Klausel FROM des Outer Fullselect identifiziert nur eine Tabelle (ohne die Klausel OUTER), eine löschfähige Sicht (ohne die Klausel OUTER), einen löschfähigen verschachtelten Tabellenausdruck oder löschfähigen allgemeinen Tabellenausdruck.

- Der Datenbankmanager muss in der Lage sein, die in der Tabelle zu löschenden Zeilen mithilfe der Sichtdefinition abzuleiten. Bestimmte Operationen verhindern dies jedoch.
  - Eine Zusammenfassung mehrerer Zeilen zu einer Gruppe mithilfe der Klausel GROUP BY oder bestimmter Spaltenfunktionen führt zum Verlust der ursprünglichen Zeile und dazu, dass die Sicht nicht zum Löschen verwendet werden kann.
  - Ebenso ist keine Tabelle vorhanden, aus der gelöscht werden kann, wenn die Zeilen aus einer Klausel VALUES abgeleitet werden. Auch in diesem Fall ist die Sicht nicht löschfähig.
- Der Outer Fullselect verwendet die Klauseln GROUP BY und HAVING nicht.
- Die SELECT-Liste des Outer Fullselect umfasst keine Spaltenfunktionen.
- Der Outer Fullselect verwendet keine Gruppenoperationen (UNION, EXCEPT oder INTERSECT). Eine Ausnahme bildet hierbei UNION ALL.
- Die Tabellen in den Operanden von UNION ALL dürfen nicht gleich sein, und jeder Operand muss löschbar sein.
- Die SELECT-Liste des Outer Fullselect enthält keine Klausel DISTINCT.

Eine Sicht muss alle hier aufgeführten Regeln erfüllen, um als löschfähige Sicht eingestuft zu werden. Die folgende Sicht ist z. B. löschfähig. Sie erfüllt alle Regeln für löschfähige Sichten.

```
CREATE VIEW löschfähige_sicht
  (number, date, start, end)
AS
  SELECT number, date, start, end
  FROM employee.summary
  WHERE date='01012007'
```

Einfügefähige Sichten

Einfügefähige Sichten ermöglichen Ihnen das Einfügen von Zeilen mithilfe der Sichtdefinition. Eine Sicht wird als einfügefähig bezeichnet, wenn ein INSTEAD OF-Trigger für die Einfügeoperation für die Sicht definiert wurde oder wenn mindestens eine Spalte der Sicht (unabhängig von einem INSTEAD OF-Trigger für die Aktualisierung) aktualisierbar ist und die Fullselect-Anweisung der Sicht keine Angabe UNION ALL enthält. Eine bestimmte Zeile kann in eine Sicht (einschließlich einer UNION ALL-Sicht) nur dann eingefügt werden, wenn sie die Prüfbedingungen von genau einer der zugrunde liegenden Tabellen erfüllt. Um eine Einfügung in eine Sicht vorzunehmen, die nicht aktualisierbare Spalten enthält, dürfen diese Spalten in der Spaltenliste nicht aufgeführt werden.

Die nachfolgend dargestellte Sicht ist einfügefähig. In diesem Beispiel schlägt die Einfügung der Sicht jedoch fehl. Dies ist darauf zurückzuführen, dass Spalten in der Tabelle enthalten sind, die keine Nullwerte akzeptieren. Einige dieser Spalten sind in der Sichtdefinition nicht vorhanden. Wenn Sie versuchen, mit der Sicht einen Wert einzufügen, versucht der Datenbankmanager, einen Nullwert in eine Spalte mit dem Merkmal NOT NULL einzufügen. Diese Aktion ist nicht zulässig.

```
CREATE VIEW einfügefähige_sicht
  (number, name, quantity)
AS
SELECT number, name, quantify FROM ace.supplies
```

Anmerkung: Die für die Tabelle definierten Integritätsbedingungen gelten unabhängig von den Operationen, die mit einer auf dieser Tabelle basierenden Sicht ausgeführt werden können.

Aktualisierungsfähige Sichten

Eine aktualisierungsfähige Sicht ist ein Sonderfall einer löschfähigen Sicht. Eine löschfähige Sicht wird zu einer aktualisierungsfähigen Sicht, wenn zumindest eine ihrer Spalten aktualisiert werden kann.

Eine Spalte einer Sicht ist aktualisierbar, wenn alle folgenden Regeln erfüllt werden:

- Die Sicht ist löschfähig.
- Die Spalte kann (ohne eine Dereferenzierungsoperation) in eine Spalte einer Tabelle aufgelöst werden und die Option READ ONLY wurde nicht angegeben.
- Alle zugehörigen Spalten der Operanden von UNION ALL verfügen über exakt übereinstimmende Datentypen (einschließlich Länge oder Genauigkeit und Maßstab) und übereinstimmende Standardwerte, wenn der Fullselect der Sicht UNION ALL umfasst.

Im folgenden Beispiel werden konstante Werte verwendet, die nicht aktualisiert werden können. Die Sicht ist jedoch löschfähig und mindestens eine der Spalten kann aktualisiert werden. Aus diesem Grund handelt es sich um eine aktualisierungsfähige Sicht.

```
CREATE VIEW aktualisierungsfähige_sicht
  (number, current_date, current_time, temperature)
AS
  SELECT number, CURRENT DATE, CURRENT TIME, temperature)
FROM weather.forecast
WHERE number = 300
```

Schreibgeschützte Sichten

Eine Sicht wird als *schreibgeschützt* bezeichnet, wenn sie *nicht* löschfähig, aktualisierungsfähig oder einfügefähig ist. Eine Sicht kann auch schreibgeschützt sein, wenn sie nicht mindestens einer der Regeln für löschfähige Sichten entspricht.

Die Spalte READONLY in der Katalogsicht SYSCAT.VIEWS gibt eine schreibgeschützte Sicht (R = Read-only) an.

Das folgende Beispiel stellt keine löschfähige Sicht dar, weil die Klausel DISTINCT verwendet wird und die SQL-Anweisung mehr als eine Tabelle betrifft:

```
CREATE VIEW schreibgeschützte_sicht
  (name, phone, address)
AS
SELECT DISTINCT viewname, viewphone, viewaddress
FROM employee.history adam, employer.dept sales
WHERE adam.id = sales.id
```

Erstellen von Sichten

Sichten werden aus einer oder mehreren Tabellen, Kurznamen oder Sichten abgeleitet und können beim Abrufen von Daten frei austauschbar mit Tabellen verwendet werden. Wenn Änderungen an den in einer Sicht gezeigten Daten vorgenommen werden, werden die Daten in der Tabelle selbst geändert. Die Tabelle, der Kurzname bzw. die Sicht, auf der die Sicht basieren soll, muss vorhanden sein, bevor die Sicht erstellt werden kann.

Eine Sicht kann zur Einschränkung des Zugriffs auf sensible Daten verwendet werden, während auf andere Daten ein allgemeinerer Zugriff zugelassen wird.

Beim Einfügen in eine Sicht, in der eine SELECT-Liste der Sichtdefinition direkt oder indirekt den Namen einer Identitätsspalte einer Tabelle umfasst, gelten dieselben Regeln, wie wenn die INSERT-Anweisung direkt auf die Identitätsspalte der Tabelle verweisen würde.

Über die bereits beschriebene Verwendung hinaus kann eine Sicht auch folgenden Zwecken dienen:

- Ändern einer Tabelle ohne Auswirkungen auf Anwendungsprogramme. Dies kann durch das Erstellen einer Sicht aufgrundlage einer zugrunde liegenden Tabelle geschehen. Anwendungen, die die zugrunde liegende Tabelle verwenden, werden durch die Erstellung der neuen Sicht nicht beeinflusst. Neue Anwendungen können die erstellte Sicht für andere Zwecke verwenden als jene Anwendungen, die die zugrunde liegende Tabelle verwenden.
- Summieren der Werte in einer Spalte, Auswählen der größten Werte oder Berechnen der Durchschnittswerte.
- Bereitstellen des Zugriffs auf Informationen an einer oder mehreren Datenquellen. Sie können innerhalb der Anweisung CREATE VIEW auf Kurznamen verweisen sowie Sichten für mehrere Positionen bzw. globale Sichten (die Sicht könnte Informationen aus mehreren Datenquellen auf verschiedenen Systemen verknüpfen) erstellen.

  Wenn Sie eine Sicht erstellen, die mit der Standardsyntax von CREATE VIEW auf Kurznamen verweist, erscheint eine Warnung, die darauf hinweist, dass die Berechtigungs-ID der Benutzer dieser Sicht für den Zugriff auf die zugrunde liegenden Objekte an den Datenquellen verwendet werden und nicht die Berechtigungs-ID des Erstellers dieser Sicht. Diese Warnung können Sie mit dem Schlüsselwort FEDERATED unterdrücken.

Eine typisierte Sicht basiert auf einem vordefinierten strukturierten Typ. Sie können mithilfe der Anweisung CREATE VIEW eine typisierte Sicht erstellen.

Eine Alternative zur Erstellung einer Sicht ist die Verwendung eines verschachtelten oder allgemeinen Tabellenausdrucks, um das Durchsuchen von Katalogen zu reduzieren und die Leistung zu erhöhen.

Das folgende Beispiel zeigt eine Anweisung CREATE VIEW. Die zugrunde liegende Tabelle EMPLOYEE enthält Spalten mit den Namen SALARY und COMM. Aus Sicherheitsgründen wird diese Sicht aus den Spalten ID, NAME, DEPT, JOB und HIREDATE erstellt. Darüber hinaus wird der Zugriff auf die Spalte DEPT eingeschränkt. Diese Definition zeigt nur die Informationen zu Mitarbeitern an, die zu der Abteilung mit der Abteilungsnummer (DEPTNO) 10 gehören.

```
CREATE VIEW EMP_VIEW1
  (EMPID, EMPNAME, DEPTNO, JOBTITLE, HIREDATE)
  AS SELECT ID, NAME, DEPT, JOB, HIREDATE FROM EMPLOYEE
    WHERE DEPT=10;
```

Nachdem die Sicht definiert wurde, können die Zugriffsberechtigungen angegeben werden. Dadurch wird die Datensicherheit gewährleistet, da nur Zugriff auf eine eingeschränkte Sicht der Tabelle besteht. Wie oben dargestellt, kann eine Sicht eine Klausel WHERE enthalten, um den Zugriff auf bestimmte Zeilen einzuschränken, oder eine Untergruppe der Spalten, um den Zugriff auf bestimmte Datenspalten einzuschränken.

Die Spaltennamen in der Sicht müssen nicht mit den Spaltennamen der Tabelle übereinstimmen. Der Tabellenname verfügt wie der Sichtname über ein zugeordnetes Schema.

Nachdem die Sicht definiert wurde, kann diese in Anweisungen wie zum Beispiel SELECT, INSERT, UPDATE und DELETE (mit Einschränkungen) verwendet werden. Der Datenbankadministrator kann entscheiden, ob einer bestimmten Benutzergruppe umfangreichere Zugriffsrechte für die Sicht gewährt werden sollen als für die Tabelle.

Erstellen von Sichten mit benutzerdefinierten Funktionen (UDFs)

Wenn Sie eine Sicht erstellen, die mit benutzerdefinierten Funktionen (UDFs = User-Defined Functions) arbeitet, dann verwendet diese Sicht während ihrer gesamten Lebensdauer die für sie definierte UDF. Dies gilt auch dann, wenn Sie zu einem späteren Zeitpunkt andere UDFs mit dem gleichen Namen erstellen. Wenn Sie eine neue UDF verwenden möchten, müssen Sie die Sicht erneut erstellen.

Mit der folgenden SQL-Anweisung wird eine Sicht mit einer Funktion in ihrer Definition erstellt:

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
    AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
    FROM EMPLOYEE
```

Die benutzerdefinierte Funktion PENSION berechnet die aktuellen Pensionsansprüche eines Mitarbeiters mit einer Formel, die mit den Werten des Einstellungsdatums (HIREDATE), des Geburtsdatums (BIRTHDATE), des Gehalts (SALARY) und der Sondervergütungen (BONUS) arbeitet.

Ändern typisierter Sichten

Bestimmte Merkmale einer typisierten Sicht können geändert werden, ohne dass hierzu die Sicht gelöscht und anschließend erneut erstellt werden muss. Eines dieser Merkmale bezieht sich auf das Hinzufügen eines Bereichs zu einer Verweisspalte einer typisierten Sicht.

Die Anweisung ALTER VIEW ändert eine vorhandene Definition einer typisierten Sicht durch Ändern einer Verweistypspalte zum Hinzufügen eines Bereichs. Die Anweisung DROP löscht eine typisierte Sicht. Außerdem können Sie folgende Operationen ausführen:

- Ändern des Inhalts einer typisierten Sicht mithilfe von INSTEAD OF-Triggern
- Ändern einer typisierten Sicht zum Aktivieren der Statistikerfassung

Für Änderungen, die Sie an dem Inhalt vornehmen, der einer typisierten Sicht zugrunde liegt, benötigen Sie Trigger. Andere Änderungen an einer typisierten Sicht können Sie nur durchführen, indem Sie die typisierte Sicht löschen und erneut erstellen.

Der Datentyp des Spaltennamens in der Anweisung ALTER VIEW muss REF sein (Typ des Namens der typisierten Tabelle oder des Namens der typisierten Sicht).

Andere Datenbankobjekte wie Tabellen und Indizes sind nicht betroffen, obwohl Pakete und im Cache zwischengespeicherte dynamische Anweisungen als ungültig markiert werden.

Geben Sie in der Befehlszeile Folgendes ein, um eine typisierte Sicht zu ändern:

```
ALTER VIEW <sichtname> ALTER <spaltenname>
   ADD SCOPE <name einer typisierten tabelle oder sicht>
```

Recovery funktionsunfähiger Sichten

Als funktionsunfähige Sicht wird eine Sicht bezeichnet, die für SQL-Anweisungen nicht mehr zur Verfügung steht.

Sichten können aufgrund folgender Ursachen *funktionsunfähig* werden:

- Infolge eines widerrufenen Zugriffsrechts für eine zugrunde liegende Tabelle.
- Wenn eine Tabelle, ein Aliasname oder eine Funktion gelöscht wurde.
- Wenn die übergeordnete Sicht funktionsunfähig wird. (Eine übergeordnete Sicht ist eine typisierte Sicht, auf der eine andere typisierte Sicht, d. h. eine untergeordnete Sicht, basiert.)
- Wenn die Sichten, von denen die Sichten abhängen, gelöscht werden.

Eine Recovery für eine funktionsunfähige Sicht kann auf folgende Weise durchgeführt werden:

1. Stellen Sie fest, mit welcher SQL-Anweisung die Sicht zu Anfang erstellt wurde. Diese Information können Sie der Spalte TEXT der Katalogsicht SYSCAT.VIEW entnehmen.
2. Definieren Sie das aktuelle Schema für den Inhalt der Spalte QUALIFIER.
3. Definieren Sie den Funktionspfad für den Inhalt der Spalte FUNC_PATH.
4. Erstellen Sie die Sicht erneut, indem Sie die Anweisung CREATE VIEW mit demselben Sichtnamen und derselben Definition verwenden.
5. Verwenden Sie die Anweisung GRANT, um alle Zugriffsrechte, die zuvor für die Sicht erteilt waren, erneut zu erteilen. (Beachten Sie, dass alle für eine funktionsunfähig gewordene Sicht erteilten Zugriffsrechte widerrufen werden.)

Wenn Sie eine funktionsunfähige Sicht nicht wiederherstellen möchten, können Sie sie explizit mit der Anweisung DROP VIEW löschen, oder Sie können eine neue Sicht mit demselben Namen, aber einer anderen Definition erstellen.

Eine funktionsunfähige Sicht hat nur noch Einträge in den Katalogsichten SYSCAT.TABLES und SYSCAT.VIEWS. Alle Einträge in den Katalogsichten SYSCAT.TABDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS und SYSCAT.COLAUTH werden entfernt.

Löschen von Sichten

Verwenden Sie zum Löschen von Sichten die Anweisung DROP VIEW. Alle Sichten, die von der gelöschten Sicht abhängig sind, werden als funktionsunfähig markiert.

Geben Sie in der Befehlszeile Folgendes ein, um eine Sicht zu löschen:

```
DROP VIEW <sichtname>
```

Das folgende Beispiel zeigt, wie die Sicht EMP_VIEW gelöscht wird:

```
DROP VIEW EMP_VIEW
```

Ebenso wie bei Tabellenhierarchien kann eine gesamte Sichtenhierarchie in einer einzigen Anweisung gelöscht werden, in der die Stammsicht der Hierarchie angegeben wird, wie im folgenden Beispiel gezeigt:

```
DROP VIEW HIERARCHY VPerson
```

Teil 4. Referenz

Kapitel 17. Namenskonventionen

Namenskonventionen

Für die Benennung aller Objekte, Benutzer und Gruppen gelten Regeln. Einige dieser Regeln sind für die Plattform spezifisch, auf der Sie arbeiten.

Es gibt zum Beispiel eine Regel für die Verwendung von Groß- und Kleinbuchstaben in einem Namen.

- Auf UNIX-Plattformen müssen Namen in Kleinbuchstaben angegeben werden.
- Auf Windows-Plattformen ist die Groß-, Kleinschreibung und gemischte Groß- und Kleinschreibung zulässig.

Wenn nichts anderes angegeben ist, dürfen alle Namen die folgenden Zeichen enthalten:

- A bis Z. In den meisten Namen werden die Zeichen A bis Z von Kleinbuchstaben in Großbuchstaben umgesetzt.
- 0 bis 9.
- ! % ( ) { } . – ^ ~ _ (Unterstreichungszeichen) @, #, $ und Leerzeichen.
- \ (Backslash).

Namen dürfen nicht mit einer Ziffer oder dem Unterstreichungszeichen beginnen.

Für SQL reservierte Wörter dürfen nicht als Namen von Tabellen, Sichten, Spalten, Indizes oder Berechtigungs-IDs verwendet werden.

In Abhängigkeit vom verwendeten Betriebssystem und dem Ort, an dem Sie mit der DB2-Datenbank arbeiten, sind möglicherweise weitere Sonderzeichen verfügbar, die ebenfalls verwendet werden können. Diese Zeichen können funktionieren, jedoch gibt es keine Garantie, dass sie funktionieren. Daher wird empfohlen, beim Benennen von Objekten in der Datenbank keine anderen als die oben aufgelisteten Sonderzeichen zu verwenden.

Für Benutzer- und Gruppennamen müssen die Regeln beachtet werden, die für bestimmte Betriebssysteme durch zugehörige Systeme erzwungen werden. Zum Beispiel sind auf Linux- und UNIX-Plattformen die folgenden Zeichen für Benutzernamen und Primärgruppennamen zulässig: a bis z in Kleinbuchstaben, 0 bis 9 und _ (Unterstreichungszeichen) für Namen, die nicht mit einer Ziffer zwischen 0 und 9 beginnen.

Längen dürfen die in SQL- und XML-Begrenzungen aufgeführten Längen nicht überschreiten.

Darüber hinaus müssen Sie auch die Namenskonventionen der einzelnen Objekte, die Namenskonventionen in einer Umgebung mit Unterstützung von Landessprachen (NLS) sowie die Namenskonventionen in einer Unicode-Umgebung beachten.

Einschränkungen für die Berechtigungs-ID (AUTHID): Ab Version 9.5 des DB2-Datenbanksystems können Sie eine 128-Byte-Berechtigungs-ID verwenden. Wenn die Berechtigungs-ID jedoch als Benutzer-ID oder Gruppenname des Betriebssystems interpretiert wird, gelten die Einschränkungen des Betriebssystems (z. B. eine Begrenzung auf 8 oder 30 Zeichen für Benutzer-IDs und Gruppennamen). Daher

können Sie zwar eine 128-Byte-Berechtigungs-ID erteilen, jedoch ist es nicht möglich, als Benutzer, der diese Berechtigungs-ID hat, eine Verbindung herzustellen. Wenn Sie ein eigenes Sicherheits-Plug-in schreiben, sind Sie in der Lage, die erweiterten Größen für die Berechtigungs-ID voll auszunutzen. Zum Beispiel können Sie an Ihr Sicherheits-Plug-in eine 30-Byte-Benutzer-ID übergeben, und das Plug-in kann eine 128-Byte-Berechtigungs-ID während der Authentifizierung zurückgeben, mit der Sie eine Verbindung herstellen können.

Namenskonventionen für DB2-Objekte

Für alle Objekte müssen die allgemeinen Namenskonventionen beachtet werden. Darüber hinaus gelten für bestimmte Objekte zusätzliche Einschränkungen, die den folgenden Tabellen zu entnehmen sind.

Tabelle 52. Namenskonventionen für Datenbanken, Aliasnamen von Datenbanken und Instanzen

Objekte	Richtlinien
• Datenbanken • Aliasnamen von Datenbanken • Instanzen	• Datenbanknamen müssen an der Position, an der sie katalogisiert werden, eindeutig sein. Bei Linux- und UNIX-Implementierungen ist diese Position ein Verzeichnispfad. Bei Windows-Implementierungen ist diese Position eine logische Platte. • Aliasnamen der Datenbanken müssen innerhalb des Systemdatenbankverzeichnisses eindeutig sein. Beim Erstellen einer neuen Datenbank wird der Aliasname der Datenbank standardmäßig so definiert, dass er mit dem Datenbanknamen identisch ist. Daher können Sie keine Datenbank mit einem Namen erstellen, der bereits als Aliasname einer Datenbank verwendet wird, auch wenn noch keine Datenbank mit diesem Namen vorhanden ist. • Datenbanknamen, Aliasnamen und Instanznamen dürfen die Länge von 8 Byte nicht überschreiten. • Unter Windows dürfen Instanzen keine Namen haben, die als Namen von Services (Diensten) verwendet werden. **Anmerkung:** Um mögliche Probleme zu vermeiden, empfiehlt es sich, die Sonderzeichen @, # und $ nicht in Datenbanknamen zu verwenden, wenn die Datenbank in einer Übertragungsumgebung verwendet werden soll. Darüber hinaus sollten Sie diese Zeichen sowie Umlaute nicht benutzen, wenn Sie die Datenbank in einer anderen Sprache verwenden wollen, da diese Zeichen nicht auf allen Tastaturen in gleicher Weise verfügbar sind.

Tabelle 53. Namenskonventionen für Datenbankobjekte

Objekte	Richtlinien
• Aliasnamen • Prüfrichtlinien • Pufferpools • Spalten • Ereignismonitore • Indizes • Methoden • Knotengruppen • Pakete • Paketversionen • Rollen • Schemata • Gespeicherte Prozeduren • Tabellen • Tabellenbereiche • Trigger • Gesicherter Kontext • Benutzerdefinierte Funktionen • Benutzerdefinierte Typen • Sichten	Die Längen dieser Objekte dürfen die in SQL- und XML-Begrenzungen aufgeführten Längen nicht überschreiten. Objektnamen dürfen auch die folgenden Elemente enthalten: • Gültige Zeichen mit Akzent und Umlaute (wie beispielsweise ö) • Mehrbytezeichen mit Ausnahme von Mehrbyteleerzeichen (in Mehrbyteumgebungen) Paketnamen und Paketversionen dürfen auch Punkte (.), Bindestriche (-) und Doppelpunkte (:) enthalten.

Tabelle 54. Namenskonventionen für Objekte in föderierten Datenbanken

Objekte	Richtlinien
• Funktionszuordnungen • Indexspezifikationen • Kurznamen • Server • Typenzuordnungen • Benutzerzuordnungen • Wrapper	Die Längen dieser Objekte dürfen die in SQL- und XML-Begrenzungen aufgeführten Längen nicht überschreiten. Namen für Objekte in föderierten Datenbanken dürfen auch die folgenden Elemente enthalten: • Gültige Zeichen mit Akzent und Umlaute (wie beispielsweise ö) • Mehrbytezeichen mit Ausnahme von Mehrbyteleerzeichen (in Mehrbyteumgebungen)

Namen von begrenzten Bezeichnern und Objekten

Schlüsselwörter dürfen verwendet werden. Wird ein Schlüsselwort in einem Kontext verwendet, in dem es auch als SQL-Schlüsselwort interpretiert werden kann, muss es als begrenzter Bezeichner angegeben werden.

Mithilfe der begrenzten Bezeichner ist es möglich, ein Objekt zu erstellen, dessen Name gegen diese Namenskonventionen verstößt. Bei der späteren Verwendung

eines solchen Objekts können jedoch Fehler auftreten. Wenn Sie zum Beispiel eine Spalte mit einem Namen erstellt haben, in dem ein Pluszeichen (+) oder ein Minuszeichen (-) vorkommt, und Sie diese Spalte später in einem Index verwenden, treten Probleme auf, wenn Sie versuchen, die Tabelle zu reorganisieren.

Weitere Informationen zu Schemanamen

- Benutzerdefinierte Typen (UDTs) dürfen keine Schemanamen verwenden, deren Längen die in SQL- und XML-Begrenzungen aufgeführten Längen überschreiten.
- Die folgenden Schemanamen sind reservierte Wörter und dürfen nicht verwendet werden: SYSCAT, SYSFUN, SYSIBM, SYSSTAT.
- Um mögliche Probleme bei einer zukünftigen Migration auszuschließen, sollten Sie keine Schemanamen verwenden, die mit der Zeichenfolge SYS beginnen. Der Datenbankmanager lässt die Erstellung von Triggern, benutzerdefinierten Typen oder benutzerdefinierten Funktionen, die einen mit SYS beginnenden Schemanamen verwenden, nicht zu.
- Es wird empfohlen, das Wort SESSION nicht als Schemanamen zu verwenden. Deklarierte temporäre Tabellen müssen durch SESSION qualifiziert werden. Daher kann es vorkommen, dass eine Anwendung eine temporäre Tabelle mit einem Namen deklariert, der mit dem einer persistenten Tabelle identisch ist. In diesem Fall kann die Anwendungslogik zu komplex werden. Vermeiden Sie die Verwendung des Schemas SESSION, außer wenn Sie mit deklarierten temporären Tabellen arbeiten.

Namen von begrenzten Bezeichnern und Objekten

Schlüsselwörter dürfen verwendet werden. Wird ein Schlüsselwort in einem Kontext verwendet, in dem es auch als SQL-Schlüsselwort interpretiert werden kann, muss es als begrenzter Bezeichner angegeben werden.

Mithilfe der begrenzten Bezeichner ist es möglich, ein Objekt zu erstellen, dessen Name gegen diese Namenskonventionen verstößt. Bei der späteren Verwendung eines solchen Objekts können jedoch Fehler auftreten. Wenn Sie zum Beispiel eine Spalte mit einem Namen erstellt haben, in dem ein Pluszeichen (+) oder ein Minuszeichen (-) vorkommt, und Sie diese Spalte später in einem Index verwenden, treten Probleme auf, wenn Sie versuchen, die Tabelle zu reorganisieren.

Namenskonventionen für Benutzer, Benutzer-IDs und Gruppen

Für Benutzernamen, Benutzer-IDs und Gruppennamen müssen die folgenden geltenden Richtlinien beachtet werden.

Tabelle 55. Namenskonventionen für Benutzer, Benutzer-IDs und Gruppen

Objekte	Richtlinien
• Gruppennamen • Benutzernamen • Benutzer-IDs	• Die Längen von Gruppennamen dürfen die in SQL- und XML-Begrenzungen aufgeführte Gruppennamenlänge nicht überschreiten. • Benutzer-IDs auf Linux- und UNIX-Betriebssystemen dürfen bis zu 8 Zeichen enthalten. • Benutzernamen unter Windows dürfen bis zu 30 Zeichen enthalten. • Wenn nicht der Authentifizierungstyp CLIENT verwendet wird, werden Clients unter einem anderen als einem 32-Bit-Windows-Betriebssystem, die auf Windows zugreifen und Benutzernamen verwenden, die länger als die in SQL- und XML-Begrenzungen aufgeführte Benutzernamenlänge sind, unterstützt, wenn der Benutzername und das Kennwort explizit angegeben werden. • Für Namen und IDs gilt Folgendes: – USERS, ADMINS, GUESTS, PUBLIC, LOCAL oder für SQL reservierte Wörter sind nicht zulässig. – Sie dürfen nicht mit IBM, SQL oder SYS beginnen.

Anmerkung:

1. Für bestimmte Betriebssysteme muss bei Benutzer-IDs und Kennwörtern die Groß-/Kleinschreibung beachtet werden. Informationen hierzu enthält die Dokumentation des Betriebssystems.
2. Die von einer erfolgreichen CONNECT- oder ATTACH-Operation zurückgegebene Berechtigungs-ID wird auf die Länge für Berechtigungsnamen abgeschnitten, die in SQL- und XML-Begrenzungen aufgeführt ist. An die Berechtigungs-ID werden drei Punkte (...) angefügt und die SQLWARN-Felder enthalten Warnungen, die auf die Abtrennung der übrigen Zeichen hinweisen.
3. Folgende Leerzeichen werden aus Benutzer-IDs und Kennwörtern entfernt.

Benennungsregeln in einer NLS-Umgebung

Der Standardzeichensatz, der für Datenbanknamen verwendet werden kann, besteht aus den großen und kleinen Einzelbytebuchstaben des lateinischen Alphabets (A...Z, a...z), den arabischen Ziffern (0...9) und dem Unterstreichungszeichen (_).

Diese Liste wird noch um drei Sonderzeichen (#, @ und $) erweitert, um Kompatibilität mit den Hostdatenbankprodukten zu gewährleisten. Die Sonderzeichen #, @ und $ sind in einer NLS-Umgebung mit Vorsicht zu verwenden, da sie nicht im unveränderlichen Zeichensatz für NLS-Hosts (EBCDIC) enthalten sind. Je nach verwendeter Codepage können auch Zeichen aus dem erweiterten Zeichensatz verwendet werden. Wenn Sie die Datenbank in einer Umgebung mit mehreren Codepages verwenden, müssen Sie darauf achten, dass alle Codepages die Elemente aus dem erweiterten Zeichensatz unterstützen, die Sie verwenden möchten.

Bei der Benennung von Datenbankobjekten (wie Tabellen und Sichten), Programmkennsätzen, Hostvariablen und Cursorn können auch Elemente aus dem erweiterten Zeichensatz (z. B. Buchstaben mit diakritischen Zeichen) verwendet werden. Welche Zeichen im Einzelnen verfügbar sind, hängt von der verwendeten Codepage ab.

Definition des erweiterten Zeichensatzes für DBCS-Bezeichner: In DBCS-Umgebungen umfasst der erweiterte Zeichensatz alle Zeichen des Standardzeichensatzes plus die folgenden Elemente:

- Alle Doppelbytezeichen in jeder DBCS-Codepage, mit Ausnahme des Doppelbyteleerzeichens, sind gültige Buchstaben.
- Das Doppelbyteleerzeichen ist ein Sonderzeichen.
- Die in jeder Mischcodepage verfügbaren Einzelbytezeichen werden verschiedenen Kategorien zugeordnet:

Kategorie	Gültige Codepunkte in jeder Mischcodepage
Ziffern	x30-39
Buchstaben	x23-24, x40-5A, x61-7A, xA6-DF (A6-DF nur für Codepages 932 und 942)
Sonderzeichen	Alle anderen gültigen Codepunkte für Einzelbytezeichen

Benennungsregeln in einer Unicode-Umgebung

In einer Unicode-Datenbank weisen alle Bezeichner das UTF-8-Mehrbyteformat auf. Daher ist es möglich, in Bezeichnern beliebige UCS-2-Zeichen zu verwenden, sofern die Verwendung eines Zeichens des erweiterten Zeichensatzes (wie beispielsweise eines Zeichens mit Akzent oder eines Mehrbytezeichens) vom DB2-Datenbanksystem zugelassen wird.

Clients können alle Zeichen eingeben, die von ihrer Umgebung unterstützt werden. Alle Zeichen in den Bezeichnern werden vom Datenbankmanager in UTF-8 umgesetzt. Zwei Punkte müssen beim Angeben von Zeichen der Landessprache in Bezeichnern für eine Unicode-Datenbank beachtet werden:

- Jedes Nicht-ASCII-Zeichen erfordert zwei bis vier Byte. Ein Bezeichner mit n Byte kann daher je nach Verhältnis von ASCII- zu anderen Zeichen nur zwischen $n/4$ und n Zeichen enthalten. Wenn Sie nur ein oder zwei Nicht-ASCII-Zeichen (z. B. Zeichen mit Akzent) haben, liegt die Grenze näher an n Zeichen, während für einen Bezeichner, der nur aus Nicht-ASCII-Zeichen besteht (z. B. in Japanisch), maximal $n/4$ bis $n/3$ Zeichen verwendet werden können.
- Wenn Bezeichner von unterschiedlichen Clientumgebungen eingegeben werden sollen, sollten sie mit der gemeinsamen Untermenge von Zeichen definiert werden, die auf diesen Clients verfügbar sind. Wenn z. B. von Lateinisch-1-, arabischen oder japanischen Umgebungen auf eine Unicode-Datenbank zugegriffen werden soll, sollten alle Bezeichner auf ASCII beschränkt werden.

Kapitel 18. SQL- und XML-Begrenzungen

In den folgenden Tabellen werden verschiedene SQL- und XML-Begrenzungen beschrieben. Die Orientierung am restriktivsten Fall erleichtert den Entwurf von Anwendungsprogrammen, die sich problemlos portieren lassen.

In Tabelle 56 werden Begrenzungen in Byte aufgeführt. Diese Begrenzungen werden bei der Erstellung von Kennungen (oder Bezeichnern) nach einer Konvertierung aus der Anwendungscodepage in die Datenbankcodepage geprüft. Die Begrenzungen werden darüber hinaus auch beim Abrufen von Kennungen aus der Datenbank nach der Konvertierung aus der Datenbankcodepage in die Anwendungscode geprüft. Wenn während eines dieser Prozesse die Längenbegrenzung einer Kennung überschritten wird, wird die Kennung abgeschnitten oder es wird ein Fehler zurückgegeben.

Zeichenbegrenzungen sind je nach Codepage der Datenbank und der Codepage der Anwendung unterschiedlich. Da die Länge eines UTF-8-Zeichens zum Beispiel zwischen 1 und 4 Byte betragen kann, kann die Zeichenbegrenzung für eine Kennung in einer Unicode-Tabelle, deren Begrenzung bei 128 Byte liegt, je nach verwendeten Zeichen zwischen 32 und 128 Zeichen betragen. Wenn versucht wird, eine Kennung zu erstellen, die nach der Konvertierung in die Datenbankcodepage länger als die Begrenzung für die betreffende Tabelle ist, wird ein Fehler zurückgegeben.

Anwendungen, die Kennungen speichern, müssen in der Lage sein, die potenzielle Längenzunahme dieser Kennungen nach einer Codepagekonvertierung aufzufangen. Wenn Kennungen aus dem Katalog abgerufen werden, werden sie in die Codepage der Anwendung konvertiert. Die Konvertierung aus der Datenbankcodepage in die Anwendungscodepage kann dazu führen, dass eine Kennung länger wird und die Bytebegrenzung für die Tabelle überschreitet. Wenn eine von der Anwendung deklarierte Hostvariable nach einer Codepagekonvertierung nicht die gesamte Kennung speichern kann, wird die Kennung abgeschnitten. Wenn dies nicht akzeptabel ist, kann der Wert für die Größe der Hostvariablen erhöht werden, sodass die Hostvariable die gesamte Kennung aufnehmen kann.

Die gleichen Regeln gelten auch für DB2-Dienstprogramme, die Daten abrufen und in eine benutzerdefinierte Codepage konvertieren. Wenn ein DB2-Dienstprogramm, wie zum Beispiel EXPORT, Daten abruft und die Konvertierung in eine benutzerdefinierte Codepage (durch den EXPORT-Modifikator CODEPAGE oder durch die Registrierdatenbankvariable **DB2CODEPAGE**) erzwingt und die Daten die Begrenzung, die in der nachfolgenden Tabelle dokumentiert ist, infolge der Codepagekonvertierung überschreiten, wird möglicherweise ein Fehler zurückgegeben oder die Daten werden abgeschnitten.

Tabelle 56. Längenbegrenzungen von Kennungen

Beschreibung	Maximum in Byte
Aliasname	128
Attributname	128
Name einer Prüfrichtlinie	128
Berechtigungsname (nur Einzelbytezeichen sind zulässig)	128
Pufferpoolname	18

Tabelle 56. Längenbegrenzungen von Kennungen (Forts.)

Beschreibung	Maximum in Byte
Spaltenname[2]	128
Integritätsbedingungsname	128
Korrelationsname	128
Cursorname	128
Datenpartitionsname	128
Name einer Datenquellenspalte	255
Indexname einer Datenquelle	128
Datenquellenname	128
Tabellenname einer Datenquelle (*ferner-tabellenname*)	128
Name einer Datenbankpartitionsgruppe	128
Name einer Datenbankpartition	128
Name eines Ereignismonitors	128
Name eines externen Programms	128
Funktionszuordnungsname	128
Gruppenname	128
Hostkennung[1]	255
Kennung für Datenquellenbenutzer (*ferner-berechtigungsname*)	128
Kennung in SQL-Prozedur (Bedingungsname, For-Schleifen-Kennung, Kennsatz, Zeigerwert für Ergebnismenge, Anweisungsname und Variablenname)	128
Indexname	128
Indexerweiterungsname	18
Name einer Indexspezifikation	128
Kennsatzname	128
Uniform-Resource-Identifier (URI) eines Namensbereichs	1000
Kurzname	128
Paketname	128
Paketversions-ID	64
Parametername	128
Kennwort für Datenquellenzugriff	32
Prozedurname	128
Rollenname	128
Sicherungspunktname	128
Schemaname[2]	128
Name einer Sicherheitskennsatzkomponente	128
Name eines Sicherheitskennsatzes	128
Name einer Sicherheitsrichtlinie	128
Sequenzname	128
Servername (Aliasname einer Datenbank)	8
Bestimmter Name	128

Tabelle 56. Längenbegrenzungen von Kennungen (Forts.)

Beschreibung	Maximum in Byte
SQL-Bedingungsname	128
SQL-Variablenname	128
Anweisungsname	128
Tabellenname	128
Tabellenbereichsname	18
Umsetzungsgruppenname	18
Triggername	128
Name eines gesicherten Kontexts	128
Name einer Zuordnung von Inhaltstypen	18
Benutzerdefinierter Funktionsname	128
Benutzerdefinierter Methodenname	128
Name eines benutzerdefinierten Datentyps[2]	128
Sichtname	128
Wrappername	128
XML-Elementname, Attributname oder Präfixname	1000
Uniform-Resource-Identifier (URI) für Position des XML-Schemas	1000

Anmerkung:

1. Für einzelne Compiler für Hostprogrammiersprachen gilt möglicherweise eine noch restriktivere Begrenzung für Variablennamen.
2. Die Struktur des SQL-Deskriptorbereichs (SQLDA) ist auf das Speichern von 30 Byte langen Spaltennamen, 18 Byte langen Namen benutzerdefinierter Datentypen und 8 Byte langen Schemanamen für benutzerdefinierte Typen beschränkt. Da der SQL-Deskriptorbereich in der Anweisung DESCRIBE verwendet wird, müssen Anwendungen mit eingebettetem SQL, die die Anweisung DESCRIBE zum Abrufen von Informationen zu Spaltennamen oder Namen von benutzerdefinierten Datentypen verwenden, diese Begrenzungen einhalten.

Tabelle 57. Numerische Begrenzungen

Beschreibung	Begrenzung
Niedrigster SMALLINT-Wert	-32.768
Höchster SMALLINT-Wert	+32.767
Niedrigster INTEGER-Wert	-2.147.483.648
Höchster INTEGER-Wert	+2.147.483.647
Niedrigster BIGINT-Wert	-9.223.372.036.854.775.808
Höchster BIGINT-Wert	+9.223.372.036.854.775.807
Größte Dezimalgenauigkeit	31
Maximaler Exponent (E_{max}) für REAL-Werte	38
Niedrigster REAL-Wert	-3,402E+38
Höchster REAL-Wert	+3,402E+38
Minimaler Exponent (E_{min}) für REAL-Werte	-37

Tabelle 57. Numerische Begrenzungen (Forts.)

Beschreibung	Begrenzung
Niedrigster positiver REAL-Wert	+1,175E-37
Höchster negativer REAL-Wert	-1,175E-37
Maximaler Exponent (E_{max}) für DOUBLE-Werte	308
Niedrigster DOUBLE-Wert	-1,79769E+308
Höchster DOUBLE-Wert	+1,79769E+308
Minimaler Exponent (E_{min}) für DOUBLE-Werte	-307
Niedrigster positiver DOUBLE-Wert	+2,225E-307
Höchster negativer DOUBLE-Wert	-2,225E-307
Maximaler Exponent (E_{max}) für DECFLOAT(16)-Werte	384
Niedrigster DECFLOAT(16)-Wert[1]	-9,999999999999999E+384
Höchster DECFLOAT(16)-Wert	9,999999999999999E+384
Minimaler Exponent (E_{min}) für DECFLOAT(16)-Werte	-383
Niedrigster positiver DECFLOAT(16)-Wert	1,000000000000000E-383
Höchster negativer DECFLOAT(16)-Wert	-1,000000000000000E-383
Maximaler Exponent (E_{max}) für DECFLOAT(34)-Werte	6144
Niedrigster DECFLOAT(34)-Wert[1]	-9,999999999999999999999999999999999E+6144
Höchster DECFLOAT(34)-Wert	9,999999999999999999999999999999999E+6144
Minimaler Exponent (E_{min}) für DECFLOAT(34)-Werte	-6143
Niedrigster positiver DECFLOAT(34)-Wert	1,000000000000000000000000000000000E-6143
Höchster negativer DECFLOAT(34)-Wert	-1,000000000000000000000000000000000E-6143

Tabelle 57. Numerische Begrenzungen (Forts.)

Beschreibung	Begrenzung
Anmerkung:	
1. Dies sind die Begrenzungen für normale Gleitkommadezimalzahlen. Gültige Gleitkommadezimalwerte sind außerdem die Sonderwerte NAN, -NAN, SNAN, -SNAN, INFINITY und -INFINITY. Darüber hinaus gehören zu den gültigen Werten subnormale Zahlen. Subnormale Zahlen sind Zahlen ungleich null, deren angepasste Exponenten kleiner als E_{min} sind. Bei einer subnormalen Zahl ist der Minimalwert des Exponenten gleich E_{min} - (*genauigkeit*-1), als E_{tiny} bezeichnet, wobei *genauigkeit* die Genauigkeit ist, mit der gearbeitet wird (16 oder 34). Das heißt, subnormale Zahlen erweitern den Bereich der Zahlen nahe null um 15 oder 33 Größenordnungen für DECFLOAT(16) bzw. DECFLOAT(34). Subnormale Zahlen unterscheiden sich von normalen Zahlen, weil die maximale Anzahl von Stellen kleiner ist als die Genauigkeit, mit der gearbeitet wird (16 bzw. 34). Gleitkommadezimalzahlen können subnormale Zahlen nicht mit derselben Präzision wie normale Zahlen darstellen. Die niedrigste positive subnormale Zahl für DECFLOAT(34) ist $1x10^{-6176}$, die nur eine Stelle hat, während für DECFLOAT(34) die niedrigste positive normale Zahl $1{,}000000000000000000000000000000000x10^{-6143}$ ist, die 34 Stellen enthält. Die niedrigste positive subnormale Zahl für DECFLOAT(16) ist $1x10^{-398}$.	

Tabelle 58. Zeichenfolgebegrenzungen

Beschreibung	Begrenzung
Maximale Länge von CHAR (in Byte)	254
Maximale Länge von VARCHAR (in Byte)	32 672
Maximale Länge von LONG VARCHAR (in Byte)[1]	32 700
Maximale Länge von CLOB (in Byte)	2 147 483 647
Maximale Länge von serialisierter XML (in Byte)	2 147 483 647
Maximale Länge von GRAPHIC (in Doppelbytezeichen)	127
Maximale Länge von VARGRAPHIC (in Doppelbytezeichen)	16 336
Maximale Länge von LONG VARGRAPHIC (in Doppelbytezeichen)[1]	16 350
Maximale Länge von DBCLOB (in Doppelbytezeichen)	1 073 741 823
Maximale Länge von BLOB (in Byte)	2 147 483 647
Maximale Länge der Zeichenkonstanten	32 672
Maximale Länge der Grafikkonstanten	16 336
Maximale Länge der verknüpften Zeichenfolge	2 147 483 647
Maximale Länge der verknüpften Grafikzeichenfolge	1 073 741 823
Maximale Länge der verknüpften Binärzeichenfolge	2 147 483 647
Maximale Anzahl von Hexadezimalkonstantenziffern	32 672
Größte Instanz eines Spaltenobjekts mit strukturiertem Typ während der Laufzeit (in Gigabyte)	1
Maximale Größe eines Katalogkommentars (in Byte)	254
Anmerkung:	
1. Die Datentypen LONG VARCHAR und LONG VARGRAPHIC sind veraltet, werden nicht empfohlen und in einem zukünftigen Release möglicherweise entfernt.	

Tabelle 59. XML-Begrenzungen

Beschreibung	Begrenzung
Maximale Verschachtelungstiefe eines XML-Dokuments (in Stufen)	125
Maximale Größe eines XML-Schemadokuments (in Byte)	31 457 280

Tabelle 60. Begrenzungen für Datums- und Uhrzeitangaben

Beschreibung	Begrenzung
Niedrigster DATE-Wert	0001-01-01
Höchster DATE-Wert	9999-12-31
Niedrigster TIME-Wert	00:00:00
Höchster TIME-Wert	24:00:00
Niedrigster TIMESTAMP-Wert	0001-01-01-00.00.00.000000
Höchster TIMESTAMP-Wert	9999-12-31-24.00.00.000000

Tabelle 61. Begrenzungen des Datenbankmanagers

Beschreibung	Begrenzung
Anwendungen	
Maximale Anzahl von Deklarationen von Hostvariablen in einem vorkompilierten Programm[3]	Speicher
Maximale Länge eines Hostvariablenwerts (in Byte)	2 147 483 647
Maximale Anzahl deklarierter Cursor in einem Programm	Speicher
Maximale Anzahl von Zeilen, die in einer Arbeitseinheit geändert wurden	Speicher
Maximale Anzahl von Cursorn, die zu einem bestimmten Zeitpunkt geöffnet wurden	Speicher
Maximale Anzahl von Verbindungen pro Prozess in einem DB2-Client	512
Maximale Anzahl von gleichzeitig geöffneten LOB-Querverweisen in einer Transaktion	4 194 304
Maximale Größe eines SQL-Deskriptorbereichs (in Byte)	Speicher
Maximale Anzahl vorbereiteter Anweisungen	Speicher
Pufferpools	
Maximale Anzahl NPAGES in einem Pufferpool für 32-Bit-Releases	1 048 576
Maximale Anzahl NPAGES in einem Pufferpool für 64-Bit-Releases	2 147 483 647
Maximale Gesamtgröße aller Pufferpoolslots (4 KB)	2 147 483 646
Gemeinsamer Zugriff	
Maximale Anzahl gleichzeitig angemeldeter Benutzer eines Servers[4]	64 000
Maximale Anzahl gleichzeitig angemeldeter Benutzer pro Instanz	64 000
Maximale Anzahl gleichzeitig ausgeführter Anwendungen pro Datenbank	60 000

Tabelle 61. Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Begrenzung
Maximale Anzahl von Datenbanken pro Instanz, die gleichzeitig benutzt werden können	256
Integritätsbedingungen	
Maximale Anzahl von Integritätsbedingungen für eine Tabelle	Speicher
Maximale Anzahl von Spalten in einer UNIQUE-Integritätsbedingung (unterstützt über einen UNIQUE-Index)	64
Maximale kombinierte Länge der Spalten in einer UNIQUE-Integritätsbedingung (unterstützt über einen UNIQUE-Index - in Byte)[9]	8192
Maximale Anzahl von verweisenden Spalten in einem Fremdschlüssel	64
Maximale kombinierte Länge von verweisenden Spalten in einem Fremdschlüssel (in Byte)[9]	8192
Maximale Länge einer Prüfbedingungsangabe (in Byte)	65 535
Datenbanken	
Maximale Anzahl von Datenbankpartitionen	999
Indizes	
Maximale Anzahl von Indizes für eine Tabelle	32 767 oder Speicher
Maximale Anzahl von Spalten in einem Indexschlüssel	64
Maximale Länge eines Indexschlüssels einschließlich des gesamten Systemaufwands[7 9]	*indexseitengröße*/4
Maximale Länge eines variablen Indexschlüsselteils (in Byte)[8]	1022 oder Speicher
Maximale Größe eines Indexes pro Datenbankpartition in einem SMS-Tabellenbereich (in Gigabyte)[7]	16 384
Maximale Größe eines Indexes pro Datenbankpartition in einem regulären DMS-Tabellenbereich (in Gigabyte)[7]	512
Maximale Größe eines Indexes pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Gigabyte)[7]	16 384
Maximale Größe für einen Index für XML-Daten pro Datenbankpartition (in Terabyte)	2
Maximale Länge eines variablen Indexschlüsselteils für einen Index für XML-Daten (in Byte)[7]	*seitengröße*/4 - 207
Protokollsätze	
Maximale Protokollfolgenummer (LSN)	281 474 976 710 655
Überwachung	
Maximale Anzahl gleichzeitig aktiver Ereignismonitore	32
Routinen	
Maximale Anzahl von Parametern in einer Prozedur	32 767
Maximale Anzahl von Parametern in einer benutzerdefinierten Funktion	90
Maximale Anzahl von Verschachtelungsebenen für Routinen	64
Maximale Anzahl von Schemata im SQL-Pfad	64

Tabelle 61. Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Begrenzung
Maximale Länge des SQL-Pfads (in Byte)	2048
Sicherheit	
Maximale Anzahl von Elementen in einer Sicherheitskennsatzkomponente des Typs 'set' oder 'tree'	64
Maximale Anzahl von Elementen in einer Sicherheitskennsatzkomponente des Typs 'array'	65 535
Maximale Anzahl von Sicherheitskennsatzkomponenten in einer Sicherheitsrichtlinie	16
SQL	
Maximale Gesamtlänge einer SQL-Anweisung (in Byte)	2 097 152
Maximale Anzahl von Tabellen, auf die in einer SQL-Anweisung oder einer Sicht verwiesen wird	Speicher
Maximale Anzahl von Verweisen auf Hostvariablen in einer SQL-Anweisung	32 767
Maximale Anzahl von Konstanten in einer Anweisung	Speicher
Maximale Anzahl von Elementen in einer SELECT-Liste[7]	1012
Maximale Anzahl von Vergleichselementen in einer Klausel WHERE oder HAVING	Speicher
Maximale Anzahl von Spalten in einer Klausel GROUP BY[7]	1012
Maximale Gesamtlänge der Spalten in einer Klausel GROUP BY (in Byte)[7]	32 677
Maximale Anzahl von Spalten in einer Klausel ORDER BY[7]	1012
Maximale Gesamtlänge der Spalten in einer Klausel ORDER BY (in Byte)[7]	32 677
Höchste Ebene der Unterabfragenverschachtelung	Speicher
Maximale Anzahl von Unterabfragen in einer einzelnen Anweisung	Speicher
Maximale Anzahl von Werten in einer Einfügeoperation[7]	1012
Maximale Anzahl von SET-Klauseln in einer einzigen Aktualisierungsoperation[7]	1012
Tabellen und Sichten	
Maximale Anzahl von Spalten in einer Tabelle[7]	1012
Maximale Anzahl von Spalten in einer Sicht[1]	5000
Maximale Anzahl von Spalten in einer Datenquellentabelle oder einer Sicht, auf die mit einem Kurznamen verwiesen wird	5000
Maximale Anzahl von Spalten in einem Verteilungsschlüssel[5]	500
Maximale Länge einer Zeile einschließlich des gesamten Systemaufwands[2 7]	32 677
Maximale Anzahl von Zeilen in einer nicht partitionierten Tabelle (pro Datenbankpartition)	128×10^{10}
Maximale Anzahl von Zeilen in einer Datenpartition (pro Datenbankpartition)	128×10^{10}

Tabelle 61. Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Begrenzung
Maximale Größe einer Tabelle pro Datenbankpartition in einem regulären Tabellenbereich (in Gigabyte)[3 7]	512
Maximale Größe einer Tabelle pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Gigabyte)[7]	16 384
Maximale Anzahl von Datenpartitionen für eine einzelne Tabelle	32 767
Maximale Anzahl von Tabellenpartitionierungsspalten	16
Tabellenbereiche	
Maximale Größe eines LOB-Objekts (in Terabyte)	4
Maximale Größe eines Langfeldobjekts (LF-Objekt, in Terabyte)	2
Maximale Anzahl von Tabellenbereichen in einer Datenbank	32 768
Maximale Anzahl von Tabellen in einem SMS-Tabellenbereich	65 534
Maximale Größe eines regulären DMS-Tabellenbereichs (in Gigabyte) [3 7]	512
Maximale Größe eines großen DMS-Tabellenbereichs (in Terabyte) [3 7]	16
Maximale Größe eines temporären DMS-Tabellenbereichs (in Terabyte) [37]	16
Maximale Anzahl von Tabellenobjekten in einem DMS-Tabellenbereich[6]	51 000
Maximale Anzahl von Speicherpfaden in einer Datenbank mit dynamischem Speicher	128
Maximale Länge eines Speicherpfads, der einer Datenbank mit dynamischem Speicher zugeordnet ist (in Byte)	175
Trigger	
Maximale Laufzeitverschachtelungstiefe von hintereinandergeschalteten Triggern	16
Benutzerdefinierte Typen	
Maximale Anzahl von Attributen in einem strukturierten Typ	4082

Tabelle 61. Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Begrenzung
Anmerkung: 1. Dieser Maximalwert kann erreicht werden, wenn in der Anweisung CREATE VIEW ein Join verwendet wird. Die Auswahl aus einer solchen Sicht unterliegt den Begrenzungen der meisten Elemente in einer SELECT-Liste. 2. Die eigentlichen Daten für BLOB-, CLOB-, LONG VARCHAR-, DBCLOB- und LONG VARGRAPHIC-Spalten sind bei dieser Zählung nicht berücksichtigt. Allerdings belegen die Informationen zur Position dieser Daten innerhalb der Zeile eine gewisse Menge an Speicherplatz. 3. Die angezeigten Nummern stellen durch die Architektur bedingte Begrenzungen und Approximationen dar. Die in der Praxis geltenden Begrenzungen können darunter liegen. 4. Der tatsächliche Wert wird durch die Konfigurationsparameter **max_connections** und **max_coordagents** des Datenbankmanagers gesteuert. 5. Dies ist eine durch die Architektur bedingte Begrenzung. Als praktische Begrenzung sollte die Begrenzung benutzt werden, die für die meisten Spalten in einem Indexschlüssel gilt. 6. Tabellenobjekte umfassen Daten, Indizes, LONG VARCHAR- oder VARGRAPHIC-Spalten sowie LOB-Spalten. Tabellenobjekte, die sich im selben Tabellenbereich befinden wie die Tabellendaten, werden bei der Ermittlung der Begrenzung nicht gesondert berücksichtigt. Jedes Tabellenobjekt, das sich in einem anderen Tabellenbereich als die Tabellendaten befindet, trägt jedoch zur Erreichung der Begrenzung für die einzelnen Tabellenobjekttypen pro Tabelle in dem Tabellenbereich bei, in dem sich das Tabellenobjekt befindet. 7. Weitere Informationen zu speziellen Werten für bestimmte Seitengrößen finden Sie in Tabelle 62. 8. Eine Begrenzung erfolgt lediglich durch den längsten Indexschlüssel einschließlich des gesamten Systemaufwands (in Byte). Wenn die Anzahl der Indexschlüsselteile ansteigt, reduziert sich die maximale Länge der einzelnen Schlüsselteile. 9. Das Maximum kann abhängig von den Indexoptionen geringer sein.	

Tabelle 62. Seitengrößenspezifische Begrenzungen des Datenbankmanagers

Beschreibung	Seitengrößenbegrenzung - 4 KB	Seitengrößenbegrenzung - 8 KB	Seitengrößenbegrenzung - 16 KB	Seitengrößenbegrenzung - 32 KB
Maximale Anzahl von Spalten in einer Tabelle	500	1012	1012	1012
Maximale Länge einer Zeile einschließlich des gesamten Systemaufwands	4005	8101	16 293	32 677
Maximale Größe einer Tabelle pro Datenbankpartition in einem regulären Tabellenbereich (in Gigabyte)	64	128	256	512
Maximale Größe einer Tabelle pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Gigabyte)	2048	4096	8192	16 384

Tabelle 62. Seitengrößenspezifische Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Seitengrößen-begrenzung - 4 KB	Seitengrößen-begrenzung - 8 KB	Seitengrößen-begrenzung - 16 KB	Seitengrößen-begrenzung - 32 KB
Maximale Länge eines Indexschlüssels einschließlich des gesamten Systemaufwands (in Byte)	1024	2048	4096	8192
Maximale Größe eines Indexes pro Datenbankpartition in einem SMS-Tabellenbereich (in Gigabyte)	2048	4096	8192	16 384
Maximale Größe eines Indexes pro Datenbankpartition in einem regulären DMS-Tabellenbereich (in Gigabyte)	64	128	256	512
Maximale Größe eines Indexes pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Gigabyte)	2048	4096	8192	16 384
Maximale Größe für einen Index für XML-Daten pro Datenbankpartition (in Terabyte)	2	2	2	2
Maximale Größe eines regulären DMS-Tabellenbereichs (in Gigabyte)	64	128	256	512
Maximale Größe eines großen DMS-Tabellenbereichs (in Gigabyte)	2048	4096	8192	16 384
Maximale Größe eines temporären DMS-Tabellenbereichs (in Terabyte)	2	4	8	16
Maximale Anzahl von Elementen in einer SELECT-Liste	500	1012	1012	1012
Maximale Anzahl von Spalten in einer Klausel GROUP BY	500	1012	1012	1012
Maximale Gesamtlänge der Spalten in einer Klausel GROUP BY (in Byte)	4005	8101	16 293	32 677
Maximale Anzahl von Spalten in einer Klausel ORDER BY	500	1012	1012	1012
Maximale Gesamtlänge der Spalten in einer Klausel ORDER BY (in Byte)	4005	8101	16 293	32 677
Maximale Anzahl von Werten in einer Einfügeoperation	500	1012	1012	1012

Tabelle 62. Seitengrößenspezifische Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Seitengrößen-begrenzung - 4 KB	Seitengrößen-begrenzung - 8 KB	Seitengrößen-begrenzung - 16 KB	Seitengrößen-begrenzung - 32 KB
Maximale Anzahl von SET-Klauseln in einer einzigen Aktualisierungsoperation	500	1012	1012	1012

Kapitel 19. Registrierdatenbank- und Umgebungsvariablen

Umgebungsvariablen und die Profilregistrierdatenbank

Ihre Datenbankumgebung wird mithilfe von Umgebungsvariablen und Variablen der Profilregistrierdatenbank gesteuert.

Sie können den Konfigurationsassistenten (db2ca) zur Konfiguration von Registrierdatenbankvariablen und Konfigurationsparametern verwenden.

Vor der Einführung der DB2-Profilregistrierdatenbank war die Änderung von Umgebungsvariablen auf Windows-Workstations beispielsweise mit der Notwendigkeit verbunden, das System neu zu starten. Jetzt wird Ihre Umgebung mit wenigen Ausnahmen durch Registrierdatenbankvariablen gesteuert, die in den DB2-Profilregistrierdatenbanken gespeichert sind. Benutzer unter UNIX-Betriebssystemen mit der Berechtigung zur Systemverwaltung (SYSADM) für eine bestimmte Instanz können die Registrierungswerte für die betreffende Instanz aktualisieren. Unter Windows ist für die Aktualisierung von Profilregisterdatenbankvariablen die lokale Administratorberechtigung oder Berechtigung SYSADM entsprechend den folgenden Bedingungen erforderlich:

- Wenn die erweiterte Sicherheit aktiviert ist, müssen SYSADM-Benutzer zur Gruppe DB2ADMNS gehören.
- Wenn die erweiterte Sicherheit nicht aktiviert ist, können SYSADM-Benutzer Aktualisierungen vornehmen, sofern ihnen die entsprechenden Berechtigungen in der Windows-Registrierdatenbank erteilt wurden.

Zur Aktualisierung der Variablen der Profilregistrierdatenbank ohne Systemneustart verwenden Sie den Befehl db2set. Diese Informationen werden sofort wirksam in den Profilregistrierdatenbanken gespeichert. Änderungen wirken sich jedoch nicht auf momentan aktive DB2-Anwendungen oder -Benutzer aus. Die DB2-Registrierdatenbank wendet die aktualisierten Informationen auf diejenigen DB2-Serverinstanzen und DB2-Anwendungen an, die nach der Durchführung der Änderungen gestartet werden.

Anmerkung: Die DB2-Umgebungsvariablen **DB2INSTANCE** und **DB2NODE** werden eventuell nicht in den DB2-Profilregistrierdatenbanken gespeichert. Bei einigen Betriebssystemen muss der Befehl set zur Aktualisierung dieser Umgebungsvariablen verwendet werden. Diese Änderungen bleiben bis zum nächsten Neustart des Systems wirksam. Auf Linux- und UNIX-Plattformen wird eventuell der Befehl export anstelle des Befehls set verwendet.

Die Verwendung der Profilregistrierdatenbank ermöglicht eine zentrale Steuerung der Umgebungsvariablen. Verschiedene Ebenen der Unterstützung werden nun durch verschiedene Profile realisiert. Eine Fernverwaltung der Umgebungsvariablen steht außerdem zur Verfügung, wenn der DB2-Verwaltungsserver (DAS) verwendet wird.

Es gibt vier Profilregistrierdatenbanken:

- Die DB2-Profilregistrierdatenbank auf Instanzebene. Die meisten der DB2-Umgebungsvariablen werden in dieser Registrierdatenbank gespeichert. Die Einstellungen der Umgebungsvariablen für eine bestimmte Instanz werden in dieser

Registrierdatenbank verwaltet. Die auf dieser Ebene definierten Werte überschreiben die Einstellungen der globalen Ebene.

- Die DB2-Profilregistrierdatenbank auf globaler Ebene. Wenn eine Umgebungsvariable nicht für eine bestimmte Instanz definiert wird, wird sie in dieser Registrierdatenbank gespeichert. Diese Registrierdatenbank ist für alle Instanzen sichtbar, die zu einer bestimmten Kopie von DB2 ESE gehören. Der Installationspfad enthält genau ein Profil auf globaler Ebene.
- Die DB2-Profilregistrierdatenbank auf Instanzknotenebene. Diese Registrierdatenbankebene enthält Variableneinstellungen, die für eine Datenbankpartition in einer Umgebung mit partitionierten Datenbanken spezifisch sind. Die auf dieser Ebene definierten Werte überschreiben die entsprechenden Einstellungen auf der Instanzebene und der globalen Ebene.
- Die DB2-Instanzprofilregistrierdatenbank. Diese Registrierdatenbank enthält eine Liste aller Instanznamen, die der aktuellen Kopie zugeordnet sind. Jede Installation besitzt eine eigene Liste. Sie können die vollständige Liste aller auf dem System verfügbaren Instanzen mithilfe des Befehls db2ilist anzeigen.

DB2 konfiguriert die Betriebsumgebung, wobei die Registrierungswerte und die Umgebungsvariablen geprüft und in folgender Reihenfolge auflöst werden:

1. Umgebungsvariablen, die mit dem Befehl `set` definiert wurden (bzw. mit dem Befehl `export` auf UNIX-Plattformen)
2. Registrierdatenbankwerte, die im Profil auf Instanzknotenebene definiert wurden (mit dem Befehl db2set -i <instanzname> <knotennummer>)
3. Registrierdatenbankwerte, die im Profil auf Instanzebene definiert wurden (mit dem Befehl db2set -i)
4. Registrierdatenbankwerte, die im Profil auf globaler Ebene definiert wurden (mit dem Befehl db2set -g)

Profilregistrierdatenbank der Instanzebene

Bei der Arbeit in einer Umgebung mit partitionierten Datenbanken sind einige Unterschiede zwischen UNIX und Windows zu beachten. Diese Unterschiede werden im folgenden Beispiel aufgezeigt.

Nehmen Sie an, eine Umgebung mit einer partitionierten Datenbank besitzt drei physische Datenbankpartitionen, die als „red", „white" und „blue" identifiziert werden. Der Instanzeigner führt nun auf UNIX-Plattformen die folgenden Befehle über eine beliebige der Datenbankpartitionen aus:

```
db2set -i FOO=BAR
```

oder

```
db2set FOO=BAR        ('-i' ist impliziert)
```

In diesem Fall wird der Wert von FOO für alle Knoten der aktuellen Instanz sichtbar (d. h. für „red", „white" und „blue").

Auf UNIX-Plattformen wird die Profilregistrierdatenbank der Instanzebene in einer Textdatei im Verzeichnis `sqllib` gespeichert. In Umgebungen mit partitionierten Datenbanken befindet sich das Verzeichnis `sqllib` in dem Dateisystem, das von allen physischen Datenbankpartitionen gemeinsam genutzt wird.

Wenn der Benutzer den gleichen Befehl in der Datenbankpartition „red" auf Windows-Plattformen ausführt, wird der Wert von FOO nur in der Datenbankpartition „red" der aktuellen Instanz sichtbar. Der DB2-Datenbankmanager speichert die

Profilregistrierdatenbank der Instanzebene in der Windows-Registrierdatenbank. Eine gemeinsame Nutzung durch mehrere physische Datenbankpartitionen findet nicht statt. Sollen die Registrierdatenbankvariablen auf allen physischen Computern definiert werden, verwenden Sie den Befehl „rah" wie folgt:

```
rah db2set -i FOO=BAR
```

Durch den Befehl rah wird der Befehl db2set in den Datenbankpartitionen „red", „white" und „blue" fern ausgeführt.

Es ist möglich, mithilfe der Umgebungsvariablen **DB2REMOTEPREG** die Registrierdatenbankvariablen auf Computern, die nicht Instanzeigner sind, so zu konfigurieren, dass sie auf die Registrierdatenbankvariablen von Computern verweisen, die Instanzeigner sind. Dadurch wird eine Umgebung erstellt, in der die Registrierdatenbankvariablen auf dem Instanzeignercomputer von allen Computern in der Instanz effektiv gemeinsam genutzt werden.

Im oben gezeigten Beispiel und unter der Annahme, dass „red" der Instanzeignercomputer ist, müsste die Umgebungsvariable **DB2REMOTEPREG** auf den Computern „white" und „blue" wie folgt definiert werden, damit die Registrierdatenbankvariablen auf „red" gemeinsam genutzt würden:

```
(auf Computer red) Keine Aktion
(auf den Computern white und blue) db2set DB2REMOTEPREG=\\red
```

Die Einstellung für **DB2REMOTEPREG** darf nicht mehr geändert werden, wenn sie definiert wurde.

Die Umgebungsvariable DB2REMOTEPREG funktioniert wie folgt:

Wenn der DB2-Datenbankmanager die Registrierdatenbankvariablen unter Windows liest, liest er zuerst den Wert von **DB2REMOTEPREG**. Wenn **DB2REMOTEPREG** definiert ist, wird die Registrierdatenbank auf dem fernen Computer geöffnet, dessen Computername in der Variablen **DB2REMOTEPREG** angegeben ist. Nachfolgende Lese- und Aktualisierungsoperationen für die Registrierdatenbankvariablen werden an den angegebenen fernen Computer weitergeleitet.

Der Zugriff auf die ferne Registrierdatenbank setzt voraus, dass auf dem Zielcomputer der Service (Dienst) für die Fernregistrierung (Remote Registry Service) aktiv ist. Außerdem müssen das Benutzeranmeldekonto und alle DB2-Serviceanmeldekonten über ausreichende Zugriffsrechte für die ferne Registrierdatenbank verfügen. Bei der Verwendung der Variablen **DB2REMOTEPREG** sollten Sie daher in einer Windows-Domänenumgebung operieren, sodass der erforderliche Registrierungszugriff dem Domänenkonto erteilt werden kann.

Für Microsoft Cluster Server (MSCS) sind ebenfalls zwei Punkte zu beachten. In einer MSCS-Umgebung sollten Sie die Variable **DB2REMOTEPREG** nicht verwenden. Bei der Nutzung einer MSCS-Konfiguration, in der alle Computer zum gleichen MSCS-Cluster gehören, werden die Registrierdatenbankvariablen in der Clusterregistrierdatenbank gepflegt. Daher haben alle Computer im gleichen MSCS-Cluster bereits Zugriff auf sie, sodass die Variable **DB2REMOTEPREG** in diesem Fall nicht erforderlich ist.

Bei der Arbeit in einer Umgebung mit mehreren Datenbankpartitionen und Funktionsübernahmekonfiguration (Failover), in der sich Datenbankpartitionen über mehrere MSCS-Cluster erstrecken, können Sie nicht mit der Variablen

DB2REMOTEPREG auf den Instanzeignercomputer verweisen, weil sich die Registrierdatenbankvariablen des Instanzeignercomputers in der Clusterregistrierdatenbank befinden.

Deklarieren, Anzeigen, Ändern, Zurücksetzen und Löschen von Registrierdatenbank- und Umgebungsvariablen

Es wird ausdrücklich empfohlen, alle spezifischen Registrierdatenbankvariablen in der DB2-Profilregistrierdatenbank zu definieren. Wenn die DB2-Variablen außerhalb der Registrierdatenbank festgelegt werden, ist keine Fernverwaltung dieser Variablen möglich und die Workstation muss erneut gestartet werden, damit die Variablenwerte wirksam werden.

Der Befehl db2set unterstützt die lokale Deklaration der Registrierdatenbank- und Umgebungsvariablen.

Wenn Sie Hilfe für den Befehl benötigen, geben Sie Folgendes ein:

```
db2set -?
```

Zum Auflisten der vollständigen Gruppe aller unterstützten Registrierdatenbankvariablen geben Sie Folgendes ein:

```
db2set -lr
```

Verwenden Sie den folgenden Befehl, um alle definierten Registrierdatenbankvariablen für die aktuelle Instanz oder die Standardinstanz aufzulisten:

```
db2set
```

Verwenden Sie den folgenden Befehl, um alle in der Profilregistrierdatenbank definierten Registrierdatenbankvariablen aufzulisten:

```
db2set -all
```

Zum Anzeigen des Wertes einer Registrierdatenbankvariablen geben Sie Folgendes ein:

```
db2set name_der_registrierdatenbankvariablen
```

Zum Anzeigen des Wertes einer Registrierdatenbankvariablen auf allen Ebenen geben Sie Folgendes ein:

```
db2set name_der_registrierdatenbankvariablen -all
```

Zum Ändern einer Registrierdatenbankvariablen in der aktuellen Instanz oder der Standardinstanz geben Sie Folgendes ein:

```
db2set name_der_registrierdatenbankvariablen=neuer_wert
```

Zum Ändern des Standardwerts einer Registrierdatenbankvariablen für alle Datenbanken der Instanz geben Sie Folgendes ein:

```
db2set name_der_registrierdatenbankvariablen=neuer_wert
    -i instanzname
```

Zum Ändern des Standardwerts einer Registrierdatenbankvariablen für eine bestimmte Datenbankpartition einer Instanz geben Sie Folgendes ein:

```
db2set name_der_registrierdatenbankvariablen=neuer_wert
    -i instanzname datenbankpartitionsnummer
```

Zum Ändern des Standardwerts einer Registrierdatenbankvariablen für alle Instanzen, die zu einer bestimmten Installation im System gehören, geben Sie Folgendes ein:

```
db2set name_der_registrierdatenbankvariablen=neuer_wert -g
```

Wenn Sie eine kumulative Registrierdatenbankvariable wie **DB2_WORKLOAD** zum Konfigurieren Ihrer Registrierdatenbankvariablen für eine SAP-Umgebung verwenden, können Sie diese Variable wie folgt definieren:

```
db2set DB2_WORKLOAD=SAP
```

Wenn Sie mit Lightweight Directory Access Protocol (LDAP) arbeiten, können Sie Registrierdatenbankvariablen in LDAP wie folgt definieren:

- Zum Definieren von Registrierdatenbankvariablen auf Benutzerebene in LDAP geben Sie Folgendes ein:

  ```
  db2set -ul
  ```

- Zum Definieren von Registrierdatenbankvariablen auf globaler Ebene in LDAP geben Sie Folgendes ein:

  ```
  db2set -gl benutzername
  ```

Bei der Ausführung in einer LDAP-Umgebung können Sie einen Wert für eine DB2-Registrierdatenbankvariable so definieren, dass der Bereich für alle Server und Benutzer, die zu einer Verzeichnispartition oder einer Windows-Domäne gehören, global gültig ist. Gegenwärtig sind nur zwei DB2-Registrierdatenbankvariablen vorhanden, die auf globaler LDAP-Ebene definiert werden können: **DB2LDAP_KEEP_CONNECTION** und **DB2LDAP_SEARCH_SCOPE**.

Verwenden Sie zum Beispiel den folgenden Befehl, um einen Wert für DB2LDAP_SEARCH_SCOPE (Suchbereichswert in LDAP) global zu festzulegen:

```
db2set -gl db2ldap_search_scope = wert
```

Dabei kann für *wert* 'local', 'domain' oder 'global' angegeben werden.

Anmerkung:

1. Wenn die DB2-Datei profile.env von zwei oder mehr Benutzern gleichzeitig oder fast gleichzeitig mit dem Befehl db2set aktualisiert wird, wird die Größe der Datei profile.env auf null reduziert. Darüber hinaus zeigt die Ausgabe des Befehls db2set -all inkonsistente Werte.
2. Es besteht ein Unterschied zwischen der Option '-g', die zur Definition von DB2-Registrierdatenbankvariablen dient, die für alle Instanzen gelten, die zur selben Installation von DB2 ESE gehören, und der Option '-gl', die speziell auf der globalen LDAP-Ebene verwendet wird.
3. Die Registrierdatenbankvariable der Benutzerebene wird unter Windows nur unterstützt, wenn in einer LDAP-Umgebung gearbeitet wird.
4. Variableneinstellungen auf der Benutzerebene enthalten benutzerspezifische Variablenwerte. Alle Änderungen an der Benutzerebene werden in das LDAP-Verzeichnis geschrieben.
5. Die Parameter "-i", "-g", "-gl" und "-ul" können nicht gleichzeitig im selben Befehl verwendet werden.
6. Manche Variablen beziehen sich standardmäßig immer auf das Profil der globalen Ebene. (Global bedeutet, dass die Variablen von allen Instanzen, die in derselben DB2-Kopie ausgeführt werden, gemeinsam genutzt werden.) Sie können nicht in Profilen der Instanz- oder Datenbankpartitionsebene definiert werden, zum Beispiel **DB2SYSTEM** und **DB2INSTDEF**.

7. Unter UNIX müssen Sie über die Berechtigung zur Systemverwaltung (SYSADM) verfügen, um die Registrierdatenbankwerte für eine Instanz zu ändern. Nur Benutzer mit Rootberechtigung können Parameter in Registrierdatenbanken der globalen Ebene ändern.

Zum Zurücksetzen einer Registrierdatenbankvariablen für eine Instanz auf den in der globalen Profilregistrierdatenbank vorhandenen Standardwert geben Sie folgenden Befehl ein:

```
db2set -r name_der_registrierdatenbankvariablen
```

Zum Zurücksetzen einer Registrierdatenbankvariablen für eine Datenbankpartition in einer Instanz auf den in der globalen Profilregistrierdatenbank vorhandenen Standardwert geben Sie folgenden Befehl ein:

```
db2set -r name_der_registrierdatenbankvariablen datenbankpartitionsnummer
```

Zum Löschen eines Variablenwerts auf einer bestimmten Ebene können Sie die gleiche Befehlssyntax wie zum Setzen der Variablen verwenden, jedoch ohne Angabe eines Variablenwerts. Geben Sie beispielsweise folgenden Befehl ein, um die Einstellung einer Variablen auf der Datenbankpartitionsebene zu löschen:

```
db2set name_der_registrierdatenbankvariablen= -i instanzname
   datenbankpartitionsnummer
```

Geben Sie zum Löschen eines Variablenwerts und zum Begrenzen der Verwendung der Variablen, wenn sie auf einer höheren Profilebene definiert ist, folgenden Befehl ein:

```
db2set name_der_registrierdatenbankvariablen= -null -r instanzname
```

Dieser Befehl löscht die Einstellung für den von Ihnen angegebenen Parameter und verhindert, dass Profile höherer Ebene (in diesem Fall das DB2-Profil der globalen Ebene) den Wert dieser Variablen ändern. Die von Ihnen angegebene Variable kann jedoch weiterhin durch ein Profil niedrigerer Ebene (in diesem Fall das Profil auf der DB2-Datenbankpartitionsebene) definiert werden.

Definieren von Umgebungsvariablen unter Windows

Windows-Betriebssysteme verfügen über eine Systemumgebungsvariable (**DB2INSTANCE**), die nur außerhalb der Profilregistrierdatenbank festgelegt werden kann. **DB2INSTANCE** muss jedoch nicht unbedingt festgelegt werden. Die Variable **DB2INSTDEF** der DB2-Profilregistrierdatenbank kann auf der globalen Profilebene festgelegt werden, um den Instanznamen anzugeben, der verwendet wird, wenn **DB2INSTANCE** nicht definiert ist.

DB2 Enterprise Server Edition-Server unter Windows verfügen über die beiden Systemumgebungsvariablen **DB2INSTANCE** und **DB2NODE**, die nur außerhalb der Profilregistrierdatenbank festgelegt werden können. Sie müssen **DB2INSTANCE** nicht unbedingt festlegen. Die Variable **DB2INSTDEF** der DB2-Profilregistrierdatenbank kann auf der globalen Profilebene festgelegt werden, um den Instanznamen anzugeben, der verwendet wird, wenn **DB2INSTANCE** nicht definiert ist.

Die Umgebungsvariable **DB2NODE** wird zum Weiterleiten von Anforderungen an einen logischen Zielknoten innerhalb eines Computers verwendet. Diese Umgebungsvariable muss in der Sitzung festgelegt werden, in der die Anwendung bzw. der Befehl ausgeführt wird, und nicht in der DB2-Profilregistrierdatenbank.

Wenn diese Variable nicht definiert wird, wird als logischer Zielknoten standardmäßig der logische Knoten verwendet, der auf dem Computer mit der Nummer Null (0) definiert ist.

Die Einstellung einer Umgebungsvariablen lässt sich mit dem Befehl echo feststellen. Wenn Sie beispielsweise den Wert der Umgebungsvariablen **DB2PATH** prüfen wollen, geben Sie folgenden Befehl ein:

```
echo %db2path%
```

Sie können die DB2-Umgebungsvariablen **DB2INSTANCE** und **DB2NODE** (in der folgenden Beschreibung unter Verwendung von **DB2INSTANCE** erläutert) wie folgt definieren:

- Klicken Sie mit der rechten Maustaste 'Arbeitsplatz' an, und wählen Sie 'Eigenschaften' aus.
- Wählen Sie die Registerkarte 'Erweitert' aus, und klicken Sie 'Umgebungsvariablen' an. Führen Sie anschließend die folgenden Schritte aus:
  1. Wenn die Variable **DB2INSTANCE** nicht vorhanden ist:
     a. Klicken Sie 'Neu' an.
     b. Tragen Sie in das Namensfeld für die Variable **DB2INSTANCE** ein.
     c. Tragen Sie in das Wertfeld für die Variable den Instanznamen (z. B. 'db2inst') ein.
  2. Wenn die Variable **DB2INSTANCE** bereits vorhanden ist, hängen Sie wie folgt einen neuen Wert an:
     a. Wählen Sie die Umgebungsvariable **DB2INSTANCE** aus.
     b. Ändern Sie das Wertfeld in den Namen der Instanz, zum Beispiel 'db2inst'.
  3. Starten Sie Ihr System neu, damit diese Änderungen wirksam werden.

Anmerkung: Die Umgebungsvariable **DB2INSTANCE** kann auch auf Sitzungsebene (Prozessebene) festgelegt werden. Wenn Sie beispielsweise eine zweite DB2-Instanz mit dem Namen TEST starten möchten, geben Sie die folgenden Befehle in ein Befehlsfenster ein:

```
set DB2INSTANCE=TEST
db2start
```

Bei Verwendung einer C-Shell geben Sie die folgenden Befehle in einem Befehlsfenster ein:

```
setenv DB2INSTANCE TEST
```

Die Profilregistrierdatenbanken befinden sich an folgenden Positionen:

- Die DB2-Profilregistrierdatenbank auf Instanzebene in der Registrierdatenbank des Windows-Betriebssystems mit folgendem Pfad:

  \HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES*instanzname*

  Anmerkung: Dabei ist *instanzname* der Name der DB2-Instanz.
- Die DB2-Profilregistrierdatenbank auf globaler Ebene in der Registrierdatenbank von Windows mit folgendem Pfad:

  \HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\GLOBAL_PROFILE
- Die DB2-Profilregistrierdatenbank auf Instanzknotenebene in der Registrierdatenbank von Windows mit folgendem Pfad:

  ...\SOFTWARE\IBM\DB2\PROFILES*instanzname*\NODES*knotennummer*

Anmerkung: Der *instanzname* und die *knotennummer* sind für die Datenbankpartition spezifisch, in der Sie arbeiten.

- Es ist keine DB2-Profilregistrierdatenbank auf Instanzebene erforderlich. Für jede der DB2-Instanzen im System wird ein Schlüssel im folgenden Pfad erstellt:

  ```
  \HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES\instanzname
  ```

  Die Liste der Instanzen kann durch Zählen der Schlüssel unter dem Schlüssel PROFILES ermittelt werden.

Definieren von Umgebungsvariablen unter Linux- und UNIX-Betriebssystemen

Unter UNIX-Betriebssystemen müssen Sie die Systemumgebungsvariable **DB2INSTANCE** festlegen.

Die Scripts db2profile (für Bourne- oder Korn-Shell) und db2cshrc (für C-Shell) werden als Beispiele bereitgestellt, um Ihnen Hilfestellung beim Einrichten der Datenbankumgebung zu geben. Diese Dateien finden Sie im Verzeichnis insthome/sqllib, wobei `insthome` das Ausgangsverzeichnis des Instanzeigners ist.

Diese Scripts enthalten Anweisungen zu folgenden Operationen:

- Aktualisieren des Pfads eines Benutzers mit folgenden Verzeichnissen:
  - insthome/sqllib/bin
  - insthome/sqllib/adm
  - insthome/sqllib/misc
- Definieren der Variablen **DB2INSTANCE** mit dem `instanznamen` der lokal auszuführenden Standardinstanz.

Anmerkung: Alle unterstützten Variablen außer PATH und **DB2INSTANCE** müssen in der DB2-Profilregistrierdatenbank definiert werden. Definieren Sie Variablen, die vom DB2-Datenbankmanager nicht unterstützt werden, in Ihren Scriptdateien userprofile und usercshrc.

Ein Instanzeigner oder Benutzer mit der Berechtigung SYSADM kann diese Scripts für alle Benutzer einer Instanz anpassen. Alternativ können Benutzer ein Script kopieren und anpassen und anschließend das Script direkt aufrufen oder ihrer Datei .profile oder .login hinzufügen.

Zum Ändern der Umgebungsvariablen für die aktuelle Sitzung setzen Sie Befehle wie die folgenden ab:

- Für Korn-Shell:

  ```
  DB2INSTANCE=<inst1>
  export DB2INSTANCE
  ```

- Für Bourne-Shell:

  ```
  export DB2INSTANCE=<inst1>
  ```

- Für C-Shell:

  ```
  setenv DB2INSTANCE <inst1>
  ```

Zur ordnungsgemäßen Verwaltung der DB2-Profilregistrierdatenbank müssen die folgenden Regeln im Hinblick auf die Zugriffsberechtigungen und Eigentumsrechte auf UNIX-Betriebssystemen eingehalten werden.

- Die Datei der DB2-Profilregistrierdatenbank auf Instanzebene befindet sich an der folgenden Position:

  INSTHOME/sqllib/profile.env

  Die Zugriffsberechtigung und Eigentumsrechte für diese Datei sollten wie folgt definiert sein:

  -rw-rw-r-- <db2inst1> <db2iadm1> profile.env

  Dabei ist <db2inst1> der Instanzeigner und <db2iadm1> die Gruppe des Instanzeigners.

  INSTHOME ist der Ausgangspfad (HOME) des Instanzeigners.
- Die DB2-Profilregistrierdatenbank auf globaler Ebene befindet sich an der folgenden Position:

  <installationspfad>/default.env

  Dies gilt für alle Linux- und UNIX-Plattformen.

  Die Zugriffsberechtigung und Eigentumsrechte für diese Datei sollten wie folgt definiert sein:

  -rw-rw-r-- root <gruppe> default.env

  Dabei ist <gruppe> der Gruppenname für die Gruppen-ID 0 (bei AIX z. B. ″system″).

  Zum Ändern einer globalen Registrierdatenbankvariablen in Rootinstallationen müssen Sie mit Rootberechtigung angemeldet sein.
- Die DB2-Profilregistrierdatenbank auf Instanzknotenebene sich an der folgenden Position:

  INSTHOME/sqllib/nodes/<knotennummer>.env

  Die Zugriffsberechtigung und Eigentumsrechte für dieses Verzeichnis und diese Datei sollten wie folgt definiert sein:

  drwxrwsr-w <instanzeigner> <instanzeignergruppe> nodes

  -rw-rw-r-- <instanzeigner> <instanzeignergruppe> <knotennummer>.env

  INSTHOME ist der Ausgangspfad (HOME) des Instanzeigners.
- Die DB2-Profilregistrierdatenbank auf Instanzebene befindet sich an der folgenden Position:

  <installationspfad>/profiles.reg

  Dies gilt für alle Linux- und UNIX-Plattformen.

  Die Zugriffsberechtigung und Eigentumsrechte für diese Datei sollten wie folgt definiert sein:

  -rw-r--r-- root system profiles.reg

Festlegen der Umgebungsvariablen der aktuellen Instanz

Wenn Sie einen Befehl zum Starten oder Stoppen des Datenbankmanagers einer Instanz ausführen, wendet DB2 diesen Befehl auf die aktuelle Instanz an. DB2 legt die aktuelle Instanz wie folgt fest:

- Wenn die Umgebungsvariable **DB2INSTANCE** für die aktuelle Sitzung definiert ist, gibt ihr Wert die aktuelle Instanz an. Geben Sie den folgenden Befehl ein, um die Umgebungsvariable **DB2INSTANCE** zu definieren:

```
set db2instance=<neuer_instanzname>
```

- Wenn die Umgebungsvariable **DB2INSTANCE** für die aktuelle Sitzung nicht definiert ist, verwendet der DB2-Datenbankmanager die Einstellung der Umgebungsvariablen **DB2INSTANCE** aus den Systemumgebungsvariablen. Unter Windows werden die Systemumgebungsvariablen in der Registrierdatenbank für die Systemumgebung definiert.
- Wenn die Umgebungsvariable **DB2INSTANCE** überhaupt nicht definiert ist, verwendet der DB2-Datenbankmanager die Registrierdatenbankvariable **DB2INSTDEF**.

  Geben Sie den folgenden Befehl ein, um die Registrierdatenbankvariable **DB2INSTDEF** auf der globalen Ebene der Registrierdatenbank zu definieren:

  ```
  db2set db2instdef=<neuer_instanzname> -g
  ```

Geben Sie den folgenden Befehl ein, um zu ermitteln, welche Instanz für die aktuelle Sitzung relevant ist:

```
db2 get instance
```

Kumulative Registrierdatenbankvariablen

Eine kumulative Registrierdatenbankvariable ermöglicht das Zusammenfassen mehrerer Registrierdatenbankvariablen zu einer Konfigurationsgruppe, die durch einen anderen Registrierdatenbankvariablennamen identifiziert wird. Jeder Registrierdatenbankvariablen, die zu dieser Gruppe gehört, ist eine vordefinierte Einstellung zugeordnet. Der kumulativen Registrierdatenbankvariablen wird ein Wert zugeordnet, der als Deklaration mehrerer Registrierdatenbankvariablen interpretiert wird.

Zweck einer kumulativen Registrierdatenbankvariablen ist die Vereinfachung der Registrierdatenbankkonfiguration für größere Betriebszielsetzungen.

Die einzige gültige kumulative Registrierdatenbankvariable ist **DB2_WORKLOAD**.

Gültige Werte für diese Variable sind:

- SAP
- 1C
- TPM
- CM
- WC

Jede Registrierdatenbankvariable, die implizit über eine kumulative Registrierdatenbankvariable konfiguriert ist, kann außerdem explizit definiert werden. Die explizite Definition einer Registrierdatenbankvariablen, der zuvor ein Wert über eine kumulative Registrierdatenbankvariable zugeordnet wurde, ist bei der Durchführung eines Leistungs- oder Diagnosetests nützlich. Die explizite Definition einer Variablen, die durch eine kumulative Registrierdatenbankvariable implizit konfiguriert wurde, wird als Überschreiben der Variablen bezeichnet.

Wenn Sie versuchen, eine explizit definierte Registrierdatenbankvariable durch eine kumulative Registrierdatenbankvariable zu ändern, wird eine Warnung ausgegeben und der explizit definierte Wert beibehalten. In dieser Warnung werden Sie informiert, dass der explizite Wert beibehalten wird. Wird zuerst die kumulative Registrierdatenbankvariable verwendet und anschließend die explizite Registrierdatenbankvariable angegeben, erhalten Sie keine Warnung.

Keine der Registrierdatenbankvariablen, die über die Definition einer kumulativen Registrierdatenbankvariablen konfiguriert wurden, wird angezeigt, sofern Sie dies nicht für jede Variable explizit anfordern. Wenn Sie die kumulative Registrierdatenbankvariable abfragen, wird nur der dieser Variablen zugeordnete Wert angezeigt. Für die meisten Benutzer sind die Werte für die einzelnen Variablen nicht von Interesse.

Im folgenden Beispiel wird die Interaktion zwischen der Verwendung der kumulativen Registrierdatenbankvariablen und der expliziten Definition einer Registrierdatenbankvariablen dargestellt. Sie haben zum Beispiel die kumulative Registrierdatenbankvariable **DB2_WORKLOAD** auf den Wert SAP gesetzt und die Registrierdatenbankvariable **DB2_SKIPDELETED** mit NO überschrieben. Wenn Sie den Befehl db2set eingeben, würden folgende Ergebnisse angezeigt:

```
DB2_WORKLOAD=SAP
DB2_SKIPDELETED=NO
```

In einem weiteren Beispiel haben Sie möglicherweise **DB2ENVLIST** definiert, die kumulative Registrierdatenbankvariable **DB2_WORKLOAD** auf den Wert SAP gesetzt und die Registrierdatenbankvariable **DB2_SKIPDELETED** mit NO überschrieben. (In diesem Beispiel wird davon ausgegangen, dass die Registrierdatenbankvariable **DB2_SKIPDELETED** Teil der Gruppe ist, aus der die SAP-Umgebung besteht.) Für die Registrierdatenbankvariablen, die automatisch über die kumulative Registrierdatenbankvariable konfiguriert wurden, wird zusätzlich der Name der kumulativen Registrierdatenbankvariablen in eckigen Klammern neben dem Wert angezeigt. Für die Registrierdatenbankvariable **DB2_SKIPDELETED** wird der Wert `NO` sowie `[O]` neben dem Wert angezeigt.

Wenn Sie die zu **DB2_WORKLOAD** gehörige Konfiguration nicht mehr benötigen, können Sie die impliziten Werte aller Registrierdatenbankvariablen in der Gruppe inaktivieren, indem Sie den Wert der kumulativen Registrierdatenbankvariablen mit dem folgenden Befehl löschen:

```
db2set DB2_WORKLOAD=
```

Nach dem Löschen des Werts der kumulativen Registrierdatenbankvariablen **DB2_WORKLOAD** müssen Sie die Datenbank erneut starten. Nach dem Neustart der Datenbank sind die Registrierdatenbankvariablen, die implizit durch die kumulative Registrierdatenbankvariable konfiguriert wurden, nicht mehr wirksam. Der Wert einer kumulativen Registrierdatenbankvariablen wird auf dieselbe Weise gelöscht wie der Wert einer einzelnen Registrierdatenbankvariablen.

Durch das Löschen des Wertes einer kumulativen Registrierdatenbankvariablen wird der explizit definierte Wert einer Registrierdatenbankvariablen nicht gelöscht. Dabei spielt es keine Rolle, dass die Registrierdatenbankvariable Mitglied der Gruppendefinition ist, die inaktiviert wird. Die explizite Definition für die Registrierdatenbankvariable bleibt bestehen.

Sie können die Werte für die einzelnen Registrierdatenbankvariablen anzeigen, die Teil der kumulativen Registrierdatenbankvariablen **DB2_WORKLOAD** sind. Sie wollen zum Beispiel die Werte prüfen, die verwendet würden, wenn Sie die Variable **DB2_WORKLOAD** auf den Wert SAP setzen würden. Zum Ermitteln der Werte, die bei **DB2_WORKLOAD**=SAP verwendet würden, führen Sie den Befehl `db2set -gd DB2_WORKLOAD=SAP`.

Registrierdatenbank- und Umgebungsvariablen von DB2

Die DB2-Produkte stellen eine Reihe von Registrierdatenbankvariablen und Umgebungsvariablen bereit, die Sie kennen sollten, um DB2 betriebsbereit zu machen.

Zum Anzeigen einer Liste aller unterstützten Registrierdatenbankvariablen führen Sie den folgenden Befehl aus:

```
db2set -lr
```

Zum Ändern des Werts für eine Variable in der aktuellen Instanz bzw. in der Standardinstanz führen Sie den folgenden Befehl aus:

```
db2set name_der_registrierdatenbankvariablen=neuer_wert
```

Ob die DB2-Umgebungsvariablen **DB2INSTANCE**, **DB2NODE**, **DB2PATH** und **DB2INSTPROF** in den DB2-Profilregistrierdatenbanken gespeichert werden, hängt vom verwendeten Betriebssystem ab. Zur Aktualisierung dieser Umgebungsvariablen verwenden Sie den Befehl set. Diese Änderungen werden nur für die lokale (aktuelle) Eingabeaufforderung wirksam und bleiben bis zum nächsten Neustart des Systems wirksam. Unter Linux- und UNIX-Betriebssystemen können Sie den Befehl export anstelle des Befehls set verwenden.

Sie müssen die Werte für die geänderten Registrierdatenbankvariablen definieren, bevor Sie den Befehl db2start ausführen.

Anmerkung: Wenn eine Registrierdatenbankvariable Boolesche Werte als Argumente erfordert, sind einerseits die Werte YES, 1 und ON sowie andererseits die Werte NO, 0 und OFF äquivalent. Für jede Variable können Sie einen beliebigen der entsprechenden äquivalenten Werte angeben.

In der folgenden Tabelle sind alle Registrierdatenbankvariablen nach Kategorie aufgelistet:

Tabelle 63. Registrierdatenbank- und Umgebungsvariablen - Übersicht

Variablenkategorie	Name der Registrierdatenbank- bzw. Umgebungsvariablen
Allgemein	**DB2ACCOUNT** **DB2BIDI** **DB2_CAPTURE_LOCKTIMEOUT** **DB2CODEPAGE** **DB2_COLLECT_TS_REC_INFO** **DB2_CONNRETRIES_INTERVAL** **DB2CONSOLECP** **DB2COUNTRY** **DB2DBDFT** **DB2DBMSADDR** **DB2DISCOVERYTIME** **DB2FFDC** **DB2FODC** **DB2_FORCE_APP_ON_MAX_LOG** **DB2GRAPHICUNICODESERVER** **DB2INCLUDE** **DB2INSTDEF** **DB2INSTOWNER** **DB2_LIC_STAT_SIZE** **DB2LOCALE** **DB2_MAX_CLIENT_CONNRETRIES** **DB2_OBJECT_TABLE_ENTRIES** **DB2_SYSTEM_MONITOR_SETTINGS** **DB2TERRITORY** **DB2_VIEW_REOPT_VALUES**
Systemumgebung	**DB2_ALTERNATE_GROUP_LOOKUP** **DB2_CLP_EDITOR** **DB2_CLP_HISTSIZE** **DB2CONNECT_IN_APP_PROCESS** **DB2_COPY_NAME** **DB2DBMSADDR** **DB2_DIAGPATH** **DB2DOMAINLIST** **DB2ENVLIST** **DB2INSTANCE** **DB2INSTPROF** **DB2LDAPSecurityConfig** **DB2LIBPATH** **DB2LOGINRESTRICTIONS** **DB2NODE** **DB2OPTIONS** **DB2_PARALLEL_IO** **DB2PATH** **DB2PROCESSORS** **DB2RCMD_LEGACY_MODE** **DB2SYSTEM** **DB2_UPDDBCFG_SINGLE_DBPARTITION** **DB2_USE_PAGE_CONTAINER_TAG** **DB2_WORKLOAD**

Tabelle 63. Registrierdatenbank- und Umgebungsvariablen - Übersicht (Forts.)

Variablenkategorie	Name der Registrierdatenbank- bzw. Umgebungsvariablen
Kommunikation	**DB2CHECKCLIENTINTERVAL** **DB2COMM** **DB2FCMCOMM** **DB2_FORCE_NLS_CACHE** **DB2RSHCMD** **DB2RSHTIMEOUT** **DB2SORCVBUF** **DB2SOSNDBUF** **DB2TCP_CLIENT_CONTIMEOUT** **DB2TCP_CLIENT_RCVTIMEOUT** **DB2TCPCONNMGRS**
Befehlszeile	**DB2BQTIME** **DB2BQTRY** **DB2_CLPPROMPT** **DB2IQTIME** **DB2RQTIME**
Umgebung mit partitionierten Datenbanken	**DB2CHGPWD_EEE** **DB2_FCM_SETTINGS** **DB2_NUM_FAILOVER_NODES** **DB2_PARTITIONEDLOAD_DEFAULT** **DB2PORTRANGE**
Abfragecompiler	**DB2_ANTIJOIN** **DB2_INLIST_TO_NLJN** **DB2_LIKE_VARCHAR** **DB2_MINIMIZE_LISTPREFETCH** **DB2_NEW_CORR_SQ_FF** **DB2_OPT_MAX_TEMP_SIZE** **DB2_REDUCED_OPTIMIZATION** **DB2_SELECTIVITY** **DB2_SQLROUTINE_PREPOPTS**

Tabelle 63. Registrierdatenbank- und Umgebungsvariablen - Übersicht (Forts.)

Variablenkategorie	Name der Registrierdatenbank- bzw. Umgebungsvariablen
Leistung	**DB2_ALLOCATION_SIZE** **DB2_APM_PERFORMANCE** **DB2ASSUMEUPDATE** **DB2_ASYNC_IO_MAXFILOP** **DB2_AVOID_PREFETCH** **DB2BPVARS** **DB2CHKPTR** **DB2CHKSQLDA** **DB2_EVALUNCOMMITTED** **DB2_EXTENDED_IO_FEATURES** **DB2_EXTENDED_OPTIMIZATION** **DB2_IO_PRIORITY_SETTING** **DB2_IO_PRIORITY_SETTING** **DB2_KEEP_AS_AND_DMS_CONTAINERS_OPEN** **DB2_KEEPTABLELOCK** **DB2_LARGE_PAGE_MEM** **DB2_LOGGER_NON_BUFFERED_IO** **DB2MAXFSCRSEARCH** **DB2_MAX_INACT_STMTS** **DB2_MAX_NON_TABLE_LOCKS** **DB2_MDC_ROLLOUT** **DB2MEMDISCLAIM** **DB2MEMMAXFREE** **DB2_MEM_TUNING_RANGE** **DB2_MMAP_READ** **DB2_MMAP_WRITE** **DB2_NO_FORK_CHECK** **DB2NTMEMSIZE** **DB2NTNOCACHE** **DB2NTPRICLASS** **DB2NTWORKSET** **DB2_OVERRIDE_BPF** **DB2_PINNED_BP** **DB2PRIORITIES** **DB2_RESOURCE_POLICY** **DB2_RCT_FEATURES** **DB2_SET_MAX_CONTAINER_SIZE** **DB2_SKIPDELETED** **DB2_SKIPINSERTED** **DB2_SMS_TRUNC_TMPTABLE_THRESH** **DB2_SORT_AFTER_TQ** **DB2_SELUDI_COMM_BUFFER** **DB2_TRUSTED_BINDIN** **DB2_USE_ALTERNATE_PAGE_CLEANING** **DB2_USE_IOCP**

Tabelle 63. Registrierdatenbank- und Umgebungsvariablen - Übersicht (Forts.)

Variablenkategorie	Name der Registrierdatenbank- bzw. Umgebungsvariablen
Verschiedene Variablen	**DB2ADMINSERVER** **DB2_ATS_ENABLE** **DB2AUTH** **DB2CLIINIPATH** **DB2_COMMIT_ON_EXIT** **DB2_CREATE_DB_ON_PATH** **DB2DEFPREP** **DB2_DISABLE_FLUSH_LOG** **DB2_DISPATCHER_PEEKTIMEOUT** **DB2_DJ_INI** **DB2DMNBCKCTLR** **DB2_DOCHOST** **DB2_DOCPORT** **DB2_ENABLE_AUTOCONFIG_DEFAULT** **DB2_ENABLE_LDAP** **DB2_EVMON_EVENT_LIST_SIZE** **DB2_EVMON_STMT_FILTER** **DB2_EXTSECURITY** **DB2_FALLBACK** **DB2_FMP_COMM_HEAPSZ** **DB2_GRP_LOOKUP** **DB2_HADR_BUF_SIZE** **DB2_HADR_NO_IP_CHECK** **DB2_HADR_PEER_WAIT_LIMIT** **DB2_HADR_SORCVBUF** **DB2_HADR_SOSNDBUF** **DB2LDAP_BASEDN** **DB2LDAPCACHE** **DB2LDAP_CLIENT_PROVIDER** **DB2LDAPHOST** **DB2LDAP_KEEP_CONNECTION** **DB2LDAP_SEARCH_SCOPE** **DB2_LOAD_COPY_NO_OVERRIDE** **DB2LOADREC** **DB2LOCK_TO_RB** **DB2_MAP_XML_AS_CLOB_FOR_DLC** **DB2_MAX_LOB_BLOCK_SIZE** **DB2_MEMORY_PROTECT** **DB2NOEXITLIST** **DB2_NUM_CKPW_DAEMONS** **DB2_OPTSTATS_LOG** **DB2REMOTEPREG** **DB2_RESOLVE_CALL_CONFLICT** **DB2ROUTINE_DEBUG** **DB2SATELLITEID** **DB2_SERVER_CONTIMEOUT** **DB2_SERVER_ENCALG** **DB2SORT** **DB2_THREAD_SUSPENSION** **DB2_TRUNCATE_REUSESTORAGE** **DB2_USE_DB2JCCT2_JROUTINE** **DB2_UTIL_MSGPATH** **DB2_VENDOR_INI** **DB2_XBSA_LIBRARY**

Allgemeine Registrierdatenbankvariablen

DB2ACCOUNT

- Betriebssystem: Alle
- Standardwert: NULL
- Diese Variable definiert die Abrechnungszeichenfolge, die an den fernen Host gesendet wird. Einzelheiten siehe DB2 Connect - Benutzerhandbuch.

DB2BIDI

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO
- Diese Variable aktiviert die bidirektionale Unterstützung von Sprachen, und die Variable **DB2CODEPAGE** dient zur Deklaration der zu verwendenden Codepage.

DB2_CAPTURE_LOCKTIMEOUT

- Betriebssystem: Alle
- Standardwert: NULL, Werte: ON oder NULL
- Diese Variable gibt an, dass beschreibende Informationen zu Sperrzeitlimitüberschreitungen zu dem Zeitpunkt zu protokollieren sind, zu dem sie auftreten. Folgende Informationen werden protokolliert: Die Hauptanwendungen, die an dem Sperrenkonflikt beteiligt sind, der zu der Sperrzeitlimitüberschreitung führte, die Details darüber, was diese Anwendungen ausführten, als die Überschreitung des Sperrzeitlimits auftrat, sowie die Details über die Sperre, die den Zugriffskonflikt verursacht hat. Die Informationen werden über den Anforderer der Sperre (die Anwendung, die den Zeitlimitfehler empfangen hat) und den aktuellen Sperreneigner erfasst. Zu jeder Sperrzeitlimitüberschreitung wird ein Textbericht geschrieben und in einer Datei gespeichert.

  Die Dateien werden mit der folgenden Namenskonvention erstellt: db2locktimeout.*par*.*AGENTID*.*jjjj-mm-tt-hh-mm-ss*. Dabei ist *par* die Datenbankpartitionsnummer, *AGENTID* ist die Agenten-ID und *jjjj-mm-tt-hh-mm-ss* ist die Zeitmarke, die aus Jahr, Monat, Tag, Stunde, Minuten und Sekunden besteht. In einer nicht partitionierten Datenbankumgebung hat *par* den Wert `0`.

  Die Position der Datei hängt von dem Wert ab, der im Datenbankkonfigurationsparameter **diagpath** angegeben ist. Wenn der Parameter **diagpath** nicht definiert ist, befindet sich die Datei in einem der folgenden Verzeichnisse:

  – In Windows-Umgebungen:
    - Wenn Sie die Umgebungsvariable **DB2INSTPROF** nicht definiert haben, werden die Informationen in das folgende Verzeichnis geschrieben: *x*:*SQLLIB**DB2INSTANCE*. Dabei ist *x* die Laufwerkangabe, *SQLLIB* ist das Verzeichnis, das Sie für die Registrierdatenbankvariable **DB2PATH** angegeben haben, und *DB2INSTANCE* ist der Name des Instanzeigners.
    - Wenn Sie die Umgebungsvariable **DB2INSTPROF** definieren, werden die Informationen in das folgende Verzeichnis geschrieben: *x*:*DB2INSTPROF**DB2INSTANCE*. Dabei ist *x* die Laufwerkangabe, *DB2INSTPROF* der Name des Instanzprofilverzeichnisses und *DB2INSTANCE* ist der Name des Instanzeigners.

– In Linux- und UNIX-Umgebungen: Informationen werden in das Verzeichnis *INSTHOME*/sqllib/db2dump geschrieben. Dabei ist *INSTHOME* das Ausgangsverzeichnis der Instanz.

Löschen Sie Berichtsdateien zu Sperrzeitlimitüberschreitungen, wenn Sie sie nicht mehr benötigen. Da diese Berichtsdateien an derselben Position wie andere Diagnoseprotokolle gespeichert werden, könnte das DB2-System herunterfahren, wenn sich das Verzeichnis vollständig füllt. Wenn Sie einige Berichtsdateien zu Sperrzeitlimitüberschreitungen behalten wollen, versetzen Sie sie in ein anderes Verzeichnis (bzw. einen anderen Ordner) als das, in dem die DB2-Protokolle gespeichert werden.

DB2CODEPAGE

- Betriebssystem: Alle
- Standardwert: abgeleitet aus der vom Betriebssystem angegebenen Sprachenkennung
- Diese Variable gibt die Codepage der Daten an, die an DB2 für Datenbankclientanwendungen übergeben werden. Der Benutzer sollte **DB2CODEPAGE** nicht definieren, sofern dies nicht explizit in DB2-Dokumenten angegeben ist oder er vom DB2-Service dazu angeleitet wird. Wird **DB2CODEPAGE** auf einen Wert gesetzt, der vom Betriebssystem nicht unterstützt wird, können unerwartete Ergebnisse auftreten. Im Normalfall brauchen Sie **DB2CODEPAGE** nicht zu definieren, weil DB2 die Informationen zur Codepage automatisch aus dem Betriebssystem abruft.

  Anmerkung: Da Windows keine Unicode-Codepage (in den Ländereinstellungen von Windows) anstelle der ANSI-Codepage meldet, verhält sich eine Windows-Anwendung nicht wie ein Unicode-Client. Um dieses Verhalten außer Kraft zu setzen, setzen Sie die Registrierdatenbankvariable **DB2CODEPAGE** auf 1208 (für die Unicode-Codepage), damit die Anwendung sich wie eine Unicode-Anwendung verhält.

DB2_COLLECT_TS_REC_INFO

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON oder OFF
- Diese Variable gibt an, ob DB2 alle Protokolldateien bei einer aktualisierenden Recovery eines Tabellenbereichs verarbeitet, unabhängig davon, ob die Protokolldateien Protokollsätze enthalten, die den Tabellenbereich betreffen. Wenn die Protokolldateien übersprungen werden sollen, die bekanntermaßen keine Protokollsätze für diesen Tabellenbereich enthalten, setzen Sie diese Variable auf den Wert ON.
  DB2_COLLECT_TS_REC_INFO muss definiert werden, bevor die Protokolldateien erstellt und verwendet werden, sodass die Informationen, die zum Überspringen von Protokolldateien erforderlich sind, erfasst werden.

DB2_CONNRETRIES_INTERVAL

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: ganzzahlige Werte in Sekunden
- Die Variable gibt die Ruhezeit (in Sekunden) zwischen aufeinanderfolgenden Verbindungsversuchen für die automatische Clientweiterleitungsfunktion an. Sie können diese Variable in Verbindung mit **DB2_MAX_CLIENT_CONNRETRIES** verwenden, um das Wiederholungsverhalten für die automatische Clientweiterleitung zu konfigurieren.

Wenn **DB2_MAX_CLIENT_CONNRETRIES** definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** nicht, nimmt **DB2_CONNRETRIES_INTERVAL** den Standardwert 30 an. Wenn **DB2_MAX_CLIENT_CONNRETRIES** nicht definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** definiert ist, gilt für **DB2_MAX_CLIENT_CONNRETRIES** der Standardwert 10. Wenn weder **DB2_MAX_CLIENT_CONNRETRIES** noch **DB2_CONNRETRIES_INTERVAL** definiert ist, kehrt die Funktion der automatischen Clientweiterleitung zur Standardfunktionsweise zurück. Das heißt, sie versucht die Verbindung zu einer Datenbank über eine Dauer von bis zu 10 Minuten wiederholt herzustellen.

DB2CONSOLECP

- Betriebssystem: Windows
- Standardwert: NULL, Werte: alle gültigen Codepagewerte
- Gibt die Codepage für die Anzeige von DB2- Nachrichtentext an. Wenn angegeben, überschreibt dieser Wert die Einstellung für die Codepage des Betriebssystems.

DB2COUNTRY

- Betriebssystem: Windows
- Standardwert: NULL, Werte: alle gültigen numerischen Codes für Land, Gebiet oder Region
- Diese Variable gibt den Code für Land, Gebiet oder Region der Clientanwendung an. Wenn angegeben, überschreibt dieser Wert die Einstellung des Betriebssystems.

Anmerkung: **DB2COUNTRY** ist veraltet und wird in einem zukünftigen Release möglicherweise entfernt. Verwenden Sie stattdessen die Variable DB2TERRITORY, die die gleichen Werte wie **DB2COUNTRY** akzeptiert.

DB2DBDFT

- Betriebssystem: Alle
- Standardwert: NULL
- Diese Variable gibt den Aliasnamen der Datenbank an, die für implizite Verbindungen zu verwenden ist. Wenn eine Anwendung keine Datenbankverbindung hat, aber SQL- oder XQuery-Anweisungen abgesetzt werden, wird eine implizite Verbindung hergestellt, wenn die Umgebungsvariable **DB2DBDFT** mit einer Standarddatenbank definiert wurde.

DB2DBMSADDR

- Betriebssystem: Windows (32 Bit)
- Standardwert: 0x20000000, Werte: 0x20000000 bis 0xB0000000 in Inkrementen von 0x10000
- Diese Variable gibt die Standardadresse des Datenbankmanagers für gemeinsam benutzten Speicher im Hexadezimalformat an. Wenn der Befehl db2start aufgrund einer Adresskollision beim gemeinsam benutzen Speicher fehlschlägt, kann diese Registrierdatenbankvariable geändert werden, um die Datenbankmanagerinstanz zu zwingen, den gemeinsam benutzen Speicher an einer anderen Adresse anzulegen.

DB2DISCOVERYTIME

- Betriebssystem: Windows
- Standardwert: 40 Sekunden, Mindestwert: 20 Sekunden

- Diese Variable gibt die Zeitdauer an, die der Discovery-Prozess SEARCH nach DB2-Systemen sucht.

DB2_EXPRESSION_RULES

- Betriebssystem: Alle
- Standardwert: Leer, Werte: RAISE_ERROR_PERMIT_SKIP oder RAISE-_ERROR_PERMIT_DROP
- Die Einstellungen für die Registrierdatenbankvariable **DB2_EXPRESSION_RULES** steuern, wie das DB2-Optimierungsprogramm den Zugriffsplan für Abfragen bestimmt, die die Funktion RAISE_ERROR enthalten. Das Standardverhalten der Funktion RAISE-_ERROR besteht darin, dass keine Filterung über den Ausdruck hinaus, der diese Funktion enthält, verschoben werden kann. Dies kann dazu führen, dass keine Vergleichselemente während Tabellenzugriffen angewendet werden, was einen übermäßigen Berechnungsaufwand für Ausdrücke, übermäßiges Sperren und eine schlechte Abfrageleistung zur Folge haben kann.

  In bestimmten Fällen ist dieses Verhalten zu strikt. Je nach den jeweiligen Geschäftsanforderungen der Anwendung spielt es möglicherweise keine Rolle, ob Vergleichselemente und Joins angewendet werden, bevor die Funktion RAISE_ERROR angewendet wird. Im Kontext einer Sicherheitsimplementierung auf Zeilenebene ist zum Beispiel in der Regel ein Ausdruck der folgenden Form vorhanden:

  ```
  CASE WHEN <bedingungen zur prüfung des zugriffs auf diese zeile>
       THEN NULL
       ELSE RAISE_ERROR(...)
  END
  ```

  Für die Anwendung ist es vielleicht nur wichtig, den Zugriff auf die Zeilen zu prüfen, die von der Abfrage ausgewählt werden, und nicht den Zugriff auf jede Zeile in der Tabelle zu prüfen. Daher könnten Vergleichselemente beim Zugriff auf die Basistabelle angewendet werden und der Ausdruck, der die Funktion RAISE_ERROR enthält, muss nur ausgeführt werden, nachdem sämtliche Filterungen erfolgt sind. In diesem Fall kann **DB2_EXPRESSION_RULES**=RAISE_ERROR_PERMIT_SKIP ein geeigneter Wert sein.

  Eine weitere Alternative ergibt sich im Kontext der Sicherheit auf Spaltenebene (COLUMN LEVEL). In diesem Fall sind in der Regel Ausdrücke der folgenden Form vorhanden:

  ```
  CASE WHEN <bedingungen zur prüfung des zugriffs auf diese zeile und spalte>
       THEN <tabelle.spalte>
       ELSE RAISE_ERROR(...)
  END
  ```

  In diesem Fall bezweckt die Anwendung vielleicht, dass Fehler nur ausgelöst werden sollen, wenn der Benutzer versucht, die Daten für eine bestimmte Zeile zu empfangen, und eine bestimmte Spalte einen Wert enthält, den der Benutzer nicht abrufen darf. In diesem Fall bewirkt die Einstellung **DB2_EXPRESSION_RULES**=RAISE_ERROR_PERMIT_DROP, dass der Ausdruck mit der Funktion RAISE_ERROR nur ausgewertet wird, wenn die betreffende Spalte von einem Vergleichselement oder einer Spaltenfunktion verwendet wird oder wenn sie als Ausgabe von der Abfrage zurückgegeben wird.

DB2FFDC

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON, CORE:OFF
- Diese Variable bietet die Möglichkeit, die Generierung der Kerndatei zu inaktivieren. Standardmäßig ist diese Registrierdatenbankvariable auf ON gesetzt. Wenn diese Registrierdatenbankvariable nicht definiert oder auf einen anderen Wert als CORE:OFF gesetzt ist, werden die Kerndateien möglicherweise generiert, wenn das DB2-Datenbanksystem abnormal beendet wird.

  Kerndateien, die zur Fehlerbestimmung verwendet und in dem durch **diagpath** angegebenen Verzeichnis erstellt werden, enthalten das gesamte Prozessimage des DB2-Prozesses, der beendet wird. Ein besonderes Augenmerk sollte dem verfügbaren Dateisystemspeicher gelten, da Kerndateien recht groß sein können. Die Größe hängt von der DB2-Datenbanksystemkonfiguration und dem Status des Prozesses zum Zeitpunkt des Auftretens des Problems ab.

  Unter Linux-Betriebssystemen ist die Standardgrößenbegrenzung für Kerndateien auf `0` gesetzt (d. h., `ulimit -c`). Bei dieser Einstellung werden keine Kerndateien generiert. Wenn Kerndateien auf Linux-Plattformen erstellt werden sollen, muss der Wert unbegrenzt sein.

Anmerkung: **DB2FFDC** wird ab Version 9.5 als veraltet betrachtet und in einem späteren Release entfernt. Die neue Registrierdatenbankvariable **DB2FODC** enthält die Funktionalität von **DB2FFDC**.

DB2FODC

- Betriebssystem: Alle
- Standardwert: die Verkettung aller FODC-Parameter (First Occurrence Data Collection, Datenerfassung beim ersten Vorkommen), siehe unten
  - Für Linux und UNIX: "CORELIMIT=*wert* DUMPCORE=ON DUMPDIR=*diagpath*"
  - Für Windows: "DUMPCORE=ON DUMPDIR=*diagpath*"

  Beachten Sie, dass die Parameter durch Leerzeichen getrennt angegeben werden.
- Diese Registrierdatenbankvariable steuert eine Gruppe von Fehlerdiagnoseparametern, die bei der FODC verwendet werden. Mit der Variablen **DB2FODC** können verschiedene Aspekte der Datenerfassung in Ausfallsituationen gesteuert werden.

  Diese Registrierdatenbankvariable wird einmal während des Starts der DB2-Instanz gelesen. Wenn Sie die FODC-Parameter online aktualisieren wollen, verwenden Sie das Tool db2pdcfg. Mit der Registrierdatenbankvariablen **DB2FODC** können Sie die Konfiguration über Neustarts hinweg beibehalten. Es müssen weder alle Parameter angegeben werden noch müssen sie in einer bestimmten Reihenfolge angegeben werden. Jedem Parameter, der nicht angegeben wird, wird der Standardwert zugewiesen. Wenn Sie zum Beispiel keinen Auszug der Kerndateien erstellen wollen, während sich die anderen Parameter standardmäßig verhalten sollen, geben Sie den folgenden Befehl ein:

  ```
  db2set DB2FODC="DUMPCORE=OFF"
  ```

  Parameter:

  CORELIMIT

– Betriebssystem: Linux und UNIX
– Standardwert: Aktueller Wert für 'ulimit', Werte 0 bis unlimited
– Diese Option gibt die maximale Größe (in GB) der Kerndateien an, die erstellt werden. Dieser Wert überschreibt die aktuelle Einstellung für die Größenbegrenzung von Kerndateien. Ein besonderes Augenmerk sollte dem verfügbaren Dateisystemspeicher gelten, da Kerndateien recht groß sein können. Die Größe hängt von der DB2-Konfiguration und dem Status des Prozesses zum Zeitpunkt des Auftretens des Problems ab.

  Wenn **CORELIMIT** definiert ist, verwendet DB2 diesen Wert, um die aktuelle Einstellung für die maximale Größe von Kerndateien für Benutzer (ulimit) zum Generieren der Kerndatei zu überschreiben.

  Wenn **CORELIMIT** nicht angegeben wird, setzt DB2 die Kerndateigröße auf den Wert, der der aktuellen ulimit-Einstellung entspricht. Eine Ausnahme gilt für AIX: Die ulimit-Einstellung "unlimited" (unbegrenzt) wird nur für DB2-Serverprozesse mit dem Wert 8 GB überschrieben. Wenn Sie einen größeren Kerndateispeicherauszug als 8 GB benötigen, setzen Sie 'ulimit' auf einen entsprechend großen Wert, zum Beispiel auf die Größe des RAM-Speichers, oder setzen Sie **CORELIMIT** auf einen ausreichend großen Wert.

  Anmerkung: Alle Änderungen an der Größenbegrenzung von Kerndateien für Benutzer bzw. an **CORELIMIT** werden erst nach dem nächsten Neustart der DB2-Instanz wirksam.

DUMPCORE

– Betriebssystem: Linux, Solaris, AIX
– Standardwert: ON, Werte: ON oder OFF
– Diese Option gibt an, ob die Generierung von Kerndateien erfolgen soll. Kerndateien, die zur Fehlerbestimmung verwendet und in dem durch **diagpath** angegebenen Verzeichnis erstellt werden, enthalten das gesamte Prozessimage des DB2-Prozesses, der beendet wird. Ob allerdings tatsächlich ein Kerndateispeicherauszug erfolgt, hängt von der aktuellen Einstellung von 'ulimit' und dem Wert des Parameters **CORELIMIT** ab. Einige Betriebssysteme besitzen auch Konfigurationseinstellungen für Kernspeicherauszüge, die das Verhalten von Anwendungsspeicherauszügen möglicherweise vorgeben.

  Die empfohlene Methode zur Inaktivierung von Kerndateispeicherauszügen besteht darin, den Parameter **DUMPCORE** auf OFFzu setzen.

DUMPDIR

– Betriebssystem: Alle
– Standardwert: **diagpath**-Verzeichnis oder das Standarddiagnoseverzeichnis, wenn **diagpath** nicht definiert ist, Werte: *pfad_zu_verzeichnis*
– Diese Option gibt den absoluten Pfadnamen des Verzeichnisses für die Erstellung von Kerndateien an. Diese Option kann auch für andere große Binärspeicherauszüge, die außerhalb

des FODC-Pakets gespeichert werden müssen, und nicht nur für Kerndateispeicherauszüge verwendet werden.

DB2_FORCE_APP_ON_MAX_LOG

- Betriebssystem: Alle
- Standardwert: TRUE, Werte: TRUE oder FALSE
- Gibt an, was geschieht, wenn der Wert des Konfigurationsparameters **max_log** überschritten wird. Bei TRUE wird die Anwendung zwangsweise von der Datenbank getrennt und die UOW (Unit of Work, Arbeitseinheit) mit ROLLBACK rückgängig gemacht.

  Bei FALSE schlägt die aktuelle Anweisung fehl. Die Anwendung kann weiterhin von vorherigen Anweisungen in der UOW fertiggestellte Arbeit mit COMMIT festschreiben, oder sie kann die ausgeführte Arbeit mit ROLLBACK rückgängig machen, um die UOW zurückzusetzen.

  Anmerkung: Diese DB2-Registrierdatenbankvariable beeinflusst die Fähigkeit des Importdienstprogramms zur Wiederherstellung nach Situationen, in denen der Platz im Protokoll nicht ausreicht. Wenn für **DB2_FORCE_APP_ON_MAX_LOG** der Wert TRUE definiert wird und Sie den Befehl IMPORT mit der Befehlsoption **COMMITCOUNT** eingeben, kann das Importdientsprogramm kein Commit durchführen, um Speicherknappheit im aktiven Protokoll zu vermeiden. Wenn das Importdienstprogramm den Fehler SQL0964C (Transaktionsprotokoll voll) feststellt, wird seine Verbindung zur Datenbank getrennt, und die aktuelle UOW wird rückgängig gemacht.

DB2GRAPHICUNICODESERVER

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Diese Registrierdatenbankvariable dient zur Zulassung vorhandener Anwendungen, die zum Einfügen von Grafikdaten in eine Unicode-Datenbank geschrieben sind. Ihre Verwendung ist nur für Anwendungen erforderlich, die sqldbchar-Daten (Grafikdaten) speziell in Unicode anstatt in der Codepage des Clients senden. ('sqldbchar' ist ein unterstützter SQL-Datentyp in C und C++, der ein einzelnes Doppelbytezeichen aufnehmen kann.) Wenn Sie diese Registrierdatenbankvariable auf den Wert ON setzen, teilen Sie der Datenbank mit, dass der Eingang von Unicode-Grafikdaten zu erwarten ist, und die Anwendung erwartet ebenfalls, Unicode-Grafikdaten zu empfangen.

DB2INCLUDE

- Betriebssystem: Alle
- Standardwert: Aktuelles Verzeichnis
- Gibt einen Pfad an, der bei der Verarbeitung der SQL-Anweisung INCLUDE textdatei während der Ausführung des DB2-Befehls PREP zu verwenden ist. Er gibt eine Liste von Verzeichnissen an, in denen sich die INCLUDE-Datei befinden könnte. Im Handbuch Developing Embedded SQL Applications finden Sie Beschreibungen, wie **DB2INCLUDE** in den verschiedenen vorkompilierten Sprachen verwendet wird.

DB2INSTDEF

- Betriebssystem: Windows
- Standardwert: DB2
- Diese Variable definiert den Wert, der zu verwenden ist, wenn **DB2INSTANCE** nicht definiert ist.

DB2INSTOWNER

- Betriebssystem: Windows
- Standardwert: NULL
- Die Registrierdatenbankvariable, die bei der ersten Erstellung der Instanz in der DB2-Profilregistrierdatenbank erstellt wird. Diese Variable wird auf den Namen der Instanzeignermaschine gesetzt.

DB2_LIC_STAT_SIZE

- Betriebssystem: Alle
- Standardwert: NULL, Bereich: 0 bis 32767
- Diese Variable legt die Maximalgröße (in MB) der Datei mit den Lizenzstatistikdaten für das System fest. Der Wert 0 schaltet das Sammeln von Lizenzstatistikdaten aus. Wenn die Variable nicht erkannt wird oder nicht definiert ist, wird standardmäßig eine uneingeschränkte Größe angenommen. Die Statistikdaten werden über die Lizenzzentrale angezeigt.

DB2LOCALE

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO
- Diese Variable gibt an, ob das Standardlocale ″C″ eines Prozesses nach dem Aufrufen von DB2 auf das Standardlocale ″C″ zurückgesetzt wird und ob das Prozesslocale nach dem Aufrufen einer DB2-Funktion auf das ursprüngliche 'C' zurückzusetzen ist. Wenn das ursprüngliche Locale nicht 'C' war, wird diese Registrierdatenbankvariable ignoriert.

DB2_MAX_CLIENT_CONNRETRIES

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: eine ganze Zahl für die maximale Anzahl von Wiederholungsversuchen zur Herstellung der Verbindung
- Diese Variable gibt die maximale Anzahl von Verbindungsversuchen an, die von der Funktion zur automatischen Clientweiterleitung unternommen werden sollen. Sie können diese Variable in Verbindung mit **DB2_CONNRETRIES_INTERVAL** verwenden, um das Wiederholungsverhalten für die automatische Clientweiterleitung zu konfigurieren.

  Wenn **DB2_MAX_CLIENT_CONNRETRIES** definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** nicht, nimmt **DB2_CONNRETRIES_INTERVAL** den Standardwert 30 an. Wenn **DB2_MAX_CLIENT_CONNRETRIES** nicht definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** definiert ist, gilt für **DB2_MAX_CLIENT_CONNRETRIES** der Standardwert 10. Wenn weder **DB2_MAX_CLIENT_CONNRETRIES** noch **DB2_CONNRETRIES_INTERVAL** definiert ist, kehrt die Funktion der automatischen Clientweiterleitung zur Standardfunktionsweise zurück. Das heißt, sie versucht die Verbindung zu einer Datenbank über eine Dauer von bis zu 10 Minuten wiederholt herzustellen.

DB2_OBJECT_TABLE_ENTRIES

- Betriebssystem: Alle
- Standardwert: 0, Werte: 0 bis 65532

  Der tatsächliche Maximalwert, der auf Ihrem System möglich ist, hängt von der Seitengröße und der Speicherbereichsgröße (EXTENTSIZE) ab, kann jedoch den Wert 65.532 nicht überschreiten.

- Diese Variable gibt die erwartete Anzahl von Objekten in einem Tabellenbereich an. Wenn Sie wissen, dass in einem DMS-Tabellenbereich eine große Anzahl von Objekten (z. B. 1000 oder mehr) erstellt wird, sollten Sie diese Registrierdatenbankvariable auf die ungefähre Anzahl setzen, bevor Sie den Tabellenbereich erstellen. Dadurch wird zusammenhängender Speicherplatz für Objektmetadaten bei der Erstellung des Tabellenbereichs reserviert. Die Reservierung von Speicher verringert die Wahrscheinlichkeit, dass ein Online-Backup Operationen blockiert, die Einträge in den Metadaten aktualisieren (wie z. B. CREATE INDEX, IMPORT REPLACE). Sie erleichtert außerdem eine spätere Änderung der Größe des Tabellenbereichs, weil die Metadaten am Anfang des Tabellenbereichs gespeichert werden.

  Wenn die Anfangsgröße des Tabellenbereichs nicht ausreichend groß ist, um zusammenhängenden Speicher zu reservieren, wird die Erstellung des Tabellenbereichs fortgesetzt, ohne den zusätzlichen Speicher zu reservieren.

DB2_SYSTEM_MONITOR_SETTINGS

- Betriebssystem: Alle
- Diese Registrierdatenbankvariable steuert eine Gruppe von Parametern, mit denen Sie das Verhalten verschiedener Aspekte der DB2-Überwachung ändern können. Trennen Sie jeden Parameter durch ein Semikolon wie im folgenden Beispiel:

  ```
  db2set DB2_SYSTEM_MONITOR_SETTINGS=OLD_CPU_USAGE:TRUE;DISABLE_CPU_USAGE:TRUE
  ```

  Bei jeder Definition von **DB2_SYSTEM_MONITOR_SETTINGS** müssen die einzelnen Parameter explizit angegeben werden. Jeder Parameter, den Sie in dieser Variablen nicht angeben, wird auf seinen Standardwert zurückgesetzt. Betrachten Sie folgendes Beispiel:

  ```
  db2set DB2_SYSTEM_MONITOR_SETTINGS=DISABLE_CPU_USAGE:TRUE
  ```

  Anmerkung: Derzeit verfügt diese Registrierdatenbankvariable nur über Einstellungen für Linux; zusätzliche Einstellungen für andere Betriebssysteme werden zukünftigen Releases hinzugefügt.
  In diesem Fall wird OLD_CPU_USAGE auf die Standardeinstellung zurückgesetzt.
- Parameter:

  OLD_CPU_USAGE
  - Betriebssystem: Linux
  - Werte: TRUE/ON, FALSE/OFF
  - Standardwert unter RHEL4 und SLES9: TRUE (Hinweis: Die Einstellung FALSE für OLD_CPU_USAGE wird ignoriert, nur die alte Funktionsweise wird verwendet.)
  - Standardwert unter RHEL5, SLES10 und anderen: FALSE
  - Dieser Parameter steuert, wie die Instanz CPU-Nutzungszeiten auf Linux-Plattformen abruft. Bei TRUE wird die ältere Methode zum Abruf von CPU-Nutzungszeiten verwendet. Diese Methode gibt CPU-Nutzungszeiten für Systemprozesse und Benutzerprozesse zurück, benötigt dazu jedoch mehr CPU-Zeit (d. h., sie ist mit größerem Systemaufwand verbunden). Bei FALSE wird die neuere Methode zum Abruf der CPU-Belastung verwendet. Diese Methode gibt nur den Wert

der CPU-Belastung durch den Benutzer zurück, ist jedoch schneller, weil sie weniger Systemaufwand verursacht.

DISABLE_CPU_USAGE

- Betriebssystem: Linux
- Werte: TRUE/ON, FALSE/OFF
- Standardwert unter RHEL4 und SLES9: TRUE
- Standardwert unter RHEL5, SLES10 und anderen: FALSE
- Mit diesem Parameter können Sie festlegen, ob die CPU-Belastung gelesen wird oder nicht. Wenn DISABLE_CPU_USAGE aktiviert (TRUE) ist, wird die CPU-Belastung nicht gelesen, sodass Sie den Aufwand umgehen können, der manchmal mit dem Abrufen der CPU-Belastung verbunden ist.

DB2TERRITORY

- Betriebssystem: Alle
- Standardwert: abgeleitet aus der vom Betriebssystem angegebenen Sprachenkennung.
- Diese Variable gibt den Regions- oder Gebietscode der Clientanwendung an, was sich auf die Formate für Datum und Uhrzeit auswirkt.

DB2_VIEW_REOPT_VALUES

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES, NO
- Diese Variable gibt allen Benutzern die Möglichkeit, die im Cache abgelegten Werte einer reoptimierten SQL- oder XQuery-Anweisung in der Tabelle EXPLAIN_PREDICATE zu speichern, wenn die Anweisung mit EXPLAIN bearbeitet wird. Wenn diese Variable auf NO gesetzt ist, kann nur der DBADM diese Werte in der Tabelle EXPLAIN_PREDICATE speichern.

Systemumgebungsvariablen

DB2_ALTERNATE_GROUP_LOOKUP

- Betriebssystem: AIX
- Standardwert: NULL, Werte: NULL oder GETGRSET
- Diese Variable ermöglicht es DB2, Gruppeninformationen aus einer alternativen Quelle abzurufen, die vom Betriebssystem bereitgestellt wird. Unter AIX wird die Funktion getgrset verwendet. Diese Funktion bietet die Möglichkeit, Gruppen aus einer anderen Position als lokalen Dateien über ladbare Authentifizierungsmodule (Loadable Authentication Modules) abzurufen.

DB2_CLP_EDITOR

- Betriebssystem: Alle
- Standardwert: Notepad (Windows), vi (UNIX), Werte: Alle gültigen Editoren, die sich im Betriebssystempfad befinden

  Anmerkung: Diese Registrierdatenbankvariable wird während der Installation nicht auf den Standardwert gesetzt. Stattdessen verwendet der Code, der diese Variable verwendet, einen Standardwert, wenn die Registrierdatenbankvariable nicht festgelegt ist.

- Diese Variable legt den Editor fest, der bei der Ausführung des Befehls EDIT verwendet wird. In einer interaktiven CLP-Sitzung startet der Befehl EDIT einen Editor, der mit einem benutzerdefinierten Befehl vorinstalliert ist, der bearbeitet und ausgeführt werden kann.

DB2_CLP_HISTSIZE

- Betriebssystem: Alle
- Standardwert: 20, Werte: 1 bis 500 einschließlich

  Anmerkung: Diese Registrierdatenbankvariable wird während der Installation nicht auf den Standardwert gesetzt. Stattdessen verwendet der Code, der diese Variable verwendet, den Standardwert 20, wenn die Registrierdatenbankvariable nicht festgelegt ist oder wenn sie auf einen Wert festgelegt wurde, der sich außerhalb des gültigen Bereichs befindet.
- Diese Variable legt die Anzahl der Befehle fest, die im Befehlsprotokoll während interaktiver CLP-Sitzungen gespeichert werden. Da das Befehlsprotokoll im Speicher gehalten wird, kann ein sehr hoher Wert für diese Variable zu Auswirkungen auf die Leistung führen. Diese hängen von der Anzahl und Länge der Befehle ab, die in einer Sitzung ausgeführt werden.

DB2CONNECT_IN_APP_PROCESS

- Betriebssystem: Alle
- Standardwert: YES, Werte: YES oder NO
- Wenn Sie diese Variable auf den Wert NO setzen, werden lokale DB2 Connect-Clients auf einem DB2 Enterprise Server Edition-System gezwungen, innerhalb eines Agenten aktiv zu sein. Einige Vorteile der Ausführung innerhalb eines Agenten bestehen darin, dass lokale Clients auf diese Weise überwacht werden und die SYSPLEX-Unterstützung nutzen können.

DB2_COPY_NAME

- Betriebssystem: Windows
- Standardwert: Der Name der Standardkopie von DB2, die auf Ihrem System installiert ist. Werte: Der Name einer Kopie von DB2, die auf Ihrem System installiert ist. Der Name kann maximal 128 Zeichen lang sein.
- Die Variable **DB2_COPY_NAME** speichert den Namen der Kopie von DB2, die momentan verwendet wird. Wenn Sie mehrere DB2-Kopien auf Ihrem System installiert haben, können Sie **DB2_COPY_NAME** nicht zum Wechseln zu einer anderen Kopie von DB2 verwenden. Sie müssen den Befehl *INSTALLPATH*\bin\db2envar.bat ausführen, um die momentan verwendete Kopie zu wechseln.

DB2DBMSADDR

- Betriebssystem: Linux auf x86 und Linux auf zSeries (31 Bit)
- Standardwert: NULL, Werte: virtuelle Adressen aus dem Bereich von 0x09000000 bis 0xB0000000 in Inkrementen von 0x10000
- Die Registrierdatenbankvariable **DB2DBMSADDR** gibt die Standardadresse der Datenbank für gemeinsam genutzten Speicher im Hexadezimalformat an.

  Anmerkung: Eine falsche Adresse kann schwer wiegende Probleme auf dem DB2-System verursachen, die von einem Fehlschlagen des Starts einer DB2-Instanz bis hin zum Fehlschlagen der Verbindung zur Datenbank reichen können. Eine falsche Adresse ist eine Adresse, die einen

Konflikt mit einem Adressbereich verursacht, der bereits verwendet wird oder der für andere Daten oder Programme reserviert ist. Dieses Problem können Sie beheben, indem Sie die Registrierdatenbankvariable **DB2DBMSADDR** mit dem folgenden Befehl auf NULL setzen:

```
db2set DB2DBMSADDR=
```

Diese Variable kann zur Feinabstimmung der Adressraumbelegung von DB2-Prozessen verwendet werden. Diese Variable ändert die Position des gemeinsam genutzten Speichers für die Instanz von der aktuellen Position bei der virtuellen Adresse 0x10000000 in den neuen Wert.

DB2_DIAGPATH

- Betriebssystem: Alle
- Standardwert: Der Standardwert ist das Verzeichnis db2dump der Instanz unter UNIX- und Linux-Betriebssystemen und das Verzeichnis DB2 der Instanz unter Windows-Betriebssystemen.
- Dieser Parameter gilt nur für ODBC- und DB2-CLI-Anwendungen.

  Mit diesem Parameter können Sie einen vollständig qualifizierten Pfad für DB2-Diagnosedaten angeben. Im angegebenen Verzeichnis können abhängig von Ihrer Plattform Speicherauszugsdateien, Trapdateien, eine Fehlerprotokolldatei, eine Benachrichtigungsdatei und eine Alertprotokolldatei gespeichert werden.

  Die Einstellung dieser Umgebungsvariablen hat im Rahmen der jeweiligen Umgebung für ODBC- und CLI-Anwendungen die gleiche Wirkung wie die Einstellung des Konfigurationsparameters **diagpath** des DB2-Datenbankmanagers und wie Einstellung des CLI/ODBC-Konfigurationsschlüsselworts **DiagPath**.

DB2DOMAINLIST

- Betriebssystem: Alle
- Standardwert: NULL, Werte: Eine Liste mit Windows-Domänennamen, die durch Kommata (',') getrennt sind
- Diese Variable definiert eine oder mehrere Windows-Domänen. Die Liste, die auf dem Server verwaltet wird, definiert die Domänen, in denen die anfordernde Benutzer-ID authentifiziert wird. Nur CONNECT- oder ATTACH-Verbindungsanforderungen von Benutzern, die zu diesen Domänen gehören, werden akzeptiert.

  Diese Variable ist nur wirksam, wenn in der Datenbankmanagerkonfiguration der Authentifizierungstyp CLIENT definiert ist. Sie wird benötigt, wenn die Funktion zur einmaligen Anmeldung (Single Sign-on) über einen Windows-Desktop in einer Windows-Domänenumgebung erforderlich ist.

  DB2-Server der Version 7.1 oder späterer Versionen unterstützen **DB2DOMAINLIST**, jedoch nur in einer reinen Windows-Domänenumgebung. Ab Version 8 Fixpack 15 und Version 9.1 Fixpack 3 wird **DB2DOMAINLIST** unterstützt, wenn entweder der Client oder der Server in einer Windows-Umgebung ausgeführt wird.

DB2ENVLIST

- Betriebssystem: UNIX
- Standardwert: NULL
- Diese Variable listet spezifische Variablennamen für gespeicherte Prozeduren oder für benutzerdefinierte Funktionen auf. Standardmäßig filtert der Befehl db2start alle Benutzerumgebungsvariablen außer denjenigen heraus, die das Präfix **DB2** oder **db2** haben. Wenn bestimmte

Umgebungsvariablen entweder an gespeicherte Prozeduren oder an benutzerdefinierte Funktionen übergeben werden müssen, können Sie die Variablennamen in der Umgebungsvariablen **DB2ENVLIST** auflisten. Trennen Sie die einzelnen Variablennamen durch ein oder mehrere Leerzeichen.

DB2INSTANCE

- Betriebssystem: Alle
- Standardwert: **DB2INSTDEF** unter 32-Bit-Windows-Betriebssystemen
- Diese Umgebungsvariable gibt die Instanz an, die standardmäßig aktiv ist. Unter UNIX müssen Benutzer einen Wert für **DB2INSTANCE** angeben.

DB2INSTPROF

- Betriebssystem: Windows
- Standardwert: Dokumente und Einstellungen\All Users\ Anwendungsdaten\IBM\DB2*Name der Kopie* (Windows XP, Windows 2003), ProgramData\IBM\DB2*Name der Kopie* (Windows Vista)
- Diese Umgebungsvariable gibt die Position des Instanzverzeichnisses unter Windows-Betriebssystemen an. Ab Version 9.5 kann sich das Instanzverzeichnis (und andere Benutzerdatendateien) nicht unter dem Verzeichnis sqllib befinden.

DB2LDAPSecurityConfig

- Betriebssystem: Alle
- Standardwert: NULL, Werte: gültiger Name und Pfad der Konfigurationsdatei für IBM LDAP-Sicherheits-Plug-ins
- Diese Variable dient zur Angabe der Position der Konfigurationsdatei für IBM LDAP-Sicherheits-Plug-ins. Wenn diese Variable nicht definiert ist, hat die Konfigurationsdatei für IBM LDAP-Sicherheits-Plug-ins den Namen IBMLDAPSecurity.ini und befindet sich an einer der folgenden Positionen:
  - Unter Linux- und UNIX-Betriebssystemen: *INSTHOME*/sqllib/cfg/
  - Unter Windows-Betriebssystemen: %**DB2PATH**%\cfg\

  Unter Windows-Betriebssystemen sollte diese Variable in der globalen Systemumgebung definiert werden, um sicherzustellen, dass sie vom DB2-Service berücksichtigt wird.

DB2LIBPATH

- Betriebssystem: UNIX
- Standardwert: NULL
- DB2 bildet einen eigenen gemeinsamen Bibliothekspfad. Wenn Sie dem Bibliothekspfad der Steuerkomponente einen Pfad (PATH) hinzufügen möchten (z. B. erfordert eine benutzerdefinierte Funktion unter AIX einen bestimmten Eintrag in **LIBPATH**), müssen Sie die Variable **DB2LIBPATH** definieren. Der tatsächliche Wert der Variablen **DB2LIBPATH** wird an das Ende des von DB2 gebildeten gemeinsam genutzten Bibliothekspfad angehängt.

DB2LOGINRESTRICTIONS

- Betriebssystem: AIX
- Standardwert: LOCAL, Werte: LOCAL, REMOTE, SU, NONE
- Durch diese Registrierdatenbankvariable können Sie eine AIX-Betriebssystems-API namens loginrestrictions() verwenden. Diese API bestimmt,

ob ein Benutzer für den Zugriff auf das System berechtigt ist. Durch Aufrufen dieser API kann die DB2-Datenbanksicherheitsfunktion die Anmeldeeinschränkungen umsetzen, die durch das Betriebssystem definiert sind. Es gibt unterschiedliche Werte, die an diese API übergeben werden können, wenn diese Registrierdatenbankvariable verwendet wird. Diese Werte sind:

– REMOTE

  DB2 setzt Anmeldeeinschränkungen nur um, um zu überprüfen, ob das Konto für ferne Anmeldevorgänge über die Programme *rlogind* bzw. *telnetd* verwendet werden kann.

– SU

  DB2 Version 9.1 setzt SU-Einschränkungen nur um, um zu überprüfen, ob der Befehl su erlaubt ist und ob der aktuelle Prozess über eine Gruppen-ID verfügt, die den Befehl su aufrufen kann, um zu dem Konto umzuschalten.

– NONE

  DB2 setzt keinerlei Anmeldeeinschränkungen um.

– LOCAL (oder die Variable ist nicht definiert)

  DB2 setzt Anmeldeeinschränkungen nur um, um zu überprüfen, ob für dieses Konto lokale Anmeldungen erlaubt sind. Dies ist das normale Verhalten bei einer Anmeldung.

Unabhängig davon, welche dieser Optionen Sie definieren, können Benutzerkonten bzw. -IDs, die über die angegebenen Zugriffsrechte verfügen, DB2 sowohl lokal auf dem Server als auch über ferne Clients erfolgreich verwenden. Eine Beschreibung der API loginrestrictions() finden Sie in der AIX-Dokumentation.

DB2NODE

- Betriebssystem: Alle
- Standardwert: NULL, Werte: 1 bis 999
- Dient zur Angabe des logischen Zielknotens eines Datenbankpartitionsservers, zu dem eine Verbindung über CONNECT oder ATTACH hergestellt werden soll. Wenn diese Variable nicht definiert wird, wird als logischer Zielknoten standardmäßig der logische Knoten angenommen, der auf der Maschine mit Port `0` definiert ist. In einer Umgebung mit partitionierten Datenbanken können sich die Verbindungseinstellungen auf die Herstellung gesicherter Verbindungen auswirken. Wenn die Variable **DB2NODE** zum Beispiel auf einen Knoten gesetzt wird, sodass die Herstellung einer Verbindung auf diesem Knoten den Weg über einen Zwischenknoten (Hop-Knoten) erfordert, wird die IP-Adresse dieses Zwischenknotens und das Kommunikationsprotokoll, das zur Kommunikation zwischen dem Zwischenknoten und dem Verbindungsknoten verwendet wird, bei der Bewertung dieser Verbindung geprüft, um festzustellen, ob sie als gesicherte Verbindung markiert werden kann oder nicht. Das bedeutet, dass nicht der ursprüngliche Knoten betrachtet wird, von dem aus die Verbindung eingeleitet wurde, sondern der Zwischenknoten.

DB2OPTIONS

- Betriebssystem: Alle
- Standardwert: NULL
- Dient zum Festlegen der Optionen für den Befehlszeilenprozessor.

DB2_PARALLEL_IO

- Betriebssystem: Alle
- Standardwert: NULL, Werte: *tabellenbereichs-id*:[*n*],... – eine durch Kommata getrennte Liste definierter Tabellenbereiche (angegeben durch ihre numerische Tabellenbereichs-ID). Wenn PREFETCHSIZE eines Tabellenbereichs auf AUTOMATIC gesetzt ist, können Sie dem DB2-Datenbankmanager die Anzahl Platten pro Container für diesen Tabellenbereich mitteilen, indem Sie die Tabellenbereichs-ID, gefolgt von einem Doppelpunkt, gefolgt von der Anzahl Platten pro Container *n* angeben. Wird *n* nicht angegeben, ist der Standardwert 6.

  Sie können *tabellenbereichs-id* durch einen Stern (*) ersetzen, um alle Tabellenbereiche anzugeben. Wenn Sie zum Beispiel **DB2_PARALLEL_IO** =* angeben, verwenden alle Tabellenbereiche 6 als Anzahl von Platten pro Container. Wenn Sie sowohl einen Stern als auch eine Tabellenbereichs-ID angeben, hat die Einstellung der Tabellenbereichs-ID Vorrang. Beispiel: Bei der Angabe **DB2_PARALLEL_IO** =*,1:3 verwenden alle Tabellenbereiche 6 als Anzahl der Platten pro Container mit Ausnahme des Tabellenbereichs 1, der 3 Platten pro Container verwendet.
- Diese Registrierdatenbankvariable dient zur Änderung der Art und Weise, wie DB2 die E/A-Parallelität eines Tabellenbereichs berechnet. Wenn die E/A-Parallelität aktiviert ist (entweder implizit durch die Verwendung mehrerer Container oder explizit durch die Einstellung der Variablen **DB2_PARALLEL_IO**), wird sie umgesetzt, indem die richtige Anzahl von Vorablesezugriffsanforderungen abgesetzt wird. Jede Vorablesezugriffsanforderung ist eine Anforderung für einen EXTENTSIZE großen Speicherbereich von Seiten. Betrachten Sie zum Beispiel den Fall, dass ein Tabellenbereich zwei Container hat und der Wert für PREFETCHSIZE das Vierfache des Werts für EXTENTSIZE beträgt. Wenn die Registrierdatenbankvariable definiert ist, wird eine Vorableseanforderung für diesen Tabellenbereich in vier Anforderungen zerlegt (ein EXTENTSIZE großer Speicherbereich pro Anforderung), dies bietet die Möglichkeit, dass vier Vorablesefunktionen die Anforderungen parallel bedienen.

  Es kann sinnvoll sein, die Registrierdatenbankvariable zu definieren, wenn die einzelnen Container in dem Tabellenbereich über mehrere physische Datenträger einheitenübergreifend verteilt gespeichert werden (Stripe-Set) oder wenn der Container in einem Tabellenbereich auf einer einzelnen RAID-Einheit erstellt wird, die aus mehr als einer physischen Platte besteht.

  Wenn diese Registrierdatenbankvariable nicht definiert ist, entspricht der Grad der Parallelität eines Tabellenbereichs der Anzahl der Container des Tabellenbereichs. Wenn **DB2_PARALLEL_IO** zum Beispiel auf den Wert NULL gesetzt wird und ein Tabellenbereich über vier Container verfügt, werden vier Vorablesezugriffsanforderungen in der Größe von EXTENTSIZE abgesetzt oder wenn ein Tabellenbereich zwei Container hat und der Wert für PREFETCHSIZE das Vierfache des Werts für EXTENTSIZE beträgt, wird eine Vorableseanforderung für diesen Tabellenbereich in zwei Anforderungen zerlegt (jeweils eine Anforderung für zwei EXTENTSIZE große Speicherbereiche).

  Wenn diese Registrierdatenbankvariable definiert ist und PREFETCHSIZE für die Tabelle nicht auf AUTOMATIC gesetzt ist, ist der Grad der Parallelität des Tabellenbereichs gleich dem Wert für PREFETCHSIZE dividiert durch den Wert von EXTENTSIZE. Wenn **DB2_PARALLEL_IO** zum Beispiel für einen Tabellenbereich definiert ist, dessen PREFETCH-

SIZE-Wert 160 und dessen EXTENTSIZE-Wert 32 Seiten beträgt, werden fünf EXTENTSIZE große Vorablesezugriffsanforderungen abgesetzt.

Wenn diese Registrierdatenbankvariable definiert ist und PREFETCHSIZE des Tabellenbereichs auf den Wert AUTOMATIC gesetzt ist, berechnet DB2 die Größe des Vorablesezugriffs des Tabellenbereichs anhand folgender Gleichung automatisch:

```
PREFETCHSIZE =
      (Anzahl Container) * (Anzahl Platten pro Container)
         * EXTENTSIZE
```

Die Zahl nach dem Doppelpunkt wird von DB2 als *Anzahl Platten pro Container* in der Gleichung verwendet. Wenn nur ein Stern und keine Zahl angegeben wird, werden standardmäßig 6 Platten pro Container verwendet.

In der folgenden Tabelle sind die verschiedenen verfügbaren Optionen sowie die Berechnung der Parallelität für die einzelnen Fälle zusammengefasst:

Tabelle 64. Berechnung der Parallelität

PREFETCHSIZE des Tabellenbereichs	**DB2_PARALLEL_IO-Einstellung**	**Die Parallelität ist gleich:**
AUTOMATIC (PREFETCHSIZE = Anzahl Container * 1 * EXTENTSIZE)	Nicht definiert	Anzahl Container
AUTOMATIC (PREFETCHSIZE = Anzahl Container * 6 * EXTENTSIZE)	*Tabellenbereichs-ID*	Anzahl Container * 6
AUTOMATIC (PREFETCHSIZE = Anzahl Container * *n* * EXTENTSIZE)	*Tabellenbereichs-ID*:*n*	Anzahl Container * *n*
Nicht AUTOMATIC	Nicht definiert	Anzahl Container
Nicht AUTOMATIC	*Tabellenbereichs-ID*	PREFETCHSIZE / EXTENTSIZE
Nicht AUTOMATIC	*Tabellenbereichs-ID*:*n*	PREFETCHSIZE / EXTENTSIZE

Nehmen Sie zum Beispiel an, Sie haben drei Tabellenbereiche, deren IDs 3, 4 und 5 lauten. Der EXTENTSIZE-Wert für alle Tabellenbereiche beträgt 4096 Byte, und alle Tabellenbereiche haben je zwei Container. Der PREFETCHSIZE-Wert der Tabellenbereiche 3 und 4 ist für beide auf AUTOMATIC gesetzt, der von Tabellenbereich 5 auf 16384 Byte. Wenn Sie nun **DB2_PARALLEL_IO**=*:5,4:10 angeben, leitet sich der Grad der Parallelität für die Tabellenbereiche wie folgt ab:

- Tabellenbereich 3: *n* (Anzahl Platten pro Container) hat den Wert 5, EXTENTSIZE den Wert 4096, die Anzahl Container ist 2 und PREFETCHSIZE hat den Wert AUTOMATIC. Dies führt zu den Gleichungen PREFETCHSIZE = 2*5*4096 und Grad der Parallelität = Anzahl Container * *n* = 2 * 5 = 10.
- Tabellenbereich 4: Beachten Sie, dass für diesen Tabellenbereich *n* (Anzahl Platten pro Container) explizit auf den Wert 10 gesetzt ist. Es gilt: EXTENTSIZE = 4096, Anzahl Container = 2, *n* = 10 und PREFETCHSIZE = AUTOMATIC. Dies führt zu den Gleichungen PREFETCHSIZE = 2*10*4096 und Grad der Parallelität = Anzahl Container * *n* = 2*10 = 20.

– Tabellenbereich 5: *n* hat weiterhin den Wert 5, jedoch ist dies nicht von Belang, da PREFETCHSIZE nicht den Wert AUTOMATIC hat. Es gilt: EXTENTSIZE = 4096, Anzahl Container = 2 und PREFETCHSIZE = 16384. Daraus ergibt sich die Gleichung: Grad der Parallelität = PREFETCHSIZE/EXTENTSIZE = 16384/4096 = 4.

In einigen Szenarios kann die Verwendung dieser Variablen zu einer Konkurrenzsituation für Datenträger führen. Wenn z. B. ein Tabellenbereich zwei Container hat und für jeden der beiden Container jeweils ein einzelner dedizierter Datenträger vorhanden wäre, kann eine Definition der Registrierdatenbankvariablen zu Konkurrenzsituationen beim Zugriff auf diese Datenträger führen, da jeweils zwei Vorableseefunktionen versuchen würden, gleichzeitig auf jeden der beiden Datenträger zuzugreifen. Wenn hingegen jeder der beiden Container über mehrere Datenträger einheitenübergreifend gespeichert wäre, würde eine Definition der Registrierdatenbankvariablen potenziell einen gleichzeitigen Zugriff auf vier verschiedene Datenträger ermöglichen.

Zur Aktivierung von Änderungen an dieser Registrierdatenbankvariablen führen Sie den Befehl db2stop und anschließend den Befehl db2start aus.

DB2PATH

- Betriebssystem: Windows
- Standardwert: je nach Betriebssystem unterschiedlich
- Diese Umgebungsvariable wird zur Angabe des Verzeichnisses verwendet, in dem das Produkt auf 32-Bit-Windows-Betriebssystemen installiert ist.

DB2PROCESSORS

- Betriebssystem: Windows
- Standardwert: NULL, Werte: 0–*n*-1 (mit *n* = Anzahl Prozessoren)
- Diese Variable definiert die Prozessaffinitätsmaske für einen bestimmten Prozess db2syscs. In Umgebungen mit mehreren logischen Knoten wird diese Variable verwendet, um einem Prozessor bzw. einer Gruppe von Prozessoren einen logischen Knoten zuzuordnen.

  Wird diese Variable angegeben, ruft DB2 die API SetProcessAffinityMask() auf. Wird diese Variable nicht angegeben, wird der Prozess db2syscs allen Prozessoren auf dem Server zugeordnet.

DB2RCMD_LEGACY_MODE

- Betriebssystem: Windows
- Standardwert: NULL, Werte: YES, ON, TRUE bzw. 1 oder NO, OFF, FALSE bzw. 0
- Diese Variable gibt Benutzern die Möglichkeit, die erweiterten Sicherheitsmerkmale für den DB2 Remote Command Service zu aktivieren oder zu inaktivieren. Zur Ausführung des DB2 Remote Command Service in einem sicheren Modus setzen Sie die Variable **DB2RCMD_LEGACY_MODE** auf NO, OFF, FALSE, 0 oder NULL. Zur Ausführung im traditionellen Modus (ohne erweiterte Sicherheit) setzen Sie die Variable **DB2RCMD_LEGACY_MODE** auf YES, ON, TRUE oder 1. Der sichere Modus steht nur zur Verfügung, wenn Ihr Domänencontroller mit Windows 2000 oder einer späteren Version des Betriebssystems ausgeführt wird.

Anmerkung: Wenn die Variable **DB2RCMD_LEGACY_MODE** auf YES, ON, TRUE oder 1 gesetzt ist, werden alle Anforderungen, die an den DB2 Remote Command Service gesendet werden, unter dem Kontext des Anforderers verarbeitet. Zu diesem Zweck müssen Sie zulassen, dass die Maschine oder das Serviceanmeldekonto (oder beide) die Identität des Clients annehmen, indem Sie das Maschinenkonto und das Serviceanmeldekonto auf dem Domänencontroller aktivieren.

Anmerkung: Wenn die Variable **DB2RCMD_LEGACY_MODE** auf NO, OFF, FALSE oder 0 gesetzt ist, müssen Sie über die Berechtigung SYSADM verfügen, damit der DB2 Remote Command Service Befehle für Sie ausführen kann.

DB2SYSTEM

- Betriebssystem: Windows und UNIX
- Standardwert: NULL
- Gibt den Namen an, der von Ihren Benutzern und Datenbankadministratoren zur Identifizierung des DB2-Serversystems verwendet wird. Dieser Name sollte nach Möglichkeit innerhalb Ihres Netzwerks eindeutig sein.

  Dieser Name wird auf der Systemebene der Objektbaumstruktur in der Steuerzentrale angezeigt, um Administratoren bei der Identifizierung von Server-Systemen zu helfen, die von der Steuerzentrale aus verwaltet werden können.

  Bei der Verwendung der Funktion Netzwerk durchsuchen des Konfigurationsassistenten gibt DB2-Discovery diesen Namen zurück, und er wird auf der Systemebene der resultierenden Baumstruktur angezeigt. Dieser Name unterstützt Benutzer bei der Identifizierung des Systems, das die Datenbank enthält, auf die Sie zugreifen wollen. Bei der Installation wird für **DB2SYSTEM** wie folgt ein Wert festgelegt:
  - Unter Windows setzt das Installationsprogramm diesen Parameter auf den Wert des Computernamens, der für das Windows-System angegeben ist.
  - Auf UNIX-Systemen wird für diesen Parameter der TCP/IP-Hostname des UNIX-Systems definiert.

DB2_UPDDBCFG_SINGLE_DBPARTITION

- Betriebssystem: Alle
- Standardwert: Nicht definiert, Werte: 0/FALSE/NO, 1/TRUE/YES
- Durch Setzen dieser Registrierdatenbankvariablen auf 1, TRUE oder YES können Sie angeben, dass alle Aktualisierungen und Zurücksetzungen an Ihrer Datenbank nur eine bestimmte Partition betreffen. Wenn die Variable nicht definiert ist, erfolgen Aktualisierungen und Zurücksetzungen in der Funktionsweise von Version 9.5.
- Ab Version 9.5 betreffen Aktualisierungen bzw. Änderungen an einer Datenbankkonfiguration alle Datenbankpartitionen, wenn Sie keine Partitionsklausel angeben. Mithilfe der Variablen **DB2_UPDDBCFG_SINGLE_DBPARTITION** können Sie zum Verhalten früherer Versionen von DB2 zurückkehren, bei dem Aktualisierungen an einer Datenbankkonfiguration nur für die lokale Datenbankpartition bzw. für die durch die Registrierdatenbankvariable **DB2NODE** angegebene Datenbankpartition gelten. Dies ermöglicht Abwärtskompatibilität zur Unterstützung vorhandener Befehlsscripts oder Anwendungen, die dieses Verhalten voraussetzen.

Anmerkung: Diese Variable bezieht sich nicht auf Aktualisierungs- oder Rücksetzanforderungen, die durch Aufrufen von ADMIN_CMD-Routinen erfolgen.

DB2_USE_PAGE_CONTAINER_TAG

- Betriebssystem: Alle
- Standardwert: NULL, Werte: ON, NULL
- Standardmäßig speichert DB2 eine Containerkennung im ersten Speicherbereich jedes DMS-Containers. Dabei spielt es keine Rolle, ob es sich um eine Datei oder eine Einheit handelt. Die Containerkennung bildet die Metadaten für den Container. Vor DB2 Version 8.1 wurde die Containerkennung auf einer einzigen Seite gespeichert und erforderte so weniger Speicherplatz im Container. Wenn die Containerkennung weiterhin in einer einzigen Seite gespeichert werden soll, setzen Sie die Variable **DB2_USE_PAGE_CONTAINER_TAG** auf den Wert ON.

  Wenn Sie jedoch diese Registrierdatenbankvariable auf ON setzen, wenn Sie RAID-Einheiten als Container verwenden, kann sich die E/A-Leistung verschlechtern. Da Sie für RAID-Einheiten Tabellenbereiche mit einem EXTENTSIZE-Wert erstellen, der der Stripegröße oder einem Vielfachen der Stripegröße der RAID-Einheiten entspricht, führt die Einstellung der Variablen **DB2_USE_PAGE_CONTAINER_TAG** auf ON dazu, dass sich die EXTENTSIZE großen Speicherbereiche nicht an den RAID-Stripes (einheitenübergreifend gespeicherten Datenblöcken) ausrichten. Infolgedessen muss eine E/A-Anforderung eventuell auf mehr physische Platten zugreifen, als es optimal der Fall wäre. Benutzern wird ausdrücklich davon abgeraten, diese Registrierdatenbankvariable zu aktivieren, sofern Sie nicht unter sehr eingeschränkten Speicherverhältnissen arbeiten oder eine Funktionsweise benötigen, die mit der früherer Versionen (vor Version 8) übereinstimmt..

  Zur Aktivierung von Änderungen an dieser Registrierdatenbankvariablen führen Sie den Befehl db2stop und anschließend den Befehl db2start aus.

DB2_WORKLOAD

- Betriebssystem: Alle
- Standardwert: Nicht definiert, Werte: 1C, CM, SAP, TPM, WC
- Jeder Wert für die Variable **DB2_WORKLOAD** stellt eine bestimmte Gruppierung verschiedener Registrierdatenbankvariablen mit vordefinierten Einstellungen dar.
- Die folgenden Werte sind gültig:

  1C Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für 1C-Anwendungen konfigurieren wollen.

  CM Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für IBM Content Manager konfigurieren wollen. Dieser Wert ist in DB2 Version 9.5 Fixpack 3 (und späteren Versionen) verfügbar.

  SAP Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für die SAP-Umgebung konfigurieren wollen.

  Wenn Sie **DB2_WORKLOAD**=SAP festgelegt haben, werden der Benutzertabellenbereich SYSTOOLSPACE und der Tabellenbereich für temporäre Benutzertabellen SYSTOOLSTMPSPACE nicht automatisch erstellt. Diese Tabellenbereiche werden für

Tabellen verwendet, die von den folgenden Assistenten, Dienstprogrammen und Funktionen automatisch erstellt werden:
- Automatische Verwaltung
- Designadvisor
- Anzeige der Datenbankinformationen in der Steuerzentrale
- Gespeicherte Prozedur SYSINSTALLOBJECTS, wenn der Eingabeparameter für Tabellenbereich nicht angegeben wird
- Gespeicherte Prozedur GET_DBSIZE_INFO

Ohne die Tabellenbereiche SYSTOOLSPACE und SYSTOOLSTMPSPACE können Sie diese Assistenten, Dienstprogramme und Funktionen nicht nutzen.

Wenn Sie diese Assistenten, Dienstprogramme oder Funktionen verwenden möchten, führen Sie eine der folgenden Aktionen aus:
- Erstellen Sie den Tabellenbereich SYSTOOLSPACE manuell, sodass er die Objekte aufnehmen kann, die von den Tools benötigt werden. (In einer Umgebung mit partitionierten Datenbanken erstellen Sie diesen Tabellenbereich in der Katalogpartition.) Beispiel:

  ```
  CREATE REGULAR TABLESPACE SYSTOOLSPACE
  IN IBMCATGROUP
  MANAGED BY SYSTEM
  USING ('SYSTOOLSPACE')
  ```
- Rufen Sie unter Angabe eines gültigen Tabellenbereichs die gespeicherte Prozedur SYSINSTALLOBJECTS auf, um die Objekte für die Tools zu erstellen, und geben Sie den Bezeichner für das bestimmte Tool an. SYSINSTALLOBJECTS erstellt einen Tabellenbereich für Sie. Wenn Sie den Tabellenbereich SYSTOOLSSPACE nicht für die Objekte verwenden wollen, geben Sie einen anderen benutzerdefinierten Tabellenbereich an.

Nachdem Sie mindestens eine dieser Optionen ausgeführt haben, erstellen Sie den Tabellenbereich SYSTOOLSTMPSPACE für temporäre Tabellen (ebenfalls in der Katalogpartition, wenn Sie in einer Umgebung mit partitionierten Datenbanken arbeiten). Beispiel:

```
CREATE USER TEMPORARY TABLESPACE SYSTOOLSTMPSPACE
IN IBMCATGROUP
MANAGED BY SYSTEM
USING ('SYSTOOLSTMPSPACE')
```

Wenn der Tabellenbereich SYSTOOLSPACE und der temporäre Tabellenbereich SYSTOOLSTMPSPACE erstellt sind, können Sie die oben genannten Assistenten, Dienstprogramme bzw. Funktionen verwenden.

TPM Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für IBM Tivoli Provisioning Manager konfigurieren wollen.

WC Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für Websphere Commerce konfigurieren wollen. Dieser Wert ist in DB2 Version 9.5 Fixpack 4 (und späteren Versionen) verfügbar.

Kommunikationsvariablen

DB2CHECKCLIENTINTERVAL

- Betriebssystem: Alle, nur Server
- Standardwert=50, Werte: ein numerischer Wert größer-gleich null
- Diese Variable gibt die Häufigkeit von TCP/IP-Clientverbindungsprüfungen an. Sie ermöglicht das frühe Erkennen einer Clientbeendigung, ohne den Abschluss der Abfrage abzuwarten. Wird diese Variable auf 0 gesetzt, wird keine Prüfung durchgeführt.

  Niedrigere Werte bedeuten häufigere Überprüfungen. Verwenden Sie 100 als Richtwert für geringe Häufigkeit, 50 für mittlere Häufigkeit und 10 für hohe Häufigkeit. Der Wert wird in einer internen DB2-Metrik gemessen. Die Werte stellen eine lineare Skala dar. Dies bedeutet, dass eine Erhöhung des Werts von 50 auf 100 die Länge des Intervalls verdoppelt. Je häufiger Sie den Clientstatus während der Ausführung einer Datenbankanforderung prüfen, desto länger dauert die Ausführung von Abfragen. Wenn die DB2-Auslastung hoch ist (d. h., wenn viele interne Anforderungen verarbeitet werden), hat die Einstellung von **DB2CHECKCLIENTINTERVAL** auf einen niedrigen Wert eine größere Auswirkung auf die Leistung als in einer Situation, in der die Auslastung gering ist.

  Seit DB2 Universal Database Version 8.1.4 hat **DB2CHECKCLIENTINTERVAL** den Standardwert 50. Vor Version 8.1.4 war der Standardwert 0.

DB2COMM

- Betriebssystem: Alle, nur Server
- Standardwert=NULL, Werte: NPIPE, TCPIP, SSL
- Diese Variable gibt die Kommunikationsmanager an, die gestartet werden, wenn der Datenbankmanager gestartet wird. Wenn diese Variable nicht definiert wird, werden auf dem Server keine DB2-Kommunikationsmanager gestartet.

DB2FCMCOMM

- Betriebssystem: Alle unterstützten DB2 Enterprise Server Edition-Plattformen
- Standardwert=TCPIP4, Werte: TCPIP4 oder TCPIP6
- Diese Variable gibt an, wie die Hostnamen in der Datei db2nodes.cfg aufgelöst werden sollen. Alle Hostnamen werden auf der Basis von IPv4 oder IPv6 aufgelöst. Wenn in der Datei db2nodes.cfg anstelle eines Hostnamens eine IP-Adresse angegeben wird, dann bestimmt das Format dieser IP-Adresse, ob IPv4 oder IPv6 verwendet wird. Wenn **DB2FCMCOMM** nicht definiert ist, dann können entsprechend der Standardeinstellung IPv4 nur IPv4-Hosts gestartet werden.

  Anmerkung: Wenn das IP-Format, das auf der Basis des in db2nodes.cfg angegebenen Hostnamens aufgelöst wurde, oder das direkt in db2nodes.cfg angegebene IP-Format nicht mit der Einstellung von **DB2FCMCOMM** übereinstimmt, schlägt die Ausführung von db2start fehl.

DB2_FORCE_NLS_CACHE

- Betriebssystem: AIX, HP_UX, Solaris
- Standardwert=FALSE, Werte: TRUE oder FALSE

- Diese Variable dient zur Verhinderung der Möglichkeit von Sperrkonflikten in Multithread-Anwendungen. Wenn diese Registrierdatenbankvariable den Wert TRUE hat, werden die Informationen zur Codepage und zum Gebietscode beim Erstzugriff eines Threads auf die Informationen im Cache gespeichert. Von da an werden die im Cache gespeicherten Informationen für jeden anderen Thread verwendet, der diese Informationen anfordert. Dadurch wird der Sperrkonflikt beseitigt, was in bestimmten Situationen zu einem Leistungsvorteil führt. Diese Einstellung sollte nicht verwendet werden, wenn die Anwendung länderspezifische Angaben (Locale-Einstellungen) zwischen Verbindungen ändert. In einer solchen Situation ist sie wahrscheinlich kaum erforderlich, da Multithread-Anwendungen ihre länderspezifischen Einstellungen in der Regel nicht ändern, weil dies nicht *threadsicher* ist.

DB2RSHCMD

- Betriebssystem: UNIX
- Standardwert=rsh (remsh unter HP-UX), Werte sind vollständige Pfadnamen für rsh, remsh oder ssh
- Standardmäßig verwendet das DB2-Datenbanksystem beim Starten ferner Datenbankpartitionen 'rsh' als Kommunikationsprotokoll. Das Script db2_all wird verwendet, um Dienstprogramme und Befehle für alle Datenbankpartitionen auszuführen. Das Setzen dieser Registrierdatenbankvariablen auf den vollständigen Pfadnamen für ssh veranlasst DB2-Datenbankprodukte zum Beispiel zur Verwendung von ssh als Kommunikationsprotokoll für die angeforderte Ausführung der Dienstprogramme und Befehle. Sie kann auch auf den vollständigen Pfadnamen eines Scripts gesetzt werden, das das ferne Befehlsprogramm mit entsprechenden Standardparametern aufruft. Diese Variable ist nur für partitionierte Datenbanken oder für Umgebungen mit einer einzigen Partition erforderlich, wenn der Befehl db2start auf einem Server ausgeführt wird, auf dem das DB2-Produkt nicht installiert ist. Der Instanzeigner muss das angegebene Programm der fernen Shell verwenden können, um sich von jedem DB2-Datenbankknoten aus bei jedem anderen DB2-Datenbankknoten anzumelden, ohne dass eine weitere Prüfung oder Authentifizierung (d. h. Kennwörter oder Kennwortphrasen) erforderlich wird.

Detaillierte Anweisungen zur Einstellung der Registrierungsdatenbankvariablen DB2RSHCMD zur Verwendung einer SSH-Shell mit DB2 finden Sie im White Paper "Configure DB2 Universal Database for UNIX to use OpenSSH" (DB2 Universal Database für UNIX zur Verwendung von OpenSSH konfigurieren).

DB2RSHTIMEOUT

- Betriebssystem: UNIX
- Standardwert=30 Sekunden, Werte: 1 - 120
- Diese Variable ist nur gültig, wenn **DB2RSHCMD** auf einen Wert ungleich NULL gesetzt ist. Diese Registrierdatenbankvariable wird für die Steuerung des Zeitlimitintervalls verwendet, während dessen das DB2-Datenbanksystem auf einen fernen Befehl wartet. Wenn bis zum Ablauf dieses Zeitlimits keine Antwort empfangen wurde, wird angenommen, dass die ferne Datenbankpartition nicht erreichbar und die Operation fehlgeschlagen ist.

Anmerkung: Bei dem angegebenen Zeitwert handelt es sich nicht um die für die Ausführung des fernen Befehls erforderliche Zeit, sondern um die Zeit, die für die Authentifizierung der Anforderung benötigt wird.

DB2SORCVBUF

- Betriebssystem: Alle
- Standardwert=65.536
- Definiert den Wert von TCP/IP-Empfangspuffern.

DB2SOSNDBUF

- Betriebssystem: Alle
- Standardwert=65.536
- Definiert den Wert von TCP/IP-Sendepuffern.

DB2TCP_CLIENT_CONTIMEOUT

- Betriebssystem: Alle, nur Client
- Standardwert=0 (kein Zeitlimit), Werte: 0 - 32.767 Sekunden
- Die Registrierdatenbankvariable **DB2TCP_CLIENT_CONTIMEOUT** gibt die Anzahl Sekunden an, die ein Client auf den Abschluss einer TCP/IP-Verbindungsoperation wartet. Wenn innerhalb der angegebenen Sekunden keine Verbindung hergestellt wird, gibt der DB2-Datenbankmanager den Fehler `-30081 selectForConnectTimeout` zurück.

  Es besteht kein Zeitlimit, wenn die Registrierdatenbankvariable nicht definiert oder auf den Wert 0 gesetzt ist.

  Anmerkung: Auch Betriebssysteme haben einen Wert für ein Verbindungszeitlimit, der früher greifen könnte als das Zeitlimit, das Sie durch **DB2TCP_CLIENT_CONTIMEOUT** definieren. Zum Beispiel hat AIX den Standardwert *tcp_keepinit*=150 (in Halbsekunden), der den Verbindungsaufbau nach 75 Sekunden beenden würde.

DB2TCP_CLIENT_RCVTIMEOUT

- Betriebssystem: Alle, nur Client
- Standardwert=0 (kein Zeitlimit), Werte: 0 - 32.767 Sekunden
- Die Registrierdatenbankvariable **DB2TCP_CLIENT_RCVTIMEOUT** gibt die Anzahl Sekunden an, die ein Client auf den Empfang von Daten durch eine TCP/IP-Operation wartet. Wenn innerhalb der angegebenen Sekunden keine Daten vom Server empfangen werden, gibt der DB2-Datenbankmanager den Fehler `-30081 selectForRecvTimeout` zurück.

  Es besteht kein Zeitlimit, wenn die Registrierdatenbankvariable nicht definiert oder auf den Wert 0 gesetzt ist.

  Anmerkung: Der Wert für DB2TCP_CLIENT_RCVTIMEOUT kann durch die CLI überschrieben werden, indem das Schlüsselwort 'Receive-Timeout' für 'db2cli.ini' oder das Verbindungsattribut SQL_ATTR_RECEIVE_TIMEOUT verwendet wird.

DB2TCPCONNMGRS

- Betriebssystem: Alle
- Standardwert=1 auf seriellen Maschinen; auf symmetrischen Multiprozessormaschinen die aufgerundete Quadratwurzel aus der Anzahl Prozessoren bis zu maximal 16 Verbindungsmanagern. Werte: 1 bis 16.

- Wenn die Registrierdatenbankvariable nicht definiert ist, wird die Standardzahl von Verbindungsmanagern erstellt. Wenn die Registrierdatenbankvariable definiert ist, setzt der zugeordnete Wert den Standardwert außer Kraft. Die Zahl der angegebenen TCP/IP-Verbindungsmanager wird bis zu einem Maximum von 16 erstellt. Wenn weniger als 1 angegeben wird, wird **DB2TCPCONNMGRS** auf den Wert 1 gesetzt und eine Warnung protokolliert, dass der Wert außerhalb des gültigen Bereichs liegt. Wenn ein Wert größer als 16 angegeben wird, wird **DB2TCPCONNMGRS** auf den Wert 16 gesetzt und eine Warnung protokolliert, dass der Wert außerhalb des gültigen Bereichs liegt. Werte zwischen 1 und 16 werden wie angegeben verwendet. Wenn mehr als ein Verbindungsmanager erstellt wird, sollte sich der Verbindungsdurchsatz verbessern, wenn mehrere Clientverbindungen gleichzeitig empfangen werden. Wenn der Benutzer eine SMP-Maschine verwendet oder die Registriervariable **DB2TCPCONNMGRS** geändert hat, können für den TCP/IP-Verbindungsmanager zusätzliche Prozesse (unter UNIX) bzw. Threads (unter Windows-Betriebssystemen) vorhanden sein. Zusätzliche Prozesse oder Threads erfordern zusätzlichen Speicher.

  Anmerkung: Wenn die Anzahl der Verbindungsmanager auf den Wert 1 gesetzt wird, kommt es bei Fernverbindungen in Systemen mit zahlreichen Benutzern und/oder häufigem Verbindungsauf- und -abbau zu einem Leistungsabfall.

Befehlszeilenvariablen

DB2BQTIME

- Betriebssystem: Alle
- Standardwert: 1 Sekunde, Mindestwert: 1 Sekunde
- Diese Variable gibt die Zeitdauer an, die das Front-End des Befehlszeilenprozessors inaktiv bleibt, bevor es prüft, ob der Back-End-Prozess aktiv ist, und eine Verbindung zu diesem Prozess herstellt.

DB2BQTRY

- Betriebssystem: Alle
- Standardwert=60 Wiederholungen, Minimalwert: 0 Wiederholungen
- Diese Variable gibt die Anzahl der Wiederholungen an, mit denen das Front-End des Befehlszeilenprozessors festzustellen versucht, ob der Back-End-Prozess bereits aktiv ist. Diese Variable arbeitet in Verbindung mit der Variablen **DB2BQTIME**.

DB2_CLPPROMPT

- Betriebssystem: Alle
- Standardwert=Keiner (falls kein Standardwert definiert ist, wird „db2 => ” als interaktive Standard-CLP-Eingabeaufforderung verwendet), Werte: Beliebige Textzeichenfolge mit einer Länge von unter 100 Zeichen, die keines oder mehrere der folgenden Token enthält: %i, %d, %ia, %da oder %n. Diese Variable braucht nur dann definiert zu werden, wenn die standardmäßig verwendete interaktive CLP-Eingabeaufforderung (db2 =>) explizit geändert werden soll.
- Mit dieser Registrierdatenbankvariablen können Benutzer die Eingabeaufforderung definieren, die im interaktiven Modus des Befehlszeilenprozessors (CLP) verwendet werden soll. Die Variable kann auf eine beliebige Textzeichenfolge von unter 100 Zeichen gesetzt werden, die keine oder mehrere der optionalen Token %i, %d, %ia, %da oder %n ent-

hält. Im interaktiven CLP-Modus wird die zu verwendende Eingabeaufforderung erstellt, indem die in der Registrierdatenbankvariablen **DB2_CLPPROMPT** angegebene Textzeichenfolge verwendet wird und alle Vorkommen der Token %i, %d, %ia, %da oder %n durch den lokalen Aliasnamen der jeweils verbundenen Instanz, den lokalen Aliasnamen der aktuellen Datenbankverbindung, die Berechtigungs-ID der jeweils verbundenen Instanz, die Berechtigungs-ID der aktuellen Datenbankverbindung bzw. eine neue Zeile (d. h. eine Zeilenschaltung) ersetzt werden.

Anmerkung:

1. Wird die Registrierdatenbankvariable **DB2_CLPPROMPT** innerhalb des interaktiven CLP-Modus geändert, tritt der neue Wert für **DB2_CLPPROMPT** erst in Kraft, nachdem der interaktive CLP-Modus geschlossen und erneut geöffnet wurde.
2. Wenn keine Instanzverbindung vorhanden ist, wird %ia durch eine leere Zeichenfolge und %i durch den Wert der Registrierdatenbankvariablen **DB2INSTANCE** ersetzt. Nur auf Windows-Plattformen: Wenn die Variable **DB2INSTANCE** nicht definiert ist, wird %i durch den Wert der Registrierdatenbankvariablen **DB2INSTDEF** ersetzt. Ist keine dieser Variablen definiert, wird %i durch eine leere Zeichenfolge ersetzt.
3. Wenn keine Datenbankverbindung vorhanden ist, wird %da durch eine leere Zeichenfolge und %d durch den Wert der Registrierdatenbankvariablen **DB2DBDFT** ersetzt. Ist die Variable **DB2DBDFT** nicht definiert, wird %d durch eine leere Zeichenfolge ersetzt.
4. Die Eingabeaufforderung für interaktive Eingabe zeigt die Werte für die Berechtigungs-IDs, Datenbanknamen und Instanznamen stets in Großbuchstaben an.

DB2IQTIME

- Betriebssystem: Alle
- Standardwert=5 Sekunden, Minimalwert: 1 Sekunde
- Diese Variable definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors an der Eingabewarteschlange darauf wartet, dass der Front-End-Prozess Befehle übergibt.

DB2RQTIME

- Betriebssystem: Alle
- Standardwert=5 Sekunden, Minimalwert: 1 Sekunde
- Diese Variable definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors auf eine Anforderung vom Front-End-Prozess wartet.

Variablen für Umgebungen mit partitionierten Datenbanken

DB2CHGPWD_EEE

- Betriebssystem: DB2 ESE unter AIX, Linux und Windows
- Standardwert=NULL, Werte: YES oder NO
- Diese Variable gibt an, ob Sie zulassen, dass andere Benutzer Kennwörter auf ESE-Systemen unter AIX oder Windows ändern. Es muss sichergestellt werden, dass die Kennwörter für alle Datenbankpartitionen oder Knoten mithilfe eines Windows-Domänencontrollers unter Windows oder LDAP unter AIX zentral verwaltet werden. Wenn Kennwörter nicht

zentral verwaltet werden, bleiben sie möglicherweise nicht für alle Datenbankpartitionen bzw. Knoten konsistent. Dies könnte dazu führen, dass ein Kennwort nur in der Datenbankpartition geändert wird, mit der der Benutzer zur Durchführung der Änderung verbunden ist.

DB2_FCM_SETTINGS

- Betriebssystem: Linux
- Standardwert=NULL, Werte: FCM_MAXIMIZE_SET_SIZE:[YES | TRUE | NO | FALSE]. Der Standardwert für FCM_MAXIMIZE_SET_SIZE ist NO.
- Seit Version 9.5 Fixpack 4 können Sie die Registrierdatenbankvariable **DB2_FCM_SETTINGS** mit dem Token FCM_MAXIMIZE_SET_SIZE definieren, um einen Standardspeicherbereich von 2 GB für den FCM-Puffer (FCM, Fast Communication Manager) vorab zuzuordnen. Das Token muss entweder den Wert YES oder den Wert TRUE haben, um diese Funktion zu aktivieren.

DB2_NUM_FAILOVER_NODES

- Betriebssystem: Alle
- Standardwert=2, Werte: 0 bis zur erforderlichen Anzahl der Datenbankpartitionen
- Auf jedem System, auf dem sich eine bestimmte Datenbank befindet, definieren Sie **DB2_NUM_FAILOVER_NODES**, um die Gesamtzahl der Datenbankpartitionen in der betreffenden Datenbank anzugeben.

  In einer DB2-Datenbanklösung mit hoher Verfügbarkeit können beim Ausfall eines Datenbankservers die Datenbankpartitionen auf der Maschine, auf der der Fehler aufgetreten ist, auf einer anderen Maschine erneut gestartet werden. FCM (Fast Communication Manager) verwendet **DB2_NUM_FAILOVER_NODES**, um zu berechnen, wie viel Speicherplatz auf den verschiedenen Maschinen reserviert werden muss, um diese Funktionsübernahme zu ermöglichen.

  Betrachten Sie zum Beispiel die folgende Konfiguration:

  – Maschine A verfügt über zwei Datenbankpartitionen: 1 und 2.
  – Maschine B verfügt über zwei Datenbankpartitionen: 3 und 4.
  – Für **DB2_NUM_FAILOVER_NODES** wird sowohl auf Maschine A als auch auf Maschine B der Wert 4 angegeben.

  Beim Starten von DB2 (DB2START) reserviert FCM genügend Speicherplatz auf A und B, um bis zu vier Datenbankpartitionen zu verwalten, sodass beim Ausfall einer Maschine die beiden Datenbankpartitionen der fehlerhaften Einheit auf der anderen Maschine erneut gestartet werden können. Wenn Maschine A ausfällt, dann können die Datenbankpartitionen 1 und 2 auf Maschine B erneut gestartet werden. Wenn Maschine B ausfällt, dann können die Datenbankpartitionen 3 und 4 auf Maschine A erneut gestartet werden.

DB2_PARTITIONEDLOAD_DEFAULT

- Betriebssystem: Alle unterstützten ESE-Plattformen
- Standardwert=YES, Werte: YES oder NO
- Die Registrierdatenbankvariable **DB2_PARTITIONEDLOAD_DEFAULT** ermöglicht Benutzern, die Standardfunktionsweise des Dienstprogramms LOAD in einer ESE-Umgebung zu ändern, wenn keine ESE-spezifischen LOAD-Optionen angegeben werden. Der Standardwert YES gibt an, dass in einer ESE-Umgebung, wenn Sie keine ESE-spezifischen LOAD-Optio-

nen angeben, das Laden in allen Datenbankpartitionen versucht wird, in denen die Zieltabelle definiert ist. Wenn der Wert NO ist, wird das Laden nur in der Datenbankpartition versucht, mit der das Dienstprogramm LOAD gegenwärtig verbunden ist.

Anmerkung: Diese Variable ist veraltet und wird in einem späteren Release möglicherweise entfernt. Der Befehl LOAD bietet verschiedene Optionen, die zur Erzielung der gleichen Funktionsweise verwendet werden können. Sie können die gleichen Ergebnisse wie mit der Einstellung NO für diese Variable erzielen, indem Sie die folgenden Optionen im Befehl LOAD angeben: `PARTITIONED DB CONFIG MODE LOAD_ONLY OUTPUT_DBPARTNUMS x`. Dabei steht x für die Partitionsnummer der Partition, in die Sie Daten laden wollen.

DB2PORTRANGE

- Betriebssystem: Windows
- Werte: nnnn:nnnn
- Dieser Wert wird auf den TCP/IP-Portbereich gesetzt, der vom FCM verwendet wird, sodass alle zusätzlichen auf einer anderen Maschine erstellten Datenbankpartitionen den gleichen Portbereich haben.

Abfragecompilervariablen

DB2_ANTIJOIN

- Betriebssystem: Alle
- Standardwert=NO in einer ESE-Umgebung, Standardwert=YES in einer anderen Umgebung (nicht ESE), Werte: YES, NO oder EXTEND
- Für DB2 Enterprise Server Edition: Bei Angabe von YES sucht das Optimierungsprogramm nach Möglichkeiten, „NOT EXISTS"-Unterabfragen in Antijoins umzusetzen, die von DB2 effizienter verarbeitet werden können. In anderen Umgebungen (nicht ESE): Bei Angabe von NO schränkt das Optimierungsprogramm die Möglichkeiten zur Umsetzung von „NOT EXISTS"-Unterabfragen in Antijoins ein.

  Wenn EXTEND angegeben ist, sucht das Optimierungsprogramm sowohl in ESE-Umgebungen als auch in anderen Umgebungen nach Möglichkeiten, NOT IN- und NOT EXISTS-Unterabfragen in Antijoins umzusetzen.

DB2_INLIST_TO_NLJN

- Betriebssystem: Alle
- Standardwert=NO, Werte: YES oder NO
- In einigen Fällen kann der SQL- und XQuery-Compiler ein Listenvergleichselement IN in einen Join umschreiben. Betrachten Sie zum Beispiel die folgende Abfrage:

```
SELECT *
 FROM EMPLOYEE
 WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

  Diese Abfrage könnte folgendermaßen geschrieben werden:

```
SELECT *
 FROM EMPLOYEE, (VALUES 'D11', 'D21', 'E21') AS V(DNO)
 WHERE DEPTNO = V.DNO
```

  Diese Überarbeitung könnte eine bessere Leistung erzielen, wenn es einen Index für die Spalte DEPTNO gibt. Auf die Liste von Werten würde zuerst zugegriffen und die Liste mit der Tabelle EMPLOYEE

durch einen Join mit Verschachtelungsschleife mithilfe des Index zur Anwendung des Joinvergleichselements verknüpft.

Manchmal verfügt das Optimierungsprogramm über keine genauen Informationen, um die beste Joinmethode für die umgeschriebene Version der Abfrage zu bestimmen. Dies kann der Fall sein, wenn die IN-Liste Parametermarken oder Hostvariablen enthält, die verhindern, dass das Optimierungsprogramm die Selektivität mithilfe der Katalogstatistiken ermitteln kann. Diese Registrierdatenbankvariable veranlasst das Optimierungsprogramm, Joins mit Verschachtelungsschleifen zu bevorzugen, um die Liste von Werten zu verknüpfen und dabei die Tabelle, die die IN-Liste beisteuert, als innere Tabelle im Join zu verwenden.

Anmerkung: Wenn mindestens eine der beiden DB2-Abfragecompilervariablen **DB2_MINIMIZE_LISTPREFETCH** und **DB2_INLIST_TO_NLJN** auf YES gesetzt ist, bleibt sie auch dann aktiv, wenn REOPT(ONCE) angegeben wird.

DB2_LIKE_VARCHAR

- Betriebssystem: Alle
- Standardwert=Y,Y,
- Steuert die Verwendung von Statistikdaten zu Unterelementen. Dabei handelt es sich um Statistikdaten zum Inhalt von Spaltendaten, wenn diese eine Struktur in Form einer Folge von Unterfeldern oder Unterelementen aufweisen, die durch Leerzeichen getrennt sind. Die Erfassung von Statistikdaten für Unterelemente ist optional und wird durch Optionen im Befehl RUNSTATS oder der API gesteuert.

  Diese Registrierdatenbankvariable beeinflusst, wie das Optimierungsprogramm ein Vergleichselement der folgenden Form behandelt:

  ```
  COLUMN LIKE '%xxxxxx%'
  ```

  Dabei ist xxxxxx eine beliebige Zeichenfolge.

  Die folgende Syntax zeigt, wie diese Registrierdatenbankvariable verwendet wird:

  ```
  db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]
  ```

  Dabei gilt Folgendes:

  – Der Term vor dem Komma bzw. der einzige Term rechts des Vergleichselements hat folgende Bedeutung, allerdings nur, wenn für den zweiten Term der Wert N angegeben wurde oder die Spalte keine positiven Statistikdaten zu Unterelementen hat:
    - S – Das Optimierungsprogramm schätzt die Länge der einzelnen Elemente einer Folge miteinander verknüpfter Elemente, die eine Spalte bilden, ausgehend von der Länge der Zeichenfolge, die von den %-Zeichen eingeschlossen wird.
    - Y – Der Standardwert. Verwenden des Standardwerts 1,9 als Algorithmusparameter. Verwenden des Algorithmus für Unterelemente mit variabler Länge mit dem Algorithmusparameter.
    - N – Verwenden eines Algorithmus für Unterelemente mit fester Länge.
    - num1 – Verwenden des Werts von num1 als Algorithmusparameter für den Algorithmus für Unterelemente mit variabler Länge.

– Der Term nach dem Komma hat folgende Bedeutung, jedoch nur für Spalten, die positive Statistikdaten zu Unterelementen haben:
  - N – Kein Verwenden von Statistikdaten zu Unterelementen. Der erste Term tritt in Kraft.
  - Y – Der Standardwert. Verwenden eines Algorithmus für Unterelemente mit variabler Länge, der Statistikdaten zu Unterelementen zusammen mit dem Standardwert 1,9 als Algorithmusparameter verwendet, wenn Spalten mit positiven Statistikdaten zu Unterelementen vorliegen.
  - num2 – Verwenden eines Algorithmus für Unterelemente mit variabler Länge, der Statistikdaten zu Unterelementen zusammen mit dem Wert num2 als Algorithmusparameter verwendet, wenn Spalten mit positiven Statistikdaten zu Unterelementen vorliegen.

DB2_MINIMIZE_LISTPREFETCH

- Betriebssystem: Alle
- Standardwert=NO, Werte: YES oder NO
- Der Vorablesezugriff über Listen ist eine spezielle Zugriffsmethode auf Tabellen, bei der die den Bedingungen entsprechenden Satz-IDs (RIDs) aus einem Index abgerufen, nach Seitennummer sortiert und anschließend die Datenseiten vorab gelesen werden. Manchmal verfügt das Optimierungsprogramm über keine genauen Informationen, um zu ermitteln, ob ein Vorablesezugriff über Listen eine gute Zugriffsmethode ist. Dies kann der Fall sein, wenn die Selektivitäten von Vergleichselementen Parametermarken oder Hostvariablen enthalten, die verhindern, dass das Optimierungsprogramm die Selektivität mithilfe der Katalogstatistiken ermitteln kann.

  Diese Registrierdatenbankvariable verhindert, dass das Optimierungsprogramm in solchen Situationen einen Vorablesezugriff über Listen in Betracht zieht.

  Anmerkung: Wenn mindestens eine der beiden DB2-Abfragecompilervariablen **DB2_MINIMIZE_LISTPREFETCH** und **DB2_INLIST_TO_NLJN** auf YES gesetzt ist, bleibt sie auch dann aktiv, wenn REOPT(ONCE) angegeben wird.

DB2_NEW_CORR_SQ_FF

- Betriebssystem: Alle
- Standardwert=OFF, Werte: ON oder OFF
- Beeinflusst, wenn auf ON gesetzt, den Selektivitätswert, der vom Abfrageoptimierungsprogramm für bestimmte Vergleichselemente von Unterabfragen berechnet wird. Diese Variable dient zur Verbesserung der Genauigkeit des Selektivitätswerts von Gleichheitsvergleichselementen in Unterabfragen, die die Spaltenfunktion MIN oder MAX in der SELECT-Liste der Unterabfrage enthalten. Beispiel:

  ```
  SELECT * FROM T WHERE
  T.COL = (SELECT MIN(T.COL)
  FROM T WHERE ...)
  ```

DB2_OPT_MAX_TEMP_SIZE

- Betriebssystem: Alle
- Standardwert=NULL, Werte: Größe des Speicherbereichs in Megabyte, der von einer Abfrage in allen Tabellenbereichen für temporäre Tabellen verwendet werden kann

- Begrenzt die Größe des Speicherbereichs, den Abfragen in den temporären Tabellenbereichen nutzen können. Wenn die Variable **DB2_OPT_MAX_TEMP_SIZE** definiert wird, kann sie das Optimierungsprogramm veranlassen, einen aufwendigeren Plan als sonst auszuwählen, der jedoch weniger Speicherplatz in den Tabellenbereichen für temporäre Tabellen erfordert. Wenn Sie die Variable **DB2_OPT_MAX_TEMP_SIZE** definieren, stellen Sie sicher, dass Sie die Notwendigkeit, den Bedarf an temporärem Tabellenbereich einzuschränken, gegen die Effizienz des Plans, der durch Ihre Einstellung ausgewählt wird, geeignet abwägen.

  Wenn **DB2_WORKLOAD=SAP** definiert wird, wird die Variable **DB2_OPT_MAX_TEMP_SIZE** automatisch auf den Wert 10.240 (10 GB) gesetzt.

  Wenn Sie eine Abfrage ausführen, die über den für **DB2_OPT_MAX_TEMP_SIZE** festgelegten Wert hinaus temporären Tabellenbereich belegt, schlägt die Abfrage nicht fehl, jedoch empfangen Sie eine Warnung, dass die Leistung vielleicht nicht optimal ist, da möglicherweise nicht alle Ressourcen verfügbar sind.

  Die folgenden vom Optimierungsprogramm in Betracht gezogenen Operationen sind von dem durch **DB2_OPT_MAX_TEMP_SIZE** definierten Grenzwert betroffen:

  – Explizite Sortierungen für Operationen wie ORDER BY, DISTINCT, GROUP BY, Mischsuchjoins und Joins mit Verschachtelungsschleife
  – Explizite temporäre Tabellen
  – Implizite temporäre Tabellen für Hash-Joins und duplizierte Mischjoins

DB2_REDUCED_OPTIMIZATION

- Betriebssystem: Alle
- Standardwert=NO, Werte: NO, YES, beliebige Ganzzahl, DISABLE, NO_SORT_NLJOIN oder NO_SORT_MGJOIN
- Mit dieser Registrierdatenbankvariablen können Sie entweder reduzierte Optimierungsfunktionen oder die strenge Anwendung der Optimierungsfunktionen der angegebenen Optimierungsklasse anfordern. Wenn Sie die Anzahl der verwendeten Optimierungstechniken reduzieren, können Sie dadurch die Zeit und die Ressourcenbelastung bei der Optimierung verringern.

  Anmerkung: Die Optimierungszeit und die Ressourcenbelastung wird zwar reduziert, jedoch erhöht sich das Risiko, dass ein nicht optimaler Datenzugriffsplan generiert wird. Verwenden Sie diese Registrierdatenbankvariable nur, wenn Sie von IBM oder einem IBM Partner dazu aufgefordert werden.

  – Wenn der Wert NO definiert ist

    Das Optimierungsprogramm ändert seine Optimierungstechniken nicht.

  – Wenn der Wert YES definiert ist

    Wenn die Optimierungsklasse 5 (Standardwert) oder eine niedrigere verwendet wird, inaktiviert das Optimierungsprogramm einige Optimierungstechniken, die möglicherweise viel Vorbereitungszeit und Ressourcen beanspruchen, jedoch in der Regel zur Generierung besserer Zugriffspläne führen.

Wenn die Optimierungsklasse exakt 5 ist, reduziert das Optimierungsprogramm einige zusätzliche Techniken oder inaktiviert sie, wodurch sich die Optimierungszeit und die Ressourcenbelastung weiter verringern können, jedoch gleichzeitig die Gefahr wächst, dass ein nicht optimaler Zugriffsplan generiert wird. Bei Optimierungsklassen unter 5 kommen einige dieser Techniken möglicherweise ohnehin nicht zur Anwendung. Wenn sie dennoch angewandt werden, bleiben sie jedoch wirksam.

– Wenn eine beliebige Ganzzahl definiert ist

  Der Effekt ist der gleiche wie bei YES jedoch mit folgenden zusätzlichen Merkmalen für dynamisch vorbereitete Abfragen, die mit Klasse 5 optimiert werden. Wenn die Gesamtanzahl von Joins in einem beliebigen Abfrageblock die Einstellung überschreitet, wechselt das Optimierungsprogramm zur schnellen Joinaufzählung (Greedy Join Enumeration), anstatt zusätzliche Optimierungstechniken wie oben für die Optimierungsklasse 5 beschrieben zu inaktivieren. Dies impliziert, dass die Abfrage mit einer ähnlichen Klasse wie Optimierungsklasse 2 optimiert wird.

– Wenn der Wert DISABLE definiert ist

  Wenn keine Einschränkung durch diese Variable **DB2_REDUCED_OPTIMIZATION** definiert ist, sieht die Funktionsweise des Optimierungsprogramms zuweilen vor, die Optimierung für dynamische Abfragen in Optimierungsklasse 5 dynamisch zu reduzieren. Diese Einstellung setzt diese Funktionsweise außer Kraft und zwingt das Optimierungsprogramm, die volle Optimierung der Klasse 5 durchzuführen.

– Wenn der Wert NO_SORT_NLJOIN definiert ist

  Das Optimierungsprogramm generiert keine Abfragepläne, die Sortierungen für Joins mit Verschachtelungsschleife (NLJN, Nested Loop Join) erzwingen. Diese Typen von Sortierungen können zur Steigerung der Leistung nützlich sein. Daher ist bei der Verwendung der Option NO_SORT_NLJOIN Vorsicht geboten, da die Leistung erheblich beeinträchtigt werden kann.

– Wenn der Wert NO_SORT_MGJOIN definiert ist

  Das Optimierungsprogramm generiert keine Abfragepläne, die Sortierungen für Mischsuchjoins (MSJN, Merge-Scan-Joins) erzwingen. Diese Typen von Sortierungen können zur Steigerung der Leistung nützlich sein. Daher ist bei der Verwendung der Option NO_SORT_MGJOIN Vorsicht geboten, da die Leistung erheblich beeinträchtigt werden kann.

  Beachten Sie, dass die dynamische Reduzierung der Optimierung in Optimierungsklasse 5 Vorrang vor der Funktionsweise hat, die für die exakte Optimierungsklasse 5 beschrieben ist, wenn die Variable **DB2_REDUCED_OPTIMIZATION** auf den Wert YES gesetzt ist, sowie vor der Funktionsweise, die für die ganzzahlige Einstellung beschrieben ist.

DB2_SELECTIVITY

- Betriebssystem: Alle
- Standardwert=NO, Werte: YES oder NO
- Diese Registrierdatenbankvariable steuert, wo die Klausel SELECTIVITY in Suchbedingungen in SQL-Anweisungen verwendet werden kann.

Wenn diese Registrierdatenbankvariable auf YES gesetzt ist, kann die Klausel SELECTIVITY für die folgenden Vergleichselemente angegeben werden:

– Ein einfaches Vergleichselement, in dem mindestens ein Ausdruck Hostvariablen enthält
– Ein LIKE-Vergleichselement, in dem Abgleichausdruck (MATCH), der Vergleichselementausdruck oder der Escape-Ausdruck Hostvariablen enthält

DB2_SQLROUTINE_PREPOPTS

- Betriebssystem: Alle
- Standardwert=Leere Zeichenfolge, Werte:
  – BLOCKING {UNAMBIG | ALL | NO}
  – DATETIME {DEF | USA | EUR | ISO | JIS | LOC}
  – DEGREE {1 | *grad-der-parallelität* | ANY}
  – DYNAMICRULES {BIND | INVOKEBIND | DEFINEBIND | RUN | INVOKERUN | DEFINERUN}
  – EXPLAIN {NO | YES | ALL}
  – EXPLSNAP {NO | YES | ALL}
  – FEDERATED {NO | YES}
  – INSERT {DEF | BUF}
  – ISOLATION {CS | RR | UR | RS | NC}
  – QUERYOPT *optimierungsgrad*
  – REOPT {NONE | ONCE | ALWAYS}
  – VALIDATE {RUN | BIND}
- Die Registrierdatenbankvariable **DB2_SQLROUTINE_PREPOPTS** kann zur Anpassung der Vorkompilierungs- und Bindeoptionen für SQL- und XQuery-Prozeduren verwendet werden. Wenn Sie diese Variable definieren, trennen Sie die einzelnen Optionen durch ein Leerzeichen wie im folgenden Beispiel:

  ```
  db2set DB2_SQLROUTINE_PREPOPTS="BLOCKING ALL VALIDATE RUN"
  ```

  Eine vollständige Beschreibung der einzelnen Optionen und ihrer Einstellungen finden Sie in den Informationen zum Befehl BIND.

Wenn Sie die gleichen Ergebnisse wie durch **DB2_SQLROUTINE_PREPOPTS** für einzelne ausgewählte Prozeduren erzielen möchten, jedoch ohne die Instanz erneut zu starten, verwenden Sie die Prozedur SET_ROUTINE_OPTS.

Leistungsvariablen

DB2_ALLOCATION_SIZE

- Betriebssystem: Alle
- Standardwert: 128 KB, Bereich: 64 KB bis 256 MB
- Gibt die Größe von Speicherzuordnungen für Pufferpools an.

  Wenn diese Registrierdatenbankvariable auf einen höheren Wert gesetzt wird, besteht der potenzielle Vorteil darin, dass weniger Zuordnungen erforderlich sind, um eine gewünschte Größe des Speichers für einen Pufferpool zu erreichen.

Der potenzielle Nachteil eines höheren Werts für diese Registrierdatenbankvariable besteht darin, dass Speicher verschwendet werden kann, wenn der Pufferpool um einen Wert geändert wird, der nicht einem Vielfachen der Zuordnungsgröße entspricht. Wenn zum Beispiel die Variable **DB2_ALLOCATION_SIZE** auf den Wert 8 gesetzt ist und ein Pufferpool um 4 MB verkleinert wird, werden diese 4 MB verschwendet, weil kein ganzes 8-MB-Segment freigegeben werden kann.

Anmerkung: DB2_ALLOCATION_SIZE ist veraltet und wird in einem späteren Release möglicherweise entfernt.

DB2_APM_PERFORMANCE

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Setzen Sie diese Variable auf den Wert ON, um die leistungsrelevanten Änderungen am Zugriffsplanmanager (APM) zu aktivieren, die sich auf die Funktionsweise des Abfragecache (Paketcache) auswirken. Diese Einstellungen werden für Produktionssysteme im Normalfall nicht empfohlen. Sie verursachen einige Einschränkungen, wie zum Beispiel die Möglichkeit von Fehlern aufgrund nicht ausreichenden Paketcacheplatzes oder erhöhter Speicherauslastung (oder beider Ursachen).

  Die Einstellung der Variablen **DB2_APM_PERFORMANCE** auf den Wert ON aktiviert außerdem den Modus NO PACKAGE LOCK ('Keine Paketsperre'). Dieser Modus ermöglicht den Betrieb des globalen Abfragecache ohne Verwendung von Paketsperren, bei denen es sich um interne Systemsperren handelt, die Paketeinträge im Cache davor schützen, entfernt zu werden. Der Modus NO PACKAGE LOCK kann zwar zu einer etwas verbesserten Leistung führen, jedoch sind bestimmte Datenbankoperationen nicht zulässig. Zu diesen unzulässigen Operationen können gehören: Operationen, die Pakete ungültig machen, Operationen, die Pakete funktionsunfähig machen, sowie die Operationen PRECOMPILE, BIND und REBIND.

DB2ASSUMEUPDATE

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Wenn diese Variable aktiviert ist, kann das DB2-Datenbanksystem annehmen, dass alle Spalten fester Länge, die in einer UPDATE-Anweisung angegeben werden, geändert werden. Dadurch muss das DB2-Datenbanksystem keinen Vergleich zwischen den vorhandenen Spaltenwerten und den angegebenen neuen Spaltenwerten durchführen, um festzustellen, ob sich die Spalte tatsächlich ändert. Wenn bei Verwendung dieser Registrierdatenbankvariablen Spalten zur Aktualisierung angegeben werden (z. B. in einer SET-Klausel), jedoch nicht wirklich geändert werden, kann dies zusätzliche Protokoll- und Indexpflegeaktivitäten zur Folge haben.

  Die Aktivierung der Registrierdatenbankvariablen **DB2ASSUMEUPDATE** erfolgt mit dem Befehl db2start.

DB2_ASYNC_IO_MAXFILOP

- Betriebssystem: Alle
- Standardwert: Wert des Konfigurationsparameters **maxfilop**; Werte: vom Wert für **maxfilop** bis zum Wert für **max_int**
- **DB2_ASYNC_IO_MAXFILOP** ist veraltet und wird in einem späteren Release möglicherweise entfernt. Diese Variable ist aufgrund der gemein-

sam genutzten Tabelle für Dateikennungen, die vom Datenbankmanager im Multithreadmodus verwaltet wird, nicht mehr benötigt. Weitere Informationen finden Sie im Abschnitt "Gemeinsam genutzte Tabelle für Dateikennungen" in der Veröffentlichung *Datenserver, Datenbanken und Datenbankobjekte*.

Die Variable **DB2_ASYNC_IO_MAXFILOP** kann in Version 9.5 weiterhin definiert werden, jedoch hat sie keine Wirkung. Wenn Sie die Anzahl der Dateikennungen begrenzen wollen, die für jede Datenbank geöffnet sein können, verwenden Sie den Konfigurationsparameter **maxfilop**.

DB2_AVOID_PREFETCH

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Definiert, ob ein Vorablesezugriff bei der Recovery nach einem Systemabsturz verwendet werden soll. Wenn **DB2_AVOID_PREFETCH** =ON definiert ist, wird der Vorablesezugriff nicht verwendet.

DB2BPVARS

- Betriebssystem: Wie für jeden Parameter angegeben
- Standardwert: Pfad
- Es sind zwei Gruppen von Parametern zur Optimierung von Pufferpools verfügbar. Die eine Gruppe von Parametern, die nur unter Windows zur Verfügung steht, gibt an, dass Pufferpools SCATTER-Lesezugriffe für bestimmte Typen von Containern verwenden sollen. Die andere Gruppe von Parametern, die auf allen Plattformen verfügbar ist, beeinflusst die Funktionsweise des Vorablesezugriffs.

  Die Parameter werden in einer ASCII-Datei, jeweils ein Parameter in einer Zeile, in der Form `parameter=wert` angegeben. Zum Beispiel könnte eine Datei mit dem Namen bpvars.vars die folgenden Zeilen enthalten:

  ```
  NO_NT_SCATTER = 1
  NUMPREFETCHQUEUES = 2
  ```

  Wenn die Datei bpvars.vars beispielsweise im Pfad F:\vars\ gespeichert ist, führen Sie zur Definition dieser Variablen den folgenden Befehl aus:

  ```
  db2set DB2BPVARS=F:\vars\bpvars.vars
  ```

  Parameter für SCATTER-Lesezugriff

  Die Parameter für den SCATTER-Lesezugriff werden für Systeme mit einem hohen Aufkommen an sequenziellen Vorableseoperationen für die jeweilige Art von Container empfohlen, für die Sie bereits den Parameter **DB2NTNOCACHE** auf ON gesetzt haben. Diese Parameter, die nur auf Windows-Plattformen verfügbar sind, heißen **NT_SCATTER_DMSFILE**, **NT_SCATTER_DMSDEVICE** und **NT_SCATTER_SMS**. Geben Sie den Parameter **NO_NT_SCATTER** an, um den SCATTER-Lesezugriff für einen Container explizit auszuschließen. Bestimmte Parameter werden dazu verwendet, den SCATTER-Lesezugriff für alle Container des angegebenen Typs zu aktivieren. Für jeden dieser Parameter ist der Standardwert 0 (bzw. OFF); die möglichen Werte sind: 0 (bzw. OFF) und 1 (bzw. ON).

  Anmerkung: Sie können den SCATTER-Lesezugriff nur aktivieren, wenn **DB2NTNOCACHE** auf den Wert ON gesetzt ist, um die Windows-Dateicachefunktion abzuschalten. Wenn **DB2NTNOCACHE** auf den Wert OFF gesetzt oder nicht definiert ist, wird eine Warnung in das

Benachrichtigungsprotokoll für die Systemverwaltung geschrieben, wenn Sie versuchen, den SCATTER-Lesezugriff für einen Container zu aktivieren. Der SCATTER-Lesezugriff bleibt in diesem Fall inaktiviert.

Parameter zur Anpassung des Vorablesezugriffs

Die Parameter zur Anpassung des Vorablesezugriffs heißen **NUMPREFETCHQUEUES** und **PREFETCHQUEUESIZE**. Diese Parameter stehen auf allen Plattformen zur Verfügung und können zur Verbesserung des Datenvorablesezugriffs für Pufferpools verwendet werden. Betrachten Sie zum Beispiel den sequenziellen Vorablesezugriff, bei dem der gewünschte **PREFETCHSIZE**-Wert in **PREFETCHSIZE**/**EXTENTSIZE** Vorablesezugriffsanforderung dividiert wird. In diesem Fall werden die Anforderungen in Vorablesewarteschlangen gestellt, aus denen E/A-Server zugeteilt werden, um asynchrone E/A-Operationen auszuführen. Standardmäßig verwaltet der DB2-Datenbankmanager eine Warteschlange in der Größe max(200,2***NUM_IOSERVERS** für jede Datenbankpartition. In einigen Umgebungen lässt sich die Leistung entweder durch mehr Warteschlangen und/oder durch Warteschlangen einer anderen Größe verbessern. Die Anzahl der Warteschlangen für den Vorablesezugriff sollte höchstens die Hälfte der Anzahl von E/A-Servern betragen. Berücksichtigen Sie bei der Einstellung dieser Parameter andere Parameter, wie beispielsweise **PREFETCHSIZE**, **EXTENTSIZE**, **NUM_IOSERVERS** und die Pufferpoolgröße, sowie Auslastungsmerkmale, wie z. B. die Anzahl der aktuellen Benutzer.

Wenn Sie meinen, dass die Standardwerte für Ihre Umgebung zu klein sind, erhöhen Sie die Werte zunächst nur leicht. Zum Beispiel könnten Sie **NUMPREFETCHQUEUES**=4 und **PREFETCHQUEUESIZE**=200 setzen. Nehmen Sie Änderungen an diesen Parametern kontrolliert vor, sodass Sie die Wirkungen der Änderung überwachen und bewerten können.

Für **NUMPREFETCHQUEUES** ist der Standardwert 1, und der Wertebereich ist 1 bis **NUM_IOSERVERS**. Wenn Sie **NUMPREFETCHQUEUES** auf einen kleineren Wert als 1 setzen, wird er auf den Wert 1 angepasst. Wenn Sie ihn auf einen höheren Wert als **NUM_IOSERVERS** setzen, wird er auf den Wert von **NUM_IOSERVERS** angepasst.

Der Standardwert für **PREFETCHQUEUESIZE** ist max(200,2***NUM_IOSERVERS**). Der Wertebereich liegt zwischen 1 und 32767. Wenn Sie **PREFETCHQUEUESIZE** auf einen kleineren Wert als 1 setzen, wird er auf den Standardwert angepasst. Wenn er auf einen höheren Wert als 32767 gesetzt wird, wird er auf den Wert 32767 angepasst.

Anmerkung: **DB2BPVARS** ist veraltet und wird in einem späteren Release möglicherweise entfernt.

DB2CHKPTR

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Definiert, ob die Zeigerprüfung für Eingaben erforderlich ist oder nicht.

DB2CHKSQLDA

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON oder OFF
- Definiert, ob die Prüfung von SQL-Deskriptorbereichen (SQLDA) für Eingaben erforderlich ist oder nicht.

DB2_EVALUNCOMMITTED

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON, OFF
- Wenn sie aktiviert ist, ermöglicht diese Variable, dass bei Tabellen- oder Indexsuchzugriffen Zeilensperren nach Möglichkeit so lange verzögert bzw. vermieden werden, bis feststeht, dass ein Datensatz der Auswertung der Vergleichselemente entspricht.

  Wenn diese Variable aktiviert ist, findet möglicherweise auch eine Auswertung von Vergleichselementen für nicht festgeschriebene Daten statt.

  DB2_EVALUNCOMMITTED gilt nur für Anweisungen, die mit der Isolationsstufe Cursorstabilität oder Lesestabilität arbeiten. Bei Indexsuchen muss es sich um einen Index des Typs 2 handeln.

  Darüber hinaus werden beim Tabellensuchzugriff gelöschte Zeilen bedingungslos übersprungen, während gelöschte Schlüssel beim Durchsuchen von Indizes des Typs 2 nicht übersprungen werden, sofern nicht die Registrierdatenbankvariable **DB2_SKIPDELETED** ebenfalls definiert ist.

  Die Aktivierung der Registrierdatenbankvariablen **DB2_EVALUNCOMMITTED** erfolgt mit dem Befehl db2start. Die Entscheidung, ob ein verzögertes Sperren anwendbar ist, wird beim Kompilieren oder Binden der Anweisung getroffen.

DB2_EXTENDED_IO_FEATURES

- Betriebssystem: AIX
- Standardwert: OFF, Werte: ON, OFF
- Setzen Sie diese Variable auf den Wert ON, um die Funktionsmerkmale zur Verbesserung der E/A-Leistung zu aktivieren. Zu diesen Verbesserungen gehören eine Erhöhung der Trefferrate von Hauptspeichercaches sowie eine Verringerung der Latenzzeit für E/A-Operationen mit hoher Priorität. Diese Funktionen sind nur für bestimmte Kombinationen aus Software- und Hardwarekonfigurationen verfügbar. Für andere Konfigurationen wird die Einstellung ON vom DB2-Datenbankverwaltungssystem oder vom Betriebssystem ignoriert. Eine Konfiguration muss die folgenden Mindestvoraussetzungen erfüllen:
  - Datenbankversion: DB2 Version 9.1
  - Eine Roheinheit (RAW) muss für Datenbankcontainer verwendet werden (Container in Dateisystemen werden nicht unterstützt).
  - Betriebssystem: AIX 5.3 TL4
  - Speichersubsystem: Shark DS8000 unterstützt alle erweiterten E/A-Leistungsfunktionen. Informationen zur Installation und zu den vorausgesetzten Komponenten finden Sie in der Dokumentation zum Shark DS8000.

  Die Standardeinstellungen der E/A-Priorität für HIGH, MEDIUM und LOW sind 3, 8 bzw. 12. Mithilfe der Registrierdatenbankvariablen **DB2_IO_PRIORITY_SETTING** können Sie diese Einstellungen ändern.

DB2_EXTENDED_OPTIMIZATION

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON, OFF oder ENHANCED_MULTIPLE_DISTINCT
- Diese Variable gibt an, ob das Abfrageoptimierungsprogramm die Optimierungserweiterungen zum Verbessern der Abfrageleistung verwendet. Die Werte ON und ENHANCED_MULTIPLE_DISTINCT geben

unterschiedliche Optimierungserweiterungen an. Verwenden Sie eine durch Komma getrennte Liste, wenn beide verwendet werden sollen.

Der Wert ENHANCED_MULTIPLE_DISTINCT kann zur Leistungsverbesserung von Abfragen beitragen, wenn mehrere unterschiedliche Spaltenoperationen in einer einzigen SELECT-Operation enthalten sind und wenn das Verhältnis der Prozessoren zur Anzahl der Datenbankpartitionen niedrig ist (z. B. kleiner oder gleich 1). Diese Einstellung muss in DPF-Umgebungen (DPF = Database Partitioning Feature) ohne symmetrische Mehrprozessoren (SMPs) verwendet werden.

Die Optimierungserweiterungen verbessern die Abfrageleistung möglicherweise nicht in allen Umgebungen. Einzelne Verbesserungen in der Abfrageleistung sollten durch Tests ermittelt werden.

DB2_HASH_JOIN

- Betriebssystem: Alle
- Standardwert: YES, Werte: YES oder NO
- Gibt Hash-Join als mögliche Joinmethode beim Kompilieren eines Zugriffsplans an. **DB2_HASH_JOIN** muss optimiert werden, um die beste Leistung zu erzielen. Die Leistung von Hash-Joins ist am besten, wenn Sie Hashschleifen und Überläufe auf den Plattenspeicher vermeiden können. Zur Optimierung der Leistung von Hash-Joins schätzen Sie die maximale Speichergröße ab, die für den Konfigurationsparameter **sheapthres** verfügbar ist, und optimieren dann den Konfigurationsparameter **sortheap**. Erhöhen Sie den Wert dieses Parameters, bis Sie möglichst viele Hashschleifen und Überläufe auf den Plattenspeicher vermeiden, jedoch nicht den durch den Konfigurationsparameter **sheapthres** definierten Grenzwert erreichen.

  Anmerkung: DB2_HASH_JOIN ist in Version 9.5 veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

DB2_IO_PRIORITY_SETTING

- Betriebssystem: AIX
- Werte: HIGH:#,MEDIUM:#,LOW:# mit # aus dem Bereich von 1 bis 15
- Diese Variable wird in Kombination mit der Registrierdatenbankvariablen **DB2_EXTENDED_IO_FEATURES** verwendet. Diese Registrierdatenbankvariable bietet die Möglichkeit, die Standardeinstellungen der Werte HIGH, MEDIUM und LOW für die E/A-Priorität für das DB2-Datenbanksystem zu ändern. Die Standardwerte sind 3, 8 bzw. 12. Diese Registrierdatenbankvariable muss vor dem Start einer Instanz definiert werden. Jede Änderung erfordert einen Neustart der Instanz. Beachten Sie, dass die Einstellung dieser Registrierdatenbankvariablen allein die erweiterten E/A-Funktionen nicht aktiviert. Diese müssen mit der Variablen **DB2_EXTENDED_IO_FEATURES** aktiviert werden. Alle Systemvoraussetzungen für **DB2_EXTENDED_IO_FEATURES** gelten auch für diese Registrierdatenbankvariable.

DB2_KEEP_AS_AND_DMS_CONTAINERS_OPEN

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO.
- Wenn Sie diese Variable auf NO setzen, hat jeder DMS-Tabellenbereichscontainer eine Dateikennung geöffnet, bis die Datenbank inaktiviert wird. Die Abfrageleistung kann sich erhöhen, da der Systemaufwand zum Öffnen der Container entfällt. Sie sollten diese Registrierdatenbank

nur in reinen DMS-Umgebungen verwenden, andernfalls kann die Leistung der Abfragen für SMS-Tabellenbereiche negativ beeinflusst werden.

DB2_KEEPTABLELOCK

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON, TRANSACTION, OFF, CONNECTION
- Wenn diese Variable auf ON oder TRANSACTION gesetzt ist, kann das DB2-Datenbanksystem mit dieser Variablen die Tabellensperre beibehalten, wenn eine Isolationsstufe UR (Nicht festgeschriebener Lesevorgang) oder CS (Cursorstabilität) geschlossen wird. Die Tabellensperre, die beibehalten wird, wird am Ende der Transaktion freigegeben, wie dies bei Suchvorgängen mit der Isolationsstufe RS (Lesestabilität) und RR (Wiederholtes Lesen) der Fall ist.

  Wenn diese Variable auf CONNECTION gesetzt ist, wird eine Tabellensperre für eine Anwendung freigegeben, bis die Anwendung ein Rollback für die Transaktion durchführt oder die Verbindung zurückgesetzt wird. Die Tabellensperre wird weiterhin während Commitoperationen gehalten, und Anwendungsanforderungen zum Löschen der Tabellensperre werden von der Datenbank ignoriert. Die Tabellensperre bleibt weiterhin der Anwendung zugeordnet. Wenn daher die Anwendung die Tabellensperre erneut anfordert, ist die Sperre bereits verfügbar.

  Für Anwendungsauslastungen, die diese Optimierung nutzen, sollte sich die Leistung verbessern. Die Auslastungen anderer Anwendungen, die gleichzeitig ausgeführt werden, könnten jedoch beeinflusst werden. Andere Anwendungen könnten beim Zugriff auf eine vorgegebene Tabelle blockiert werden, was zu einem schlechten gemeinsamen Zugriff führt. DB2-SQL-Katalogtabellen sind von dieser Einstellung nicht betroffen. Die Einstellung CONNECTION schließt dieses Verhalten ebenfalls mit ein, das für die Einstellung ON oder TRANSACTION beschrieben wird.

  Diese Registrierdatenbankvariable wird zum Zeitpunkt des Kompilierens oder Bindens einer Anweisung geprüft.

DB2_LARGE_PAGE_MEM

- Betriebssystem: AIX, Linux, Windows Server 2003
- Standardwert: NULL, Werte: Geben Sie mithilfe eines Sterns (*) an, dass alle gültigen Speicherbereiche große Seitenspeicher verwenden sollen, oder geben Sie eine durch Kommas getrennte Liste bestimmter Speicherbereiche an, die den großen Seitenspeicher verwenden sollen. Die verfügbaren Bereiche variieren je nach dem verwendeten Betriebssystem. Unter AIX können die folgenden Bereiche angegeben werden: DB, DBMS, FCM oder PRIVATE. Unter Linux kann der folgende Bereich angegeben werden: DB. Unter Windows Server 2003 kann der folgende Bereich angegeben werden: DB. Die Unterstützung sehr großer Speicherseiten (so genannte Huge-Pages) ist nur unter AIX verfügbar.
- Die Registrierdatenbankvariable **DB2_LARGE_PAGE_MEM** wird dazu verwendet, die Unterstützung für große oder sehr große Speicherseiten (so genannte Huge-Pages) zu aktivieren. Durch die Definition von **DB2_LARGE_PAGE_MEM**=DB wird der Speicher mit großen Seiten für den gemeinsam genutzten Bereich des Datenbankspeichers aktiviert. Wird für **database_memory** der Wert AUTOMATIC definiert, wird die automatische Optimierung dieses gemeinsam genutzten Speicherbereichs durch STMM inaktiviert. Unter AIX wird mit der Einstellung DB2_LARGE_PAGE_MEM=DB:16GB der Huge-Page-Speicher für den gemeinsam genutzten Bereich des Datenbankspeichers aktiviert.

Bei Anwendungen, die durch einen intensiven Speicherzugriff gekennzeichnet sind und große Mengen an virtuellem Speicher verwenden, kann die Verwendung großer bzw. sehr großer Speicherseiten zu einer Leistungsverbesserung führen. Um das DB2-Datenbanksystem für die Verwendung großer bzw. sehr großer Speicherseiten einzurichten, müssen Sie zunächst das Betriebssystem für die Verwendung solcher Seiten konfigurieren.

Zum Aktivieren großer Seiten für privaten Agentenspeicher unter DB2 für AIX (64 Bit) über die Einstellung **DB2_LARGE_PAGE_MEM**=PRIVATE müssen Sie das Betriebssystem für die Verwendung großer Seiten konfigurieren, und der Instanzeigner muss über die Funktionen CAP_BYPASS_RAC_VMM und CAP_PROPAGATE verfügen.

Unter AIX 5L können Sie diese Variable auf FCM setzen. Da sich der FCM-Speicher in einer eigenen Speichergruppe befindet, müssen Sie das Schlüsselwort FCM dem Wert der Registrierdatenbankvariablen **DB2_LARGE_PAGE_MEM** hinzufügen, um große Seiten für den FCM-Speicher zu aktivieren.

Unter Linux besteht die zusätzliche Voraussetzung, dass die Bibliothek libcap.so.1 verfügbar sein muss. Diese Bibliothek muss installiert sein, damit diese Option funktioniert. Wenn diese Option aktiviert wird und die Bibliothek im System nicht vorhanden ist, inaktiviert die DB2-Datenbank die großen Kernelseiten und setzt die Verarbeitung wie ohne sie fort.

Zum Prüfen, ob unter Linux große Kernelseiten verfügbar sind, können Sie den folgenden Befehl absetzen:

```
cat /proc/meminfo
```

Wenn große Kernelseiten verfügbar sind, sollten die drei folgenden Zeilen (mit je nach Größe des auf Ihrem Server konfigurierten Speichers unterschiedlichen Werten) angezeigt werden:

```
HugePages_Total:   200
HugePages_Free:    200
Hugepagesize:    16384 kB
```

Wenn diese Zeilen nicht angezeigt werden oder `HugePages_Total` den Wert 0 hat, ist eine Konfiguration des Betriebssystems oder des Kernels erforderlich.

Unter Windows ist die Menge an Speicher mit großen Seiten, die auf dem System verfügbar ist, kleiner als der insgesamt verfügbare Speicher. Wenn das System einige Zeit aktiv war, kann der Speicher fragmentiert werden, sodass die Menge an Speicher mit großen Seiten abnimmt.

DB2_LOGGER_NON_BUFFERED_IO

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON, OFF
- Diese Variable ermöglicht die direkte Ein-/Ausgabe auf dem Protokolldateisystem.

Die Registrierdatenbankvariable **DB2_LOGGER_NON_BUFFERED_IO** ist ab DB2 Version 9.5 Fixpack 1 verfügbar.

DB2MAXFSCRSEARCH

- Betriebssystem: Alle
- Standardwert: 5, Werte: -1, 1 bis 33554

- Gibt die Anzahl der Steuersätze für freien Speicherbereich (FSCRs - Free Space Control Records) an, die beim Einfügen eines Eintrags in eine Tabelle zu durchsuchen sind. Standardmäßig werden fünf Steuersätze für freien Speicherbereich durchsucht. Mit diesem Wert können Sie die Gewichtung zwischen Einfügegeschwindigkeit und Speicherwiederverwendung beeinflussen. Mit hohen Werten optimieren Sie die Speicherwiederverwendung. Mit niedrigen Werten optimieren Sie die Einfügegeschwindigkeit. Der Wert -1 zwingt den Datenbankmanager, alle Steuersätze für freien Speicherbereich zu durchsuchen.

DB2_MAX_INACT_STMTS

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: bis zu 4 GB
- Diese Variable überschreibt den Standardgrenzwert für die Anzahl inaktiver Anweisungen, die jeweils von einer Anwendung behalten werden. Sie können einen anderen Wert auswählen, um die Größe des Systemmonitorzwischenspeichers zu erhöhen bzw. zu verringern, der für Informationen inaktiver Anweisungen verwendet wird. Der Standardgrenzwert ist 250.

  Die Kapazität des Systemmonitorzwischenspeichers kann sich erschöpfen, wenn eine Anwendung eine sehr große Anzahl von Anweisungen in einer UOW (Unit of Work, Arbeitseinheit) enthält oder wenn eine große Anzahl von Anwendungen gleichzeitig ausgeführt wird.

DB2_MAX_NON_TABLE_LOCKS

- Betriebssystem: Alle
- Standardwert: YES, Werte: Siehe Beschreibung
- Diese Variable definiert die maximale Anzahl von NON-Tabellensperren, die eine Transaktion haben kann, bevor sie alle diese Sperren freigibt. NON-Tabellensperren sind Tabellensperren, die in der Hashtabelle und der Transaktionskette verbleiben, selbst wenn sie von der Transaktion nicht mehr verwendet werden. Da Transaktionen häufig mehrmals auf die gleiche Tabelle zugreifen, kann das Beibehalten der Sperren und das Ändern ihres Status in NON die Leistung verbessern.

  Im Hinblick auf optimale Ergebnisse entspricht der für diese Variable empfohlene Wert der maximalen Anzahl von Tabellen, auf die voraussichtlich durch eine beliebige Verbindung zugegriffen wird. Wenn kein benutzerdefinierter Wert angegeben wird, gilt folgender Standardwert: Wenn die Sperrenliste größer oder gleich

  ```
  SQLP_THRESHOLD_VAL_OF_LRG_LOCKLIST_SZ_FOR_MAX_NON_LOCKS
  ```

  (zurzeit 8000) ist, ist der Standardwert

  ```
  SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_LARGE
  ```

  (zurzeit 150). Ansonsten ist der Standardwert

  ```
  SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_SMALL
  ```

  (zurzeit 0).

DB2_MDC_ROLLOUT

- Betriebssystem: Alle
- Standardwert: IMMEDIATE, Werte: IMMEDIATE, OFF oder DEFER
- Mit dieser Variablen wird eine Leistungsverbesserung für Löschvorgänge in MDC-Tabellen aktiviert, die als „Rollout" bezeichnet wird. Ein Rollout

ist eine schnellere Möglichkeit, Zeilen in einer MDC-Tabelle zu löschen, wenn ganze Zellen (Schnittpunkte von Dimensionswerten) in einer Suchanweisung mit DELETE gelöscht werden. Die Vorteile sind eine reduzierte Protokollierung und eine effizientere Verarbeitung.

- Die Einstellung der Variablen kann drei mögliche Ergebnisse haben:
  - Kein Rollout - wenn der Wert OFF angegeben wird.
  - Sofortiger Rollout - wenn der Wert IMMEDIATE angegeben wird.
  - Rollout mit verzögerter Indexbereinigung - wenn der Wert DEFER angegeben wird.
- Wenn der Wert nach dem Start geändert wird, berücksichtigen alle neuen Kompilierungen einer Anweisung die neue Einstellung der Registrierdatenbankvariablen. Bei Anweisungen, die sich im Paketcache befinden, erfolgt die Änderung in der Löschverarbeitung erst, wenn die Anweisung erneut kompiliert wird. Die Anweisung SET CURRENT MDC ROLLOUT MODE überschreibt den Wert der Variablen **DB2_MDC_ROLLOUT** auf der Anwendungsverbindungsebene.

DB2MEMDISCLAIM

- Betriebssystem: Alle
- Standardwert: YES, Werte: YES oder NO.
- Der von den Prozessen des DB2-Datenbanksystems verwendete Speicher verfügt möglicherweise über einen zugeordneten Paging-Bereich. Die Reservierung dieses Paging-Bereichs bleibt möglicherweise auch dann bestehen, wenn der zugehörige Speicher freigegeben wird. Ob dies der Fall ist, hängt von der (optimierbaren) Richtlinie für die Speicherzuordnung der Verwaltung des virtuellen Speichers des Betriebssystems ab. Die Registrierdatenbankvariable **DB2MEMDISCLAIM** steuert, ob DB2-Agenten explizit anfordern, dass das Betriebssystem die Zuordnung des reservierten Paging-Bereichs zu dem freigegebenen Speicher aufhebt.

  Wenn **DB2MEMDISCLAIM** auf YES gesetzt wird, sinkt der Bedarf an Paging-Bereich und möglicherweise auch die auf Seitenwechsel (Paging) zurückzuführende Plattenaktivität. Wenn **DB2MEMDISCLAIM** auf NO gesetzt wird, steigt der Bedarf an Paging-Bereich und möglicherweise auch die auf Seitenwechsel zurückzuführende Plattenaktivität. In einigen Fällen, in denen reichlich Paging-Bereich und so viel Realspeicher verfügbar ist, dass keine Seitenwechsel auftreten, bietet die Einstellung NO eine geringfügige Leistungssteigerung.

DB2MEMMAXFREE

- Betriebssystem: Alle
- Standardwert: NULL, Werte: 0 bis 2^{32}-1 Byte
- Gibt die maximale Anzahl von Byte an ungenutztem privaten Speicher an, der von Prozessen des DB2-Datenbanksystems zurückbehalten wird, bevor ungenutzter Speicher an das Betriebssystem zurückgegeben wird.

  Wenn **DB2MEMMAXFREE** nicht festgelegt wird, behalten Prozesse des DB2-Datenbanksystems bis zu 20 % des ungenutzten privaten Speichers (auf der Grundlage der Menge an privatem Speicher, die momentan belegt ist) zurück, bevor Speicher freigegeben und an das Betriebssystem zurückgegeben wird.

Anmerkung: DB2MEMMAXFREE ist veraltet und wird in einem zukünftigen Release entfernt. Diese Variable ist nicht mehr erforderlich, weil der Datenbankmanager jetzt mit einem Multithread-Modell der Steuer-

komponente arbeitet. Definieren Sie diese Variable nicht. Anderenfalls kann sich dies negativ auf die Leistung auswirken und zu unerwartetem Verhalten führen.

DB2_MEM_TUNING_RANGE

- Betriebssystem: AIX und Windows
- Standardwert: NULL, Werte: eine Folge von Prozentsätzen *n*, *m* mit *n=minfree* und *m=maxfree*
- Die Größe des physischen Speichers, der von der DB2-Instanz nicht beansprucht wird, spielt eine wichtige Rolle, weil durch sie festgelegt wird, wie viel Speicher andere Anwendungen, die auf derselben Maschine aktiv sind, nutzen können. Wenn die automatische Optimierung für den gemeinsam genutzten Speicher aktiviert ist, hängt die Größe des physischen Speichers, der von einer bestimmten Instanz nicht genutzt wird, vom Speicherbedarf der Instanz (und ihrer aktiven Datenbanken) ab. Wenn eine Instanz dringend zusätzlichen Speichers bedarf, ordnet sie Speicher zu, bis der freie physische Speicher auf dem System den Prozentsatz erreicht, der durch den Parameter *minfree* angegeben wird. Wenn die Instanz weniger Speicher bedarf, behält sie eine größere Kapazität an freiem physischen Speicher bei, die als Prozentsatz durch den Parameter *maxfree* angegeben wird. Daher ist es erforderlich, dass der für *minfree* definierte Wert kleiner als der Wert für *maxfree* ist.

  Wenn diese Variable nicht festgelegt wird, berechnet der DB2-Datenbankmanager Werte für *minfree* und *maxfree* auf der Basis der Speicherkapazität auf dem Server. Es wird empfohlen, diese Variable *nicht* festzulegen, sofern Sie nicht mit aktiviertem Self-Tuning Memory Manager (STMM) arbeiten, **database_memory** auf den Wert AUTOMATIC gesetzt haben und Probleme wegen unzureichenden freien physischen Speichers auftreten.

DB2_MMAP_READ

- Betriebssystem: AIX
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable wird in Verbindung mit **DB2_MMAP_WRITE** verwendet, damit das DB2-Datenbanksystem 'mmap' als alternative E/A-Methode verwenden kann.

  Werden diese Variablen auf ON gesetzt, umgehen Daten, die in die DB2-Pufferpools gelesen und aus diesen geschrieben werden, den AIX-Speichercache. Wenn der DB2-Pufferpool relativ klein ist und Sie diesen Pufferpool nicht vergrößern können oder wollen, sollten Sie die Nutzung des AIX-Speichercaches in Erwägung ziehen, indem Sie die Variablen **DB2_MMAP_READ** und **DB2_MMAP_WRITE** auf OFF setzen.

DB2_MMAP_WRITE

- Betriebssystem: AIX
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable wird in Verbindung mit **DB2_MMAP_READ** verwendet, damit das DB2-Datenbanksystem 'mmap' als alternative E/A-Methode verwenden kann.

  Werden diese Variablen auf ON gesetzt, umgehen Daten, die in die DB2-Pufferpools gelesen und aus diesen geschrieben werden, den AIX-Speichercache. Wenn der DB2-Pufferpool relativ klein ist und Sie diesen Pufferpool nicht vergrößern können oder wollen, sollten Sie die Nutzung

des AIX-Speichercaches in Erwägung ziehen, indem Sie die Variablen **DB2_MMAP_READ** und **DB2_MMAP_WRITE** auf OFF setzen.

DB2_NO_FORK_CHECK

- Betriebssystem: UNIX
- Standardwert: OFF, Werte: ON oder OFF
- Wenn diese Variable aktiviert ist, minimiert der DB2-Laufzeitclient Prüfungen zur Bestimmung, ob der aktuelle Prozess ein Ergebnis eines fork-Aufrufs ist. Dies kann die Leistung von DB2-Anwendungen erhöhen, die nicht mit der API fork() arbeiten.

Anmerkung: Diese Variable ist veraltet und wird in einem zukünftigen Release entfernt. Sie ist nicht erforderlich, weil die aktuelle Prozess-ID (pid) im Cache zwischengespeichert wird, wenn der Prozess gestartet oder neu verzweigt (fork) wird.

DB2NTMEMSIZE

- Betriebssystem: Windows
- Standardwert: (je nach Speichersegment unterschiedlich)
- Windows erfordert, dass alle gemeinsam genutzten Speichersegmente während der DLL-Initialisierung reserviert werden, um übereinstimmende Adressen in allen Prozessen zu gewährleisten. **DB2NTMEMSIZE** ermöglicht Benutzern das Überschreiben der DB2-Standardwerte unter Windows (falls erforderlich). In den meisten Situationen sollte der Standardwert ausreichen. Folgende Speichersegmente, Standardgrößen und Überschreibungsoptionen sind verfügbar:
  1. Parallele FCM-Puffer: Standardgröße ist 512 MB auf 32-Bit-Plattformen; 4,5 GB auf 64-Bit-Plattformen; Überschreibungsoption ist FCM:<anzahl_byte>.
  2. Kommunikation im abgeschirmten Modus: Standardgröße ist 80 MB auf 32-Bit-Plattformen; 512 MB auf 64-Bit-Plattformen; Überschreibungsoption ist APLD:<anzahl_byte>

  Sie können mehrere Segmente überschreiben, indem Sie die Überschreibungsoptionen durch ein Semikolon (;) voneinander trennen. Wenn Sie z. B. die FCM-Puffer in einer 32-Bit-Version von DB2 auf 1 GB und die abgeschirmten gespeicherten Prozeduren auf 256 MB begrenzen wollen, geben Sie Folgendes an:

```
db2set DB2NTMEMSIZE=FCM:1073741824;APLD:268435456
```

DB2NTNOCACHE

- Betriebssystem: Windows
- Standardwert: OFF, Werte: ON oder OFF
- Die Registrierdatenbankvariable **DB2NTNOCACHE** gibt an, ob das DB2-Datenbanksystem Datenbankdateien mit der Option NOCACHE öffnet. Wenn **DB2NTNOCACHE**=ON definiert ist, wird das Zwischenspeichern im Dateisystemcache inaktiviert. Wenn **DB2NTNOCACHE**=OFF definiert ist, speichert das Betriebssystem DB2-Dateien im Cache. Dies gilt für alle Daten außer für Dateien, die Langfelddaten oder LOB-Daten enthalten. Durch Inaktivieren der Cachefunktion des Betriebssystems steht mehr Speicher für die Datenbank zur Verfügung, sodass der Pufferpool oder der Sortierspeicher vergrößert werden kann.

  Unter Windows werden Dateien im Cache zwischengespeichert, sobald sie geöffnet werden. Dies ist das Standardverhalten. Für jedes GB in der

Datei wird aus einem Systempool ein MB reserviert. Verwenden Sie diese Registrierdatenbankvariable, um die nicht dokumentierte Begrenzung von 192 MB für den Cache außer Kraft zu setzen. Wenn die Cachebegrenzung erreicht ist, wird ein Fehler wegen nicht ausreichender Ressource ausgegeben.

Anmerkung: **DB2NTNOCACHE** ist seit Version 8.2 veraltet und wird in einem zukünftigen Release entfernt. Sie können denselben Zweck für Tabellenbereichscontainer durch die SQL-Anweisungen CREATE TABLESPACE und ALTER TABLESPACE erreichen.

DB2NTPRICLASS

- Betriebssystem: Windows
- Standardwert: NULL, Werte: R, H, (beliebiger anderer Wert)
- Stellt die Prioritätsklasse für die DB2-Instanz (Programm DB2SYSCS.EXE) ein. Es gibt drei Prioritätsklassen:
  - NORMAL_PRIORITY_CLASS (die Standardprioritätsklasse)
  - REALTIME_PRIORITY_CLASS (wird durch R definiert)
  - HIGH_PRIORITY_CLASS (wird durch H definiert)

  Diese Variable wird in Verbindung mit einzelnen Threadprioritäten (mit **DB2PRIORITIES** eingestellt) verwendet, um die absolute Priorität von DB2-Threads relativ zu anderen Threads im System zu bestimmen.

  Anmerkung: Die Variable **DB2NTPRICLASS** ist veraltet und sollte nur auf Empfehlung des Kundendienstes verwendet werden. Verwenden Sie DB2-Serviceklassen zur Anpassung der Agentenpriorität und der Vorableseprioritä t. Bei Verwendung dieser Variablen ist Sorgfalt geboten. Eine falsche Verwendung kann sich negativ auf die Gesamtleistung des Systems auswirken.

  Weitere Informationen finden Sie unter der API SetPriorityClass() in der Win32-Dokumentation.

DB2NTWORKSET

- Betriebssystem: Windows
- Standardwert: 1,1
- Dient zur Änderung der minimalen und der maximalen Größe des Arbeitsbereichs, der für den DB2-Datenbankmanager verfügbar ist. Standardmäßig kann der Arbeitsbereich eines Prozesses, wenn unter Windows keine Seitenwechsel (Paging) auftreten, nach Bedarf wachsen. Wenn es jedoch zum Paging kommt, liegt der maximale Arbeitsbereich, den ein Prozess haben kann, bei annähernd 1 MB. Mit **DB2NTWORKSET** kann dieser Standardwert überschrieben werden.

  Geben Sie **DB2NTWORKSET** in der Syntax **DB2NTWORKSET**=*min*, *max* an, wobei *min* und *max* in Megabyte angegeben werden.

DB2_OVERRIDE_BPF

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: eine positive Anzahl von Seiten ODER <eintrag>[;<eintrag>...] mit <eintrag>=<pufferpool-ID>,<anzahl der seiten>
- Diese Variable gibt die Größe des Pufferpools (in Seiten) an, der bei der Aktivierung der Datenbank, bei der aktualisierenden Recovery oder bei der Recovery nach einem Systemabsturz zu erstellen ist. Diese Variable ist nützlich, wenn es während der Datenbankaktivierung, der aktualisie-

renden Recovery oder der Recovery nach einem Systemabsturz aufgrund begrenzter Speicherkapazitäten zu Fehlern kommt. Die Speicherknappheit kann infolge des seltenen Falls eines Mangels an Realspeicher auftreten oder durch den Versuch seitens des Datenbankmanagers verursacht werden, einen großen Pufferpool für nicht korrekt konfigurierte Pufferpools zuzuordnen. Wenn z. B. selbst ein Pufferpool der Minimalgröße von 16 Seiten nicht vom Datenbankmanager bereitgestellt werden kann, haben Sie die Möglichkeit, mithilfe dieser Umgebungsvariablen eine kleinere Anzahl von Seiten anzugeben. Der für diese Variable angegebene Wert überschreibt die aktuelle Pufferpoolgröße.

Sie können auch <eintrag>[;<eintrag>...] (mit <eintrag>=<pufferpool-ID>,<anzahl der seiten>) angeben, um die Größe aller oder einer Teilmenge der Pufferpools temporär zu ändern, sodass sie gestartet werden können.

DB2_PINNED_BP

- Betriebssystem: AIX, HP-UX, Linux
- Standardwert: NO, Werte: YES oder NO
- Die Einstellung dieser Variablen auf den Wert YES bewirkt, dass DB2 anfordert, dass das Betriebssystem den gemeinsam genutzten Datenbankspeicher von DB2 im Hauptspeicher fixiert und dort belässt. Wenn DB2 so konfiguriert wird, dass der gemeinsam genutzte Datenbankspeicher im Hauptspeicher behalten wird, muss sorgfältig sichergestellt werden, dass das System nicht überlastet wird, da das Betriebssystem die Speicherkapazitäten nun weniger flexibel verwalten kann.

  Unter Linux ist neben einer Modifizierung dieser Registrierdatenbankvariablen auch die Bibliothek libcap.so.1 erforderlich.

  Wenn diese Variable auf YES gesetzt wird, kann die automatische Optimierung für den gemeinsam genutzten Datenbankspeicher (aktiviert durch die Einstellung AUTOMATIC für den Konfigurationsparameter **database_memory**) nicht aktiviert werden.

  Wenn Sie unter AIX-Betriebssystemen das Fixieren von Datenbankspeicher mit der Unterstützung mittlerer Seitengrößen nutzen möchten (die Standardfunktionsweise), müssen Sie sicherstellen, dass der Instanzeigner über die Funktionen CAP_BYPASS_RAC_VMM und CAP_PROPAGATE verfügt, indem Sie sich mit Rootberechtigung anmelden und den folgenden Befehl eingeben:

  ```
  chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE <benutzer-id_des_
  instanzeigners>
  ```

  Für HP-UX in einer 64-Bit-Umgebung müssen Sie zusätzlich zur Änderung dieser Registrierdatenbankvariablen der DB2-Instanzgruppe das Zugriffrecht MLOCK erteilen. Dazu führt ein Benutzer mit Rootberechtigung die folgenden Aktionen aus:

  1. Er fügt die DB2-Instanzgruppe der Datei /etc/privgroup hinzu. Wenn die DB2-Instanzgruppe beispielsweise zur Gruppe `db2iadm1` gehört, muss der Datei /etc/privgroup die folgende Zeile hinzugefügt werden:

     ```
     db2iadm1 MLOCK
     ```

  2. Er führt Sie den folgenden Befehl aus:

     ```
     setprivgrp -f /etc/privgroup
     ```

DB2PRIORITIES

- Betriebssystem: Alle
- Einstellung der Werte von der Plattform abhängig
- Steuert die Prioritäten von DB2-Prozessen und -Threads.

Anmerkung: Die Variable **DB2PRIORITIES** ist veraltet und sollte nur auf Empfehlung des Kundendienstes verwendet werden. Verwenden Sie DB2-Serviceklassen zur Anpassung der Agentenpriorität und der Vorableseprioritä t.

DB2_RCT_FEATURES

- Betriebssystem: Alle
- Standardwert: NULL. Werte: `GROUPUPDATE=[ON|OFF]`. Der Standardwert für `GROUPUPDATE` ist OFF.
- Diese Variable ermöglicht eine optimierte und reduzierte Aktualisierungsverarbeitung für eine UPDATE-Anweisung mit Suche, die mehrere Zeilen in einer Bereichsclustertabelle zum Ziel hat, wenn nur Gleichheitsvergleichselemente für die führenden und eine Teilmenge der Schlüsselsortierfolgespalten angegeben werden. Die Protokollierung wird ebenfalls verringert, weil nur ein Protokollsatz für alle Zeilen, die auf einer Seite aktualisiert werden, anstatt eines Protokollsatzes für jede einzelne aktualisierte Zeile aufgezeichnet wird.

  Syntax:

  ```
  db2set DB2_RCT_FEATURES=GROUPUPDATE=ON
  ```

  Anmerkung: Diese Registrierdatenbankvariable ist in Version 9.5 Fixpack 3 verfügbar. Es ist nicht möglich, eine Rückmigration auf ältere Fixpacks auszuführen, wenn diese Registrierdatenbankvariable einmal aktiviert wurde. Außerdem wird die Gruppenaktualisierung nicht ausgeführt, wenn die Zieltabelle der UPDATE-Anweisung die Klausel DATA CAPTURE CHANGES, Trigger oder Spalten variabler Länge verwendet oder Sekundärindizes besitzt oder wenn die Aktualisierung Primär- oder Fremdschlüsselspalten ändert.

DB2_RESOURCE_POLICY

- Betriebssystem: AIX 5 oder höher, alle Linux-Versionen außer zSeries (32-Bit), Windows Server 2003 oder höher
- Standardwert: nicht definiert, Werte: gültiger Pfad zur Konfigurationsdatei
- Definiert eine Ressourcenrichtlinie, mit deren Hilfe die von der DB2-Datenbank genutzten Betriebssystemressourcen begrenzt werden können, oder sie enthält Regeln für die Zuordnung bestimmter Betriebssystemressourcen zu bestimmten DB2-Datenbankobjekten. Unter AIX, Linux oder Windows kann diese Registrierdatenbankvariable zum Beispiel zur Begrenzung der vom DB2-Datenbanksystem genutzten Gruppe von Prozessoren verwendet werden. Der Speicherbereich der Ressourcensteuerung variiert in Abhängigkeit vom Betriebssystem.

  Auf Maschinen unter AIX und Linux mit aktiviertem NUMA kann eine Richtlinie definiert werden, die angibt, welche Ressourcengruppen das DB2-Datenbanksystem verwendet. Wenn die Ressourcengruppenbindung verwendet wird, wird jeder einzelne DB2-Prozess an eine bestimmte Ressourcengruppe gebunden. Dies kann in einigen Leistungsoptimierungsszenarios von Vorteil sein.

  Sie können die Registrierdatenbankvariable festlegen, um den Pfad zu einer Konfigurationsdatei anzugeben, die eine Richtlinie für das Binden von DB2-Prozessen an Betriebssystemressourcen definiert. Die

Ressourcenrichtlinie ermöglicht Ihnen die Angabe einer Gruppe von Betriebssystemressourcen, auf die das DB2-Datenbanksystem zu beschränken ist. Jeder DB2-Prozess wird an eine einzige Ressource der Gruppe gebunden. Die Ressourcenzuordnung erfolgt in einem zirkulären Umlauf.

Beispiele für Konfigurationsdateien:

Beispiel 1: Binden aller DB2-Prozesse entweder an CPU 1 oder an CPU 3.

```
<RESOURCE_POLICY>
  <GLOBAL_RESOURCE_POLICY>
   <METHOD>CPU</METHOD>
   <RESOURCE_BINDING>
    <RESOURCE>1</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>3</RESOURCE>
   </RESOURCE_BINDING>
  </GLOBAL_RESOURCE_POLICY>
 </RESOURCE_POLICY>
```

Beispiel 2 (nur AIX): Binden von DB2-Prozessen an eine der folgenden Ressourcengruppen: sys/node.03.00000, sys/node.03.00001, sys/node.03.00002, sys/node.03.00003.

```
<RESOURCE_POLICY>
  <GLOBAL_RESOURCE_POLICY>
   <METHOD>RSET</METHOD>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00000</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00001</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00002</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00003</RESOURCE>
   </RESOURCE_BINDING>
  </GLOBAL_RESOURCE_POLICY>
 </RESOURCE_POLICY>
```

Anmerkung: (Nur AIX:) Die Verwendung der Methode RSET setzt die CAP_NUMA_ATTACH-Funktionalität voraus.

Beispiel 3 (nur Linux): Binden des gesamten Speichers aus den Pufferpool-IDs 2 und 3, die der Datenbank SAMPLE zugeordnet sind, an NUMA-Knoten 3. Darüber hinaus: Verwenden von 80 % des gesamten Datenbankspeichers für das Binden an NUMA-Knoten 3 sowie von 20 % zum einheitenübergreifenden Verteilen (Striping) über alle Knoten für nicht pufferpoolspezifischen Speicher:

```
<RESOURCE_POLICY>
  <DATABASE_RESOURCE_POLICY>
   <DBNAME>sample</DBNAME>
   <METHOD>NODEMASK</METHOD>
   <RESOURCE_BINDING>
    <RESOURCE>3</RESOURCE>
    <DBMEM_PERCENTAGE>80</DBMEM_PERCENTAGE>
    <BUFFERPOOL_BINDING>
     <BUFFERPOOL_ID>2</BUFFERPOOL_ID>
     <BUFFERPOOL_ID>3</BUFFERPOOL_ID>
```

```
    </BUFFERPOOL_BINDING>
   </RESOURCE_BINDING>
  </DATABASE_RESOURCE_POLICY>
 </RESOURCE_POLICY>
```

Beispiel 4 (nur für Linux und Windows): Definieren zweier voneinander unabhängiger Prozessorgruppen (Processor Sets), die durch die CPU-Masken 0x0F und 0xF0 angegeben werden. Binden von DB2-Prozessen und Pufferpool-ID 2 an die Prozessergruppe 0x0F und von DB2-Prozessen und Pufferpool-ID 3 an die Prozessorgruppe 0xF0. Verwenden von 50 % des gesamten Datenbankspeichers für die Bindung jeder Prozessorgruppe.

Diese Ressourcenrichtlinie ist nützlich, wenn eine Zuordnung zwischen Prozessoren und NUMA-Knoten gewünscht wird. Ein Beispiel für solch ein Szenario ist ein System mit 8 Prozessoren und 2 NUMA-Knoten, bei dem die Prozessoren 0 bis 3 zum NUMA-Knoten 0 und die Prozessoren 4 bis 7 zum NUMA-Knoten 1 gehören. Diese Ressourcenrichtlinie ermöglicht eine Prozessorbindung, indem sie gleichzeitig die Speicherlokalität (d. h. einen Hybrid der CPU-Methode und der NODEMASK-Methode) beibehält.

```
<RESOURCE_POLICY>
  <DATABASE_RESOURCE_POLICY>
    <DBNAME>sample</DBNAME>
    <METHOD>CPUMASK</METHOD>
    <RESOURCE_BINDING>
      <RESOURCE>0x0F</RESOURCE>
      <DBMEM_PERCENTAGE>50</DBMEM_PERCENTAGE>
      <BUFFERPOOL_BINDING>
        <BUFFERPOOL_ID>2</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
    </RESOURCE_BINDING>
      <RESOURCE>0x0F</RESOURCE>
      <DBMEM_PERCENTAGE>50</DBMEM_PERCENTAGE>
      <BUFFERPOOL_BINDING>
        <BUFFERPOOL_ID>3</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
    </RESOURCE_BINDING>
  </DATABASE_RESOURCE_POLICY>
</RESOURCE_POLICY>
```

Anmerkung: Die Verwendung der Methode RSET setzt die CAP_NUMA_ATTACH-Funktionalität voraus und wird unter Linux nicht unterstützt.

Die in der Registrierdatenbankvariablen **DB2_RESOURCE_POLICY** angegebene Konfigurationsdatei akzeptiert ein SCHEDULING_POLICY-Element. Über das SCHEDULING_POLICY-Element können Sie auf einigen Plattformen Folgendes auswählen:

– Die Planungsrichtlinie des Betriebssystems, die vom DB2-Server verwendet wird

  Sie können eine Betriebssystemplanungsrichtlinie für DB2 unter AIX und für DB2 unter Windows über die Registrierdatenbankvariable **DB2NTPRICLASS** definieren.
– Die Betriebssystemprioritäten, die von einzelnen DB2-Serveragenten verwendet werden

Alternativ können Sie zur Steuerung der Betriebssystemplanungsrichtlinie und zur Festlegung der DB2-Agentenprioritäten die Registrierdatenbankvariablen **DB2PRIORITIES** und **DB2NTPRICLASS** verwenden. Die Spezifikation eines SCHEDULING_POLICY-Elements in der Konfigurationsdatei für Ressourcenrichtlinien stellt eine gemeinsame

Position zur Angabe sowohl der Planungsrichtlinie als auch der zugeordneten Agentenprioritäten zur Verfügung.

Beispiel 1: Auswahl der AIX-Planungsrichtlinie SCHED_FIFO2 mit Zuordnung einer höheren Priorität für die Prozesse zum Schreiben und Lesen des DB2-Protokolls.

```
<RESOURCE_POLICY>
 <SCHEDULING_POLICY>
  <POLICY_TYPE>SCHED_FIFO2</POLICY_TYPE>
  <PRIORITY_VALUE>60</PRIORITY_VALUE>

  <EDU_PRIORITY>
   <EDU_NAME>db2loggr</EDU_NAME>
   <PRIORITY_VALUE>56</PRIORITY_VALUE>
  </EDU_PRIORITY>

  <EDU_PRIORITY>
   <EDU_NAME>db2loggw</EDU_NAME>
   <PRIORITY_VALUE>56</PRIORITY_VALUE>
  </EDU_PRIORITY>
 </SCHEDULING_POLICY>
</RESOURCE_POLICY>
```

Beispiel 2: Ersatz für DB2NTPRICLASS=H unter Windows.

```
<RESOURCE_POLICY>
 <SCHEDULING_POLICY>
  <POLICY_TYPE>HIGH_PRIORITY_CLASS</POLICY_TYPE>
 </SCHEDULING_POLICY>
</RESOURCE_POLICY>
```

DB2_SET_MAX_CONTAINER_SIZE

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte -1, beliebige positive ganze Zahl größer als 65.536 Byte
- Mit dieser Registrierdatenbankvariablen können Sie die Größe einzelner Container für Tabellenbereiche mit dynamischem Speicher bei aktivierter automatischer Größenänderungsfunktion (AutoResize) begrenzen.

  Anmerkung: Obwohl Sie **DB2_SET_MAX_CONTAINER_SIZE** in Byte, Kilobyte oder Megabyte angeben können, zeigt der Befehl db2set den Wert in Byte an.
- Durch den Wert -1 wird keine Begrenzung für die Größe eines Containers festgelegt.

DB2_SKIPDELETED

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Wenn diese Variable aktiviert ist, ermöglicht sie Anweisungen mit den Isolationsstufen Cursorstabilität oder Lesestabilität, gelöschte Schlüssel bei einem Indexzugriff und gelöschte Zeilen bei einem Tabellenzugriff bedingungslos zu überspringen. Wenn **DB2_EVALUNCOMMITTED** aktiviert ist, werden gelöschte Zeilen automatisch übersprungen. Nicht festgeschriebene Pseudolöschungen von Schlüsseln in Indizes des Typs 2 werden jedoch nur übersprungen, wenn auch **DB2_SKIPDELETED** aktiviert ist.

  Diese Registrierdatenbankvariable wirkt sich nicht auf das Verhalten von Cursorn in den DB2-Katalogtabellen aus.

  Diese Registrierdatenbankvariable wird mit dem Befehl db2start aktiviert.

DB2_SKIPINSERTED

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Wenn die Registrierdatenbankvariable **DB2_SKIPINSERTED** aktiviert ist, können Anweisungen mit der Isolationsstufe Cursorstabilität oder Lesestabilität nicht festgeschriebene eingefügte Zeilen so überspringen, als wären sie nicht eingefügt worden. Diese Registrierdatenbankvariable wirkt sich nicht auf das Verhalten von Cursorn in den DB2-Katalogtabellen aus. Diese Registrierdatenbankvariable wird beim Datenbankstart aktiviert. Die Entscheidung darüber, ob nicht festgeschriebene eingefügte Zeilen zu überspringen sind, wird beim Kompilieren oder Binden getroffen.

  Anmerkung: Das Überspringen eingefügter Zeilen ist nicht mit Tabellen kompatibel, für die eine Rolloutbereinigung ansteht. Infolgedessen ist es möglich, dass Suchoperationen auf Sperren für eine Satz-ID (RID) warten und anschließend lediglich feststellen, dass die Satz-ID zu dem durch den Rollout entfernten Block gehört.

DB2_SMS_TRUNC_TMPTABLE_THRESH

- Betriebssystem: Alle
- Standardwert: 0, Werte: -1, 0-*n*, wobei *n* = Anzahl der Speicherbereiche pro temporärer Tabelle im SMS-Tabellenbereichscontainer ist, die zu verwalten sind
- Diese Variable gibt einen minimalen Schwellenwert für die Dateigröße an, mit der die Datei, die eine temporäre Tabelle darstellt, in SMS-Tabellenbereichen verwaltet wird.

  Diese Variable wird standardmäßig auf den Wert 0 gesetzt, d. h. es wird keine spezielle Schwellenwertbehandlung ausgeführt. Stattdessen wird die Datei auf die Größe 0 abgeschnitten, wenn die Tabelle nicht mehr benötigt wird. Wenn der Wert dieser Variablen größer als 0 ist, wird eine größere Datei behalten. Dadurch kann der Systemaufwand verringert werden, der mit dem Löschen und erneuten Erstellen der Datei bei jeder Verwendung einer temporären Tabelle verbunden ist.

  Wenn diese Variable auf den Wert -1 gesetzt wird, wird die Datei nicht abgeschnitten und kann uneingeschränkt an Größe zunehmen, soweit die Systemressourcen dies zulassen.

DB2_SORT_AFTER_TQ

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO.
- Gibt an, wie das Optimierungsprogramm mit gezielt übertragenen Tabellenwarteschlangen in einer Umgebung mit partitionierten Datenbanken arbeitet, wenn die Daten für den Empfänger sortiert sein müssen und die Anzahl der Empfängerknoten der Anzahl der Senderknoten entspricht.

  Wenn **DB2_SORT_AFTER_TQ**=NO ist, sortiert das Optimierungsprogramm Zeilen in der Regel auf der sendenden Seite und mischt die Zeilen auf der empfangenden Seite zusammen.

  Wenn **DB2_SORT_AFTER_TQ**=YES ist, überträgt das Optimierungsprogramm in der Regel die Zeilen unsortiert, mischt sie auf der empfangenden Seite nicht zusammen, sondern sortiert die Zeilen nach Empfang aller Zeilen auf der empfangenden Seite.

DB2_SELUDI_COMM_BUFFER

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable wird bei der Verarbeitung von Blockcursorn über Abfragen mit SELECT from UPDATE/INSERT/DELETE (UDI) verwendet. Wenn aktiviert, verhindert diese Registrierdatenbankvariable, dass das Ergebnis einer Abfrage in einer temporären Tabelle gespeichert wird. Stattdessen versucht das DB2-Datenbanksystem bei der OPEN-Verarbeitung eines Blockcursors für eine Abfrage SELECT from UDI, das gesamte Ergebnis der Abfrage direkt im Speicherbereich des Kommunikationspuffers zu puffern.

  Anmerkung: Wenn der Kommunikationspufferbereich für das gesamte Ergebnis der Abfrage nicht ausreicht, wird der Fehler SQLCODE -906 ausgegeben und die Transaktion mit ROLLBACK rückgängig gemacht. Informationen zur Anpassung der Größe des Kommunikationspufferbereichs für lokale und ferne Anwendungen finden Sie in den Beschreibungen der Konfigurationsparameter **aslheapsz** bzw. **rqrioblk** des Datenbankmanagers.

  Diese Registrierdatenbankvariable wird in Umgebungen mit partitionierten Datenbanken oder bei aktivierter partitionsinterner Parallelität nicht unterstützt.

DB2_TRUSTED_BINDIN

- Betriebssystem: Alle
- Standardwert: OFF, Werte: OFF, ON oder CHECK
- Wenn die Variable **DB2_TRUSTED_BINDIN** aktiviert ist, beschleunigt sie die Ausführung von Abfrageanweisungen, die Hostvariablen innerhalb einer eingebetteten, nicht abgeschirmten gespeicherten Prozedur enthalten.

  Wenn diese Variable aktiviert ist, findet keine Konvertierung vom externen SQLDA-Format in ein internes DB2-Format beim Binden von SQL- und XQuery-Anweisungen statt, die in einer eingebetteten, nicht abgeschirmten gespeicherten Prozedur enthalten sind. Dies führt zu einer beschleunigten Verarbeitung eingebetteter SQL- und XQuery-Anweisungen.

  Die folgenden Datentypen werden in eingebetteten, nicht abgeschirmten gespeicherten Prozeduren nicht unterstützt, wenn diese Variable aktiviert ist:
  - SQL_TYP_DATE
  - SQL_TYP_TIME
  - SQL_TYP_STAMP
  - SQL_TYP_CGSTR
  - SQL_TYP_BLOB
  - SQL_TYP_CLOB
  - SQL_TYP_DBCLOB
  - SQL_TYP_CSTR
  - SQL_TYP_LSTR
  - SQL_TYP_BLOB_LOCATOR
  - SQL_TYP_CLOB_LOCATOR
  - SQL_TYP_DCLOB_LOCATOR
  - SQL_TYP_BLOB_FILE
  - SQL_TYP_CLOB_FILE
  - SQL_TYP_DCLOB_FILE

– SQL_TYP_BLOB_FILE_OBSOLETE
– SQL_TYP_CLOB_FILE_OBSOLETE
– SQL_TYP_DCLOB_FILE_OBSOLETE

Wenn diese Datentypen auftreten, wird ein Fehler mit dem SQLCODE-Wert -804 und dem SQLSTATE-Wert 07002 zurückgegeben.

Anmerkung: Der Datentyp und die Länge der Eingabehostvariablen muss exakt mit dem internen Datentyp und der Länge des entsprechenden Elements übereinstimmen. Für Hostvariablen wird diese Voraussetzung immer erfüllt. Bei Parametermarken muss jedoch sorgfältig darauf geachtet werden, dass übereinstimmende Datentypen verwendet werden. Zur Sicherstellung, dass die Datentypen und Längen für alle Eingabehostvariablen übereinstimmen, kann die Option CHECK verwendet werden. Allerdings macht diese Option die meisten Leistungsvorteile wieder zunichte.

Anmerkung: DB2_TRUSTED_BINDIN ist veraltet und wird in einem späteren Release entfernt.

DB2_USE_ALTERNATE_PAGE_CLEANING

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: ON oder OFF
- Diese Variable gibt an, ob eine DB2-Datenbank die alternative Methode der Seitenlöschalgorithmen oder die Standardmethode der Seitenlöschfunktion verwendet. Wenn diese Variable auf den Wert ON gesetzt ist, verwendet das DB2-System eine Seitenlöschmethode, die geänderte Seiten auf die Platte schreibt, dem Wert von LSN_GAP voraus bleibt und proaktiv Seiten ermittelt, die bereinigt werden können. Dadurch können die Seitenlöschfunktionen die verfügbare E/A-Bandbreite der Platte besser nutzen. Wenn diese Variable auf den Wert ON gesetzt ist, ist der Datenbankkonfigurationsparameter **chngpgs_thresh** nicht länger relevant, weil er keinen Einfluss auf die Aktivitäten der Seitenlöschfunktionen hat.

DB2_USE_IOCP

- Betriebssystem: AIX 5.3 TL9
- Standardwert: OFF Werte: OFF oder ON
- Konfigurieren Sie `IOCP` (E/A-Ausführungsports), bevor Sie diese Registrierdatenbankvariable aktivieren.
- Diese Variable ermöglicht die Verwendung von AIX-E/A-Ausführungsports (IOCP - I/O Completion Ports) bei der Übergabe und Erfassung asynchroner E/A-Anforderungen (AIO-Anforderungen). Diese Funktion dient zur Erhöhung der Leistung in einer NUMA-Umgebung (NUMA - Non-Uniform Memory Access, nicht gleichmäßiger Speicherzugriff), indem der ferne Speicherzugriff vermieden wird. Diese Leistungsvariable steht in Version 9.5 Fixpack 3 zur Verfügung.

Verschiedene Variablen

DB2ADMINSERVER

- Betriebssystem: Windows und UNIX
- Standardwert: NULL
- Gibt den DB2-Verwaltungsserver (DAS, DB2 Administration Server) an.

DB2_ATS_ENABLE

- Betriebssystem: Alle
- Standardwert: NULL, Werte: YES/TRUE/ON/1 oder NO/FALSE/OFF/0
- Diese Variable steuert, ob der Scheduler für Verwaltungstasks (ATS - Administrative Task Scheduler) ausgeführt wird. Der Scheduler für Verwaltungstasks ist standardmäßig inaktiviert. Wenn der Scheduler inaktiviert ist, können Sie die integrierten Prozeduren und Sichten zum Definieren und Ändern von Tasks verwenden, jedoch führt der Scheduler die Tasks nicht aus.

DB2AUTH

- Betriebssystem: Alle
- Werte: TRUSTEDCLIENT_SRVRENC, TRUSTEDCLIENT_DATAENC, DISABLE_CHGPASS
- Mit dieser Variablen können Sie die Funktionsweise der Benutzerauthentifizierung optimieren.
  - TRUSTEDCLIENT_SRVRENC: Durch diesen Wert werden ungesicherte Clients dazu gezwungen, SERVER_ENCRYPT zu verwenden.
  - TRUSTEDCLIENT_DATAENC: Durch diesen Wert werden ungesicherte Clients dazu gezwungen, DATA_ENCRYPT zu verwenden.
  - DISABLE_CHGPASS: Durch diesen Wert wird die Möglichkeit, das Kennwort vom Client aus zu ändern, inaktiviert.
  - OSAUTHDB: Ab DB2 Version 9.5 Fixpack 4 wird DB2 durch diesen Wert angewiesen, die Authentifizierungs- und Gruppeneinstellung für einen Benutzer unter dem Betriebssystem AIX zu verwenden. Indem Sie **DB2AUTH** auf den Wert OSAUTHDB setzen, können Sie DB2 so konfigurieren, dass die Authentifizierung von Benutzern und die Anforderung der Gruppen dieser Benutzer durch das Betriebssystem erfolgt. Wenn die Einstellungen des Betriebssystems AIX für die Authentifizierungs- und Gruppeninformationen eines Benutzers zur Verwendung von LDAP konfiguriert sind, führt das Betriebssystem AIX die Authentifizierung wiederum durch einen LDAP-Server aus. Bei dem LDAP-Server kann es sich um einen beliebigen der folgenden handeln:
    - IBM Tivoli Directory Server (ITDS)
    - Microsoft Active Directory (MSAD)
    - Sun One Directory Server

DB2CLIINIPATH

- Betriebssystem: Alle
- Standardwert: NULL
- Dient zum Überschreiben des Standardpfads der DB2-CLI/ODBC-Konfigurationsdatei (db2cli.ini) und zum Angeben einer alternativen Speicherposition auf dem Client. Der definierte Wert muss ein gültiger Pfad auf dem Clientsystem sein.

DB2_COMMIT_ON_EXIT

- Betriebssystem: UNIX
- Standardwert: OFF, Werte: OFF/NO/0 oder ON/YES/1
- Unter UNIX-Betriebssystemen vor DB2 UDB Version 8 schrieb DB2 alle verbleibenden unvollständigen Transaktionen bei erfolgreicher Beendigung der Anweisung fest. In DB2 UDB Version 8 wurde diese

Funktionsweise geändert, sodass unvollständige Transaktionen bei Beendigung mit ROLLBACK rückgängig gemacht wurden. Diese Registrierdatenbankvariable gibt Benutzern von Anwendungen mit eingebettetem SQL, die von der früheren Funktionsweise abhängig sind, die Möglichkeit, diese Funktionsweise in Version 9 weiterhin zu aktivieren. Diese Registrierdatenbankvariable hat keine Auswirkungen auf JDBC-, CLI- und ODBC-Anwendungen.

Beachten Sie, dass diese Registrierdatenbankvariable veraltet ist und dass die Funktionsweise mit Commit bei Beendigung in zukünftigen Releases nicht weiter unterstützt wird. Benutzer sollten bestimmen, ob ihre Anwendungen, die vor Version 9 entwickelt wurden, weiterhin von dieser Funktionsweise abhängig sind, und die entsprechenden expliziten COMMIT- bzw. ROLLBACK-Anweisungen nach Bedarf in die Anwendungen einfügen. Wenn die Registrierdatenbankvariable aktiviert ist, sollte sorgsam vermieden werden, neue Anwendungen zu implementieren, die keine expliziten COMMIT-Anweisungen vor ihrer Beendigung ausführen.

Die meisten Benutzer sollten die Standardeinstellung dieser Registrierdatenbankvariablen belassen.

DB2_CREATE_DB_ON_PATH

- Betriebssystem: Windows
- Standardwert: NULL, Werte: YES oder NO
- Setzen Sie diese Registrierdatenbankvariable auf den Wert YES, um die Unterstützung für die Verwendung eines Pfads (und eines Laufwerks) als Datenbankpfad zu aktivieren. Die Einstellung der Variablen **DB2_CREATE_DB_ON_PATH** wird überprüft, wenn eine Datenbank erstellt wird, wenn der Konfigurationsparameter **dftdbpath** des Datenbankmanagers definiert wird und wenn ein Restore einer Datenbank durchgeführt wird. Der vollständig qualifizierte Datenbankpfad kann bis zu 215 Zeichen lang sein.

  Wenn **DB2_CREATE_DB_ON_PATH** nicht definiert wird (oder auf den Wert NO gesetzt wird) und Sie beim Erstellen oder beim Restore einer Datenbank einen Pfad für den Datenbankpfad angeben, wird der Fehler SQL1052N zurückgegeben.

  Wenn **DB2_CREATE_DB_ON_PATH** nicht definiert wird (oder auf den Wert NO gesetzt wird) und Sie den Konfigurationsparameter **dftdbpath** der Datenbank aktualisieren, wird der Fehler SQL5136N zurückgegeben.

  Achtung:
  Wenn die Pfadunterstützung zur Erstellung neuer Datenbanken verwendet wird, ist es möglich, dass vor Version 9.1 geschriebene Anwendungen, die die API db2DbDirGetNextEntry() oder eine ältere Version dieser API verwenden, nicht korrekt funktionieren. Detaillierte Informationen zu verschiedenen Szenarios sowie zur geeigneten Vorgehensweise finden Sie unter http://www.ibm.com/software/data/db2/support/db2_9/.

DB2DEFPREP

- Betriebssystem: Alle
- Standardwert: NO, Werte: ALL, YES oder NO
- Simuliert das Laufzeitverhalten der Vorkompilierungsoption DEFERRED_PREPARE für Anwendungen, die vor der Verfügbarkeit dieser Option vorkompiliert wurden. Wenn zum Beispiel eine Anwendung von DB2 Version 2.1.1 (oder später) in einer Umgebung von DB2 Version

2.1.2 (oder später) ausgeführt würde, könnte der Parameter **DB2DEFPREP** verwendet werden, um das gewünschte Verhalten einer verzögerten Vorbereitung („Deferred Prepare") anzugeben.

Anmerkung: DB2DEFPREP ist veraltet und wird in einem zukünftigen Release entfernt. Diese Variable wird nur von Benutzern benötigt, die mit alten Versionen von DB2 arbeiten, in denen die Vorkompilierungsoption DEFERRED_PREPARE nicht verfügbar ist.

DB2_DISABLE_FLUSH_LOG

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Gibt an, ob das Schließen der aktiven Protokolldatei nach Abschluss eines Online-Backups inaktiviert werden soll.

  Wenn ein Online-Backup abgeschlossen ist, wird die letzte aktive Protokolldatei abgeschnitten, geschlossen und zur Archivierung verfügbar gemacht. Hierdurch wird sichergestellt, dass Ihrem Online-Backup alle zur Recovery benötigten archivierten Protokolldateien zur Verfügung stehen. Sie möchten möglicherweise das Schließen der letzten aktiven Protokolldatei inaktivieren, wenn Sie Sorge haben, dass Sie Teile des Protokollfolgenummernbereichs (LSN) verschwenden. Jedes Mal, wenn eine aktive Protokolldatei abgeschnitten wird, wird die Protokollfolgenummer um einem zum abgeschnittenen Speicherplatz proportionalen Betrag erhöht. Wenn Sie jeden Tag eine große Anzahl von Online-Backups durchführen, ist es vielleicht sinnvoll, das Schließen der letzten aktiven Protokolldatei zu inaktivieren.

  Darüber hinaus kann es sinnvoll sein, das Schließen der letzten aktiven Protokolldatei zu inaktivieren, wenn Sie feststellen, dass Sie Nachrichten über ein volles Protokoll kurz nach Abschluss des Online-Backups empfangen. Wenn eine Protokolldatei abgeschnitten wird, wird der reservierte Speicherbereich für aktive Protokolle um einen Betrag vergrößert, der proportional zur Größe des abgeschnittenen Protokolls ist. Der Speicherbereich für aktive Protokolle wird freigegeben, wenn die abgeschnittene Protokolldatei wieder zur Verwendung verfügbar gemacht wird. Die Wiederverfügbarmachung erfolgt kurz nach dem Zeitpunkt, zu dem die Protokolldatei inaktiv wird. Während des kurzen Intervalls zwischen diesen beiden Ereignissen empfangen Sie möglicherweise Nachrichten über ein volles Protokoll.

  Bei jedem Backup, das Protokolle mit einbezieht, wird diese Registrierdatenbankvariable ignoriert, da die aktive Protokolldatei abgeschnitten und geschlossen werden muss, um die Protokolle mit in das Backup einzubeziehen.

DB2CONNECT_DISCONNECT_ON_INTERRUPT

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES/TRUE/1 oder NO/FALSE/0
- Wenn diese Variable auf den Wert YES (TRUE oder 1) gesetzt ist, gibt sie an, dass die Verbindung zu einem z/OS-Server mit DB2 Universal Database der Version 8 (oder einer höheren Version) sofort abzubrechen ist, wenn ein Interrupt auftritt. Sie können diese Variable in den folgenden Konfigurationen verwenden:

– Wenn Sie einen DB2-Client mit einem z/OS-Server mit DB2 UDB der Version 8 (oder einer höheren Version) ausführen, setzen Sie die Variable **DB2CONNECT_DISCONNECT_ON_INTERRUPT** auf dem Client auf den Wert YES.
– Wenn Sie einen DB2-Client über ein DB2 Connect-Gateway mit einem z/OS-Server mit DB2 UDB der Version 8 (oder einer höheren Version) ausführen, setzen Sie die Variable **DB2CONNECT_DISCONNECT_ON_INTERRUPT** auf dem Gateway auf den Wert YES.

DB2_DISPATCHER_PEEKTIMEOUT

- Betriebssystem: Alle
- Standardwert: 1, Werte: 0 bis 32767 Sekunden; 0 gibt an, dass das Zeitlimit sofort abläuft
- Mit **DB2_DISPATCHER_PEEKTIMEOUT** können Sie die Zeit (in Sekunden) anpassen, die eine Zuteilerroutine (Dispatcher) auf die Verbindungsanforderung eines Clients wartet, bevor sie den Client an einen Agenten weitergibt. In den meisten Fällen sollte eine Anpassung dieser Registrierdatenbankvariablen nicht nötig sein. Diese Registrierdatenbankvariable wirkt sich nur auf Instanzen aus, für die der DB2 Connect-Verbindungskonzentrator aktiviert ist.

  Diese Registrierdatenbankvariable und die Registrierdatenbankvariable **DB2_SERVER_CONTIMEOUT** dienen beide zur Konfiguration der Behandlung eines neuen Clients während der Verbindungszeit. Wenn viele langsame Clients eine Verbindung zu einer Instanz herstellen, wird die Zuteilerroutine möglicherweise bis zu 1 Sekunde aufgehalten, um das Zeitlimit jedes Clients abzuwarten. Auf diese Weise wird die Zuteilerroutine zu einem Engpass, wenn viele Clients gleichzeitig eine Verbindung herstellen. Wenn in einer Instanz mit mehreren aktiven Datenbanken sehr langsame Verbindungszeiten festgestellt werden, kann **DB2_DISPATCHER_PEEKTIMEOUT** auf den Wert 0 herabgesetzt werden. Die Herabsetzung von **DB2_DISPATCHER_PEEKTIMEOUT** bewirkt, dass die Zuteilerroutine nur kurz prüft, ob die Verbindungsanforderung des Clients bereits da ist, ohne auf das Eintreffen der Verbindungsanforderung zu warten. Wenn ein ungültiger Wert definiert wird, wird der Standardwert verwendet. Diese Registrierdatenbankvariable ist nicht dynamisch.

DB2_DJ_INI

- Betriebssystem: Alle
- Standardwert:
  – UNIX: *db2-instanzverzeichnis*/cfg/db2dj.ini
  – Windows: *db2-installationsverzeichnis*\cfg\db2dj.ini
- Gibt den absoluten Pfadnamen der Konfigurationsdatei für die Föderation von Datenquellen an, zum Beispiel: `db2set DB2_DJ_INI=$HOME/sqllib/cfg/my_db2dj.ini`. Diese Datei enthält die Einstellungen für Umgebungsvariablen von Datenquellen. Diese Umgebungsvariablen werden vom Informix-Wrapper und von den durch WebSphere Federation Server bereitgestellten Wrappern verwendet.

  Das folgende Beispiel zeigt eine Konfigurationsdatei für die Föderation von Datenquellen:

```
INFORMIXDIR=/informix/client_sdk
INFORMIXSERVER=inf93
ORACLE_HOME=/usr/oracle9i
SYBASE=/sybase/V12
SYBASE_OCS=OCS-12_5
```

Für die Datei db2dj.ini gelten die folgenden Einschränkungen:

- Einträge müssen das Format *umgebvarname=wert* haben, wobei *umgebvarname* der Name der Umgebungsvariablen und *wert* der entsprechende Wert ist.
- Der Name der Umgebungsvariablen kann maximal 255 Byte lang sein.
- Der Wert der Umgebungsvariablen kann maximal 765 Byte lang sein.

Diese Variable wird ignoriert, sofern nicht der Parameter **FEDERATED** des Datenbankmanagers den Wert YES hat.

DB2DMNBCKCTLR

- Betriebssystem: Windows
- Standardwert: NULL, Werte: ? oder ein Domänenname
- Wenn Sie den Namen der Domäne kennen, für die der DB2-Server der Backup-Domänen-Controller ist, stellen Sie **DB2DMNBCKCTLR**=DOMÄNENNAME ein. Die Angabe DOMÄNENNAME muss in Großbuchstaben erfolgen. Sie können DB2 die Domäne ermitteln lassen, für die die lokale Maschine ein Backup-Domänen-Controller ist, indem Sie **DB2DMNBCKCTLR**=? einstellen. Wenn die Profilvariable **DB2DMNBCKCTLR** nicht oder mit leerem Wert definiert wird, führt DB2 die Authentifizierung auf dem primären Domänencontroller aus.

  Anmerkung: DB2 verwendet standardmäßig keinen vorhandenen Backup-Domänen-Controller, weil ein Backup-Domänen-Controller die Synchronisation mit dem primären Domänencontroller verlieren und damit ein Sicherheitsrisiko darstellen kann. Es kann zum Verlust der Synchronisation kommen, wenn die Sicherheitsdatenbank des primären Domänencontrollers aktualisiert wird, die Änderungen jedoch nicht an einen Backup-Domänen-Controller weitergegeben werden. Der Grund hierfür könnten Netzwerklatenzzeiten oder ein nicht funktionierender Computersuchdienst sein.

Anmerkung: **DB2DMNBCKCTLR** ist veraltet und wird in einem späteren Release entfernt. Diese Variable ist nicht mehr erforderlich, weil im Active Directory keine Backup-Domänen-Controller mehr vorhanden sind.

DB2_DOCHOST

- Betriebssystem: Alle
- Standardwert: nicht definiert (DB2 versucht dennoch, über die IBM Website unter publib.boulder.ibm.com/infocenter/db2luw/v9r5 auf die Informationszentrale zuzugreifen); Werte: http://*hostname*, wobei *hostname* ein gültiger Hostname oder eine gültige IP-Adresse ist
- Gibt den Namen des Hosts an, auf dem die *DB2-Informationszentrale* installiert ist. Diese Variable kann bei der Installation der *DB2-Informationszentrale* automatisch definiert werden, wenn die Option zur automatischen Konfiguration im DB2-Installationsassistenten ausgewählt wird.

DB2_DOCPORT

- Betriebssystem: Alle
- Standardwert: NULL, Werte: jede gültige Portnummer
- Gibt die Portnummer an, über die das DB2-Hilfesystem die DB2-Dokumentation bereitstellt. Diese Variable kann bei der Installation der *DB2-Informationszentrale* automatisch definiert werden, wenn die Option zur automatischen Konfiguration im DB2-Installationsassistenten ausgewählt wird.

DB2_ENABLE_AUTOCONFIG_DEFAULT

- Betriebssystem: Alle
- Standardwert: NULL, Werte: YES oder NO
- Mit dieser Variable wird gesteuert, ob der Konfigurationsadvisor automatisch bei der Datenbankerstellung ausgeführt wird. Wenn die Variable **DB2_ENABLE_AUTOCONFIG_DEFAULT** nicht definiert (null) ist, bewirkt dies dasselbe wie die Einstellung der Variablen auf den Wert YES. Das heißt, der Konfigurationsadvisor wird bei der Datenbankerstellung ausgeführt. Sie brauchen die Instanz nach dem Definieren dieser Variablen nicht erneut zu starten. Wenn Sie den Befehl AUTOCONFIGURE oder die Anweisung CREATE DB AUTOCONFIGURE ausführen, überschreiben diese die Einstellung von **DB2_ENABLE_AUTOCONFIG_DEFAULT**.

DB2_ENABLE_LDAP

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO.
- Gibt an, ob LDAP (Lightweight Directory Access Protocol) verwendet wird. LDAP ist eine Zugriffsmethode auf Verzeichnisservices.

DB2_EVMON_EVENT_LIST_SIZE

- Betriebssystem: Alle
- Standardwert: 0 (keine Begrenzung), Werte: Ein in KB (Kb/kb), MB (Mb/mb) oder GB (Gb/gb) angegebener Wert. Obwohl es keine feste obere Grenze für diese Variable gibt, wird sie durch die Größe des verfügbaren Speichers aus dem Monitorzwischenspeicher begrenzt.
- Diese Registrierdatenbankvariable gibt die maximale Anzahl Byte an, die in einer Warteschlange gesammelt werden können, indem sie darauf warten, in einen bestimmten Ereignismonitor geschrieben zu werten. Wenn diese Begrenzung erreicht wird, warten Agenten, die versuchen, Ereignismonitorsätze zu senden, bis die Größe der Warteschlange unter diesen Schwellenwert sinkt.

  Anmerkung: Wenn Aktivitätssätze nicht aus dem Monitorzwischenspeicher zugeordnet werden können, werden sie übergangen. Um dies zu vermeiden, setzen Sie den Konfigurationsparameter **mon_heap_sz** auf den Wert AUTOMATIC. Wenn der Konfigurationsparameter **mon_heap_sz** auf einen bestimmten Wert gesetzt wird, stellen Sie sicher, dass die Variable **DB2_EVMON_EVENT_LIST_SIZE** auf einen kleineren Wert gesetzt wird. Diese Maßnahmen können jedoch nicht garantieren, dass Aktivitätssätze nicht übergangen werden, da der Monitorzwischenspeicher auch zur Verfolgung anderer Monitorelemente verwendet wird.

DB2_EVMON_STMT_FILTER

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte:

– ALL: Gibt an, dass die Ausgabe für alle Anweisungsereignismonitore zu filtern ist. Diese Option ist exklusiv.
– '*nameA nameB nameC*': Jeder Name in der Zeichenfolge stellt den Namen eines Ereignismonitors dar, für den Datensätze zu filtern sind. Wenn mehr als ein Name angegeben wird, muss jeder Name durch ein und nur ein Leerzeichen getrennt angegeben werden. Alle Eingabenamen werden von DB2 in Großbuchstaben umgesetzt. Die maximale Anzahl von Ereignismonitoren, die Sie angeben können, beträgt 32. Jeder Monitorname kann maximal bis zu 18 Zeichen lang sein.
– '*nameA:op1,op2 nameB:op1,op2 nameC:op1*': Jeder Name in der Zeichenfolge stellt den Namen eines Ereignismonitors dar, für den Datensätze zu filtern sind. Jede Option (*op1*, *op2* usw.) stellt einen ganzzahligen Wert dar, der einer bestimmten SQL-Operation zugeordnet ist. Die Angabe ganzzahliger Werte ermöglicht Benutzern festzustellen, welche Regeln auf welche Ereignismonitore anzuwenden sind.

• Die Variable **DB2_EVMON_STMT_FILTER** kann dazu verwendet werden, die Anzahl von Datensätzen zu verringern, die durch einen Anweisungsereignismonitor geschrieben werden. Wenn sie definiert wird, sorgt diese Registrierdatenbankvariable dafür, dass nur die Datensätze für die folgenden SQL-Operationen an den angegebenen Ereignismonitor geschrieben werden:

Tabelle 65. Liste der ganzzahligen Werte, die in der Variablen **DB2_EVMON_STMT_FILTER** *zur Darstellung bestimmter SQL-Operationen angegeben werden können*

SQL-Operation	Zuordnung ganzzahliger Werte
SELECT	15
EXECUTE	2
EXECUTE_IMMEDIATE	3
CLOSE	6
STATIC COMMIT	8
STATIC ROLLBACK	9
CALL	12
PRE_EXEC	17

Alle anderen Operationen werden in der Ausgabe des Anweisungsereignismonitors nicht berücksichtigt. Um die Gruppe von Operationen anzupassen, für die Datensätze in den Ereignismonitor geschrieben werden, verwenden Sie ganzzahlige Werte.

Beispiel 1:

```
db2set DB2_EVMON_STMT_FILTER= 'mon1 monitor3'
```

In diesem Beispiel empfangen die Ereignismonitore 'mon1' und 'monitor3' einen Datensatz für eine eingeschränkte Liste von Anwendungsanforderungen. Wenn zum Beispiel eine Anwendung, die durch den Anweisungsereignismonitor 'mon1' überwacht wird, eine dynamische SQL-Anweisung vorbereitet, auf der Basis dieser Anweisung einen Cursor öffnet, 10.000 Zeilen aus diesem Cursor abruft und anschließend eine Anforderung zum Schließen des Cursors absetzt, wird nur ein Datensatz für die Schließanforderung in der Ausgabe des Ereignismonitors 'mon1' angezeigt.

Beispiel 2:

```
db2set DB2_EVMON_STMT_FILTER='evmon1:3,8 evmon2:9,15
```

In diesem Beispiel empfangen 'evmon1' und 'evmon2' einen Datensatz für eine eingeschränkte Liste von Anwendungsanforderungen. Wenn z. B. eine Anwendung, die durch den Anweisungsereignismonitor 'evmon1' überwacht wird, eine Anweisung CREATE absetzt, werden nur die Operationen EXECUTE IMMEDIATE und STATIC COMMIT in der Ausgabe des Ereignismonitors 'evmon1' angezeigt. Wenn eine Anwendung, die durch den Anweisungsereignismonitor 'evmon2' überwacht wird, SQL mit SELECT und STATIC ROLLBACK ausführt, werden nur diese zwei Operationen in der Ausgabe des Ereignismonitors 'evmon2' angezeigt.

Anmerkung: Definitionen von Konstanten für Datenbanksystemmonitore finden Sie in der Headerdatei sqlmon.h.

DB2_EXTSECURITY

- Betriebssystem: Windows
- Standardwert: ON, Werte: ON oder OFF
- Verhindert unbefugten Zugriff auf DB2 durch Sperren der DB2-Systemdateien. Zur Vermeidung potenzieller Probleme sollte diese Registrierdatenbankvariable nicht auf OFF gesetzt werden.

DB2_FALLBACK

- Betriebssystem: Windows
- Standardwert: OFF, Werte: ON oder OFF
- Mit dieser Variablen können Sie alle Datenbankverbindungen bei der Verarbeitung einer Datenbankrückübertragung (Fallback) zwangsweise trennen. Sie wird in Verbindung mit der Unterstützung für die Funktionsübernahme (Failover) in der Windows-Umgebung mit Microsoft Cluster Server (MSCS) verwendet. Wenn **DB2_FALLBACK** nicht definiert oder auf OFF gesetzt ist und bei der Rückübertragung (Fallback) eine Datenbankverbindung besteht, kann die DB2-Ressource nicht in den Offlinestatus versetzt werden. Dies bedeutet, dass die Rückübertragung fehlschlägt.

DB2_FMP_COMM_HEAPSZ

- Betriebssystem: Windows, UNIX
- Standardwert: 20 MB oder ausreichend Speicherbereich, um 10 abgeschirmte Routinen ausführen zu können (je nachdem, welcher Wert größer ist). Unter AIX ist der Standardwert 256 MB.
- Diese Variable gibt die Größe des Pools (in 4-KB-Seiten) an, der für Aufrufe abgeschirmter Routinen, wie zum Beispiel Aufrufe gespeicherter Prozeduren oder benutzerdefinierter Funktionen, verwendet wird. Der von den einzelnen abgeschirmten Routinen verwendete Speicherbereich entspricht dem zweifachen Wert des Konfigurationsparameters **aslheapsz**.

  Wenn Sie auf Ihrem System eine große Anzahl an abgeschirmten Routinen ausführen, müssen Sie den Wert dieser Variablen unter Umständen erhöhen. Wenn Sie eine sehr kleine Anzahl an abgeschirmten Routinen ausführen, können Sie den Wert senken.

  Wenn dieser Wert auf 0 gesetzt wird, bedeutet dies, dass keine Gruppe erstellt wird, sodass keine abgeschirmten Routinen aufgerufen werden können. Dies bedeutet außerdem, dass der Diagnosemonitor und die Funktionalität zur automatischen Datenbankpflege (z. B. automatische

Backups, Statistikerfassungen und Reorganisationen) inaktiviert wird, da diese Funktionalität von der Infrastruktur für abgeschirmte Routinen abhängig ist.

DB2_GRP_LOOKUP

- Betriebssystem: Windows
- Standardwert: NULL, Werte: LOCAL, DOMAIN, TOKEN, TOKENLOCAL, TOKENDOMAIN
- Diese Variable gibt an, welcher Windows-Sicherheitsmechanismus zur Aufzählung der Gruppen verwendet wird, denen ein Benutzer angehört.

DB2_HADR_BUF_SIZE

- Betriebssystem: Alle
- Standardwert: 2***logbufsz**
- Diese Variable gibt die Protokollempfangspuffergröße des Bereitschaftssystems in Einheiten von Protokollseiten an. Wenn nicht definiert, verwendet DB2 das Zweifache des Werts des Konfigurationsparameters **logbufsz** für die Empfangspuffergröße des Bereitschaftssystems. Diese Variable muss in der Bereitschaftsinstanz definiert werden. Sie wird von der Primärdatenbank ignoriert.

  Wenn der HADR-Synchronisationsmodus (Datenbankkonfigurationsparameter **hadr_syncmode**) auf den Wert ASYNC gesetzt ist, kann ein langsames Bereitschaftssystem im Peerstatus die Sendeoperation auf dem Primärsystem aufhalten und so die Transaktionsverarbeitung in der Primärdatenbank blockieren. In einer Bereitschaftsdatenbank kann ein Protokollempfangspuffer mit einer größeren als der Standardgröße konfiguriert werden, um die Aufnahme einer größeren Menge unverarbeiteter Protokolldaten zu ermöglichen. Dadurch lassen sich kurze Zeiträume auffangen, in denen die Primärdatenbank schneller Protokolldaten generiert, als die Bereitschaftsdatenbank sie entgegennehmen kann, ohne dass die Transaktionsverarbeitung in der Primärdatenbank blockiert wird.

  Anmerkung: Durch einen größeren Protokollempfangspuffer können Transaktionslastspitzen auf der Primärdatenbank besser ausgeglichen werden, der Puffer wird jedoch trotzdem voll, wenn die durchschnittliche Wiederholungsrate auf der Bereitschaftsdatenbank niedriger ist als die Protokollierungsrate auf der Primärdatenbank.

DB2_HADR_NO_IP_CHECK

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON | OFF
- Gibt an, ob die IP-Überprüfung für HADR-Verbindungen umgangen werden soll.
- Diese Variable wird vorwiegend in NAT-Umgebungen (NAT = Network Address Translation) verwendet, um die IP-Überprüfung von HADR-Verbindungen zu umgehen. Die Verwendung dieser Variablen in anderen Umgebungen wird nicht empfohlen, weil dadurch die Prüfung auf ordnungsgemäßen Zustand der HADR-Konfiguration beeinträchtigt wird. Standardmäßig wird die Konsistenz der Konfiguration der lokalen und fernen Hostparameter beim Herstellen einer HADR-Verbindung überprüft. Zur Überprüfung wird eine Zuordnung zwischen Hostnamen und IP-Adressen hergestellt. Dabei werden zwei Überprüfungsoperationen ausgeführt:

– Parameter **HADR_LOCAL_HOST** auf Primäreinheit = Parameter **HADR_REMOTE_HOST** auf Bereitschaftseinheit
– Parameter **HADR_REMOTE_HOST** auf Primäreinheit = Parameter **HADR_LOCAL_HOST** auf Bereitschaftseinheit

Die Verbindung wird geschlossen, wenn bei dieser Überprüfung Fehler festgestellt werden.

Wenn dieser Parameter aktiviert wird, dann führt das System keine IP-Überprüfung durch.

DB2_HADR_PEER_WAIT_LIMIT

- Betriebssystem: Alle
- Standardwert: 0 (d. h. kein Grenzwert), Werte: 0 bis maximal 32-Bit-Integer (einschließlich) ohne Vorzeichen
- Ab DB2 Version 9.5 Fixpack 1 wird durch das Setzen der Registrierdatenbankvariablen **DB2_HADR_PEER_WAIT_LIMIT** für die HADR-Primärdatenbank der Peerstatus aufgehoben, wenn die Protokollierung in der Primärdatenbank für die angegebene Anzahl von Sekunden blockiert wurde, weil das Protokoll in die Bereitschaftsdatenbank repliziert wird. Wenn dieser Grenzwert erreicht wird, unterbricht die Primärdatenbank die Verbindung zur Bereitschaftsdatenbank. Wenn das Peerfenster inaktiviert ist, befindet sich die Primärdatenbank im Status 'unterbrochen' und die Protokollierung wird wieder aufgenommen. Wenn das Peerfenster aktiviert ist, befindet sich die Primärdatenbank im Status 'Unterbrochener Peer', in dem die Protokollierung weiterhin blockiert ist. Die Primärdatenbank verlässt den Status 'Unterbrochener Peer', wenn die Verbindung wieder hergestellt wird oder das Peerfenster abläuft. Die Protokollierung wird wieder aufgenommen, sobald die Primärdatenbank den Status 'Unterbrochener Peer' verlässt. Dieser Parameter ist in der Bereitschaftsdatenbank nicht wirksam. Es empfiehlt sich jedoch, dass in der Primär- und der Bereitschaftsdatenbank der gleiche Wert verwendet wird. Ungültige Werte (keine Zahl oder negative Zahlen) werden als ″0″ interpretiert, d. h. es gibt keinen Grenzwert. Dieser Parameter ist statisch. Die Datenbankinstanz muss erneut gestartet werden, um diesen Parameter zu aktivieren.

DB2_HADR_SORCVBUF

- Betriebssystem: Alle
- Standardwert: TCP-Socketempfangspuffergröße des Betriebssystems, Werte: 1024 bis 4294967295
- Diese Variable gibt für die HADR-Verbindung die Größe des Empfangspuffers für den TCP-Socket des Betriebssystems an und ermöglicht es Benutzern damit, die HADR-TCP/IP-Funktionsweise separat von anderen Verbindungen anzupassen. Bei manchen Betriebssystemen wird der vom Benutzer angegebene Wert automatisch gerundet oder begrenzt. Die tatsächliche Puffergröße für die HADR-Verbindung wird in der Datei db2diag.log protokolliert. Im Handbuch zur Netzoptimierung für das verwendete Betriebssystem finden Sie Informationen zu den optimalen Einstellungen für diesen Parameter auf der Basis des anfallenden Netzdatenverkehrs. Diese Variable sollte zusammen mit **DB2_HADR_SOSNDBUF** verwendet werden.

DB2_HADR_SOSNDBUF

- Betriebssystem: Alle

- Standardwert: TCP-Socketsendepuffergröße des Betriebssystems, Werte: 1024 bis 4294967295
- Diese Variable gibt für die HADR-Verbindung die Größe des Sendepuffers für den TCP-Socket des Betriebssystems an und ermöglicht es Benutzern damit, die HADR-TCP/IP-Funktionsweise separat von anderen Verbindungen anzupassen. Bei manchen Betriebssystemen wird der vom Benutzer angegebene Wert automatisch gerundet oder begrenzt. Die tatsächliche Puffergröße für die HADR-Verbindung wird in der Datei db2diag.log protokolliert. Im Handbuch zur Netzoptimierung für das verwendete Betriebssystem finden Sie Informationen zu den optimalen Einstellungen für diesen Parameter auf der Basis des anfallenden Netzdatenverkehrs. Diese Variable sollte zusammen mit **DB2_HADR_SORCVBUF** verwendet werden.

DB2LDAP_BASEDN

- Betriebssystem: Alle
- Standardwert: NULL, Werte: alle gültigen definierten Basisdomänennamen.
- Wenn diese Variable definiert ist, werden die LDAP-Objekte für DB2 im LDAP-Verzeichnis unter

```
CN=System
CN=IBM
CN=DB2
```

  unter dem angegebenen definierten Basisnamen (Basis-DN) gespeichert.

  Wenn diese Variable für Microsoft Active Directory Server verwendet wird, stellen Sie sicher, dass die Werte CN=DB2, CN=IBM und CN=System unter diesem definierten Namen (DN) definiert sind.

DB2LDAPCACHE

- Betriebssystem: Alle
- Standardwert: YES, Werte: YES oder NO.
- Gibt an, dass der LDAP-Cache aktiviert werden soll. Dieser Cache wird zum Katalogisieren der Datenbank-, Knoten- und DCS-Verzeichnisse auf der lokalen Maschine verwendet.

  Führen Sie die folgenden Befehle aus, um sicherzustellen, dass sich in Ihrem Cache die aktuellsten Einträge befinden:

```
REFRESH LDAP IMMEDIATE ALL
```

  Dieser Befehl aktualisiert das Datenbank- und das Knotenverzeichnis und entfernt falsche Einträge.

DB2LDAP_CLIENT_PROVIDER

- Betriebssystem: Windows
- Standardwert: NULL (wenn verfügbar, wird Microsoft verwendet, andernfalls wird IBM verwendet). Werte: IBM oder Microsoft.
- Bei der Ausführung in einer Windows-Umgebung unterstützt DB2 die Verwendung von Microsoft-LDAP-Clients oder IBM LDAP-Clients zum Zugriff auf das LDAP-Verzeichnis. Diese Registrierdatenbankvariable wird dazu verwendet, den von DB2 zu verwendenden LDAP-Client explizit auszuwählen.

  Anmerkung: Verwenden Sie zum Anzeigen des aktuellen Werts dieser Registrierdatenbankvariablen den Befehl db2set:

```
db2set DB2LDAP_CLIENT_PROVIDER
```

DB2LDAPHOST

- Betriebssystem: Alle
- Standardwert: NULL, Werte: beliebiger gültiger Hostname
- Gibt den Hostnamen des Standorts für das LDAP-Verzeichnis an.

DB2LDAP_KEEP_CONNECTION

- Betriebssystem: Alle
- Standardwert: YES, Werte: YES oder NO.
- Gibt an, ob DB2 die internen LDAP-Verbindungskennungen in den Cache stellt. Wenn diese Variable auf NO gesetzt wird, stellt DB2 die Kennungen für LDAP-Verbindungen zum Verzeichnisserver nicht in den Cache. Dies führt mit einiger Wahrscheinlichkeit zu einer Leistungsbeeinträchtigung. Es kann jedoch wünschenswert sein, die Variable **DB2LDAP_KEEP_CONNECTION** auf den Wert NO zu setzen, wenn die Anzahl simultan aktiver LDAP-Clientverbindungen zum Verzeichnisserver auf ein Minimum reduziert werden muss.

  Diese Variable wird standardmäßig auf den Wert YES gesetzt, um die optimale Leistung zu erzielen.

  Die Registrierdatenbankvariable **DB2LDAP_KEEP_CONNECTION** ist nur als Profilregistrierdatenbankvariable der globalen Ebene in LDAP implementiert, sodass sie mit dem Befehl db2set unter Angabe der Option -gl wie folgt definiert werden muss:

  ```
  db2set -gl DB2LDAP_KEEP_CONNECTION=NO
  ```

DB2LDAP_SEARCH_SCOPE

- Betriebssystem: Alle
- Standardwert: DOMAIN, Werte: LOCAL, DOMAIN oder GLOBAL
- Gibt den Bereich für die Suche nach Informationen an, die sich in Datenbankpartitionen oder Domänen im Lightweight Directory Access Protocol (LDAP) befinden. Der Wert LOCAL inaktiviert das Suchen im LDAP-Verzeichnis. Beim Wert DOMAIN wird das LDAP-Verzeichnis nur nach der aktuellen Verzeichnispartition durchsucht. Beim Wert GLOBAL wird das LDAP-Verzeichnis in allen Verzeichnispartitionen durchsucht, bis das Objekt gefunden wird.

DB2_LOAD_COPY_NO_OVERRIDE

- Betriebssystem: Alle
- Standardwert: NONRECOVERABLE, Werte: COPY YES oder NONRECOVERABLE
- Diese Variable konvertiert jeden Befehl LOAD COPY NO je nach Wert der Variablen entweder in LOAD COPY YES oder NONRECOVERABLE. Diese Variable gilt für HADR-Primärdatenbanken und für Standarddatenbanken (nicht HADR). Sie wird in einer HADR-Bereitschaftsdatenbank ignoriert. Wenn in einer HADR-Primärdatenbank diese Variable nicht definiert ist, wird der Befehl LOAD COPY NO in LOAD NONRECOVERABLE konvertiert. Der Wert dieser Variablen gibt entweder eine nicht wiederherstellbare Ladeoperation oder das Kopierziel an, wobei die gleiche Syntax wie bei der Klausel COPY YES zu verwenden ist.

DB2LOADREC

- Betriebssystem: Alle
- Standardwert: NULL.

- Dient zum Überschreiben der Speicherposition der Ladekopie bei einer aktualisierenden Recovery (ROLLFORWARD). Wenn der Benutzer die physische Speicherposition der Ladekopie geändert hat, muss die Variable **DB2LOADREC** vor dem Ausführen der aktualisierenden Recovery definiert werden.

DB2LOCK_TO_RB

- Betriebssystem: Alle
- Standardwert: NULL, Werte: STATEMENT
- Gibt an, ob ein Rollback bei Überschreitungen von Sperrzeiten jeweils für die gesamte Transaktion oder nur für die aktuelle Anweisung durchgeführt werden soll. Wenn **DB2LOCK_TO_RB** den Wert STATEMENT hat, führen Überschreitungen von Sperrzeiten dazu, dass nur die aktuelle Anweisung durch ein Rollback rückgängig gemacht wird. Jeder andere Wert für diesen Parameter bewirkt, dass die gesamte Transaktion durch ein Rollback rückgängig gemacht wird.

DB2_MAP_XML_AS_CLOB_FOR_DLC

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO.
- Die Registrierdatenbankvariable **DB2_MAP_XML_AS_CLOB_FOR_DLC** bietet die Möglichkeit, das DESCRIBE- und FETCH-Standardverhalten für XML-Werte für Clients (bzw. DRDA-Anwendungsrequester) zu überschreiben, die XML als Datentyp nicht unterstützen. Der Standardwert NO gibt an, dass für diese Clients ein DESCRIBE von XML-Werten den Ergebnistyp BLOB(2GB) liefert und ein FETCH von XML-Werten eine implizite XML-Serialisierung in BLOB bewirkt, die eine XML-Deklaration enthält, die eine UTF-8-Codierung angibt.

  Wenn die Variable auf den Wert YES gesetzt ist, liefert ein DESCRIBE von XML-Werten den Ergebnistyp CLOB(2GB) und ein FETCH von XML-Werten bewirkt eine implizite XML-Serialisierung in CLOB, die keine XML-Deklaration enthält.

Anmerkung: **DB2_MAP_XML_AS_CLOB_FOR_DLC** ist veraltet und wird in einem zukünftigen Release entfernt. Diese Variable wird nicht mehr benötigt, weil die meisten vorhandenen DB2-Anwendungen, die auf XML-Werte zugreifen, dies mit einem XML-fähigen Client tun.

DB2_MAX_LOB_BLOCK_SIZE

- Betriebssystem: Alle
- Standardwert: 0 (kein Zeitlimit), Werte: 0 bis 21487483647
- Legt die maximale Größe von LOB- oder XML-Daten fest, die in einem Block zurückgegeben werden sollen. Dieser Wert ist kein fixer Maximalwert. Wenn dieser Maximalwert auf dem Server während eines Datenabrufs erreicht wird, beendet der Server das Schreiben der aktuellen Zeile, bevor er eine Antwort für den Befehl, zum Beispiel FETCH, an den Client generiert.

DB2_MEMORY_PROTECT

- Betriebssystem: AIX mit Speicherschlüsselunterstützung
- Standardwert: NO, Werte: NO oder YES
- Diese Registrierdatenbankvariable aktiviert eine Speicherschutzfunktion, die mit Speicherschlüsseln arbeitet, um durch unzulässige Speicherzugriffe verursachte Datenverluste im Pufferpool zu verhindern. Der Speicherschutz funktioniert in der Weise, dass bestimmt wird, zu wel-

chen Zeiten die Threads der DB2-Steuerkomponente Zugriff auf den Pufferpoolspeicher haben sollen und zu welchen sie keinen Zugriff haben sollen. Wenn **DB2_MEMORY_PROTECT** auf YES gesetzt ist, wird jeder unzulässige Versuch eines Threads der DB2-Steuerkomponente, auf den Pufferpoolspeicher zuzugreifen, abgefangen (Trap).

DB2_MEMORY_PROTECT muss aktiviert sein, damit die Funktion **DB2_THREAD_SUSPENSION** für die Ausfallsicherheit bei Traps funktioniert.

Anmerkung: Sie können keinen Speicherschutz verwenden, wenn **DB2_LGPAGE_BP** auf YES gesetzt ist. Selbst wenn **DB2_MEMORY_PROTECT** auf YES gesetzt ist, wird DB2 den Pufferpoolspeicher nicht schützen und die Funktion inaktivieren.

DB2NOEXITLIST

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable gibt an, dass DB2 keine Exitlistenroutine laden darf und dass unabhängig von der Einstellung der Registrierdatenbankvariablen **DB2_COMMIT_ON_EXIT** keine Commitoperation ausgeführt werden darf, wenn die Anwendung beendet wird.

  Wenn **DB2NOEXITLIST** inaktiviert und **DB2_COMMIT_ON_EXIT** aktiviert ist, werden alle unvollständigen Transaktionen für Anwendungen mit eingebettetem SQL automatisch festgeschrieben. Sie sollten explizit COMMIT- oder ROLLBACK-Anweisungen hinzufügen, wenn eine Anwendung beendet wird.

  Anwendungen, die vor der Beendigung der Anwendung die DB2-Bibliothek dynamisch laden und entladen, stürzen möglicherweise ab, wenn die DB2-Exitroutine aufgerufen wird. Dieser Absturz kann darauf zurückzuführen sein, dass die Anwendung versucht, eine Funktion aufzurufen, die im Speicher nicht vorhanden ist. Um diese Situation zu vermeiden, muss die Registrierdatenbankvariable **DB2NOEXITLIST** gesetzt werden.

DB2_NUM_CKPW_DAEMONS

- Betriebssystem: UNIX
- Standardwert: 3, Werte: 1[:FORK] bis 100[:FORK]
- Mithilfe der Registrierdatenbankvariablen **DB2_NUM_CKPW_DAEMONS** können Sie eine konfigurierbare Anzahl von Kennwortprüfdämonen starten. Die Dämonen werden bei Ausführung von db2start erstellt und verarbeiten Anforderungen zum Prüfen von Kennwörtern, wenn das Standardsicherheits-Plug-in IBMOSauthserver verwendet wird. Eine Erhöhung des Werts von **DB2_NUM_CKPW_DAEMONS** kann die Zeit verkürzen, die zur Herstellung einer Datenbankverbindung benötigt wird. Dies gilt jedoch nur in Szenarios, in denen viele Verbindungen gleichzeitig hergestellt werden und in denen die Authentifizierung aufwendig ist.

  Die Variable **DB2_NUM_CKPW_DAEMONS** kann auf einen Wert zwischen 1 und 100 gesetzt werden. Der Datenbankmanager erstellt die Anzahl von Dämonen, die durch **DB2_NUM_CKPW_DAEMONS** angegeben wird. Jeder Dämon kann Anforderungen zum Prüfen von Kennwörtern direkt verarbeiten.

  Der optionale Parameter FORK kann hinzugefügt werden, um die Dämonen zum Prüfen von Kennwörtern zu befähigen, ein externes

Kennwortprüfprogramm (db2ckpw) zur Verarbeitung von Anforderungen zum Prüfen von Kennwörtern explizit zu starten. Dies ist der Einstellung von **DB2_NUM_CKPW_DAEMONS** auf den Wert null in früheren Releases ähnlich. Im Modus FORK startet jeder Dämon zum Prüfen von Kennwörtern das Kennwortprüfprogramm für jede Anforderung zum Prüfen eines Kennworts. Die Dämonen im Modus FORK werden als Instanzeigner gestartet.

Wenn **DB2_NUM_CKPW_DAEMONS** auf den Wert null gesetzt wird, wird der effektive Wert auf 3:FORK gesetzt, sodass drei Dämonen zum Prüfen von Kennwörtern im Modus FORK gestartet werden.

DB2_OPTSTATS_LOG

- Betriebssystem: Alle
- Standardwert: nicht definiert (Details siehe unten), Werte: OFF, ON {NUM | SIZE | NAME | DIR}
- Die Variable **DB2_OPTSTATS_LOG** gibt die Attribute der Protokolldateien für Statistikereignisse an, die zum Überwachen und Analysieren der Statistikerfassung in Bezug auf Aktivitäten verwendet werden. Wenn **DB2_OPTSTATS_LOG** nicht definiert oder auf ON gesetzt wird, wird die Protokollierung von Statistikereignissen aktiviert, sodass Sie zur besseren Fehlerbestimmung die Systemleistung überwachen und ein Verlaufsprotokoll anlegen können. Die Protokollsätze werden in die erste Protokolldatei geschrieben, bis diese gefüllt ist. Nachfolgende Protokollsätze werden in die nächste verfügbare Protokolldatei geschrieben. Wenn die maximale Anzahl von Dateien erreicht ist, wird die älteste Protokolldatei mit den neuen Protokollsätzen überschrieben. Wenn die Systemressourcennutzung in Ihrer Umgebung problematisch ist, können Sie diese Registrierdatenbankvariable inaktivieren, indem Sie sie auf OFF setzen.

  Wenn die Protokollierung von Statistikereignissen explizit aktiviert (auf ON gesetzt) wird, können Sie eine Reihe von Optionen ändern:

  - NUM: Die maximale Anzahl rollierender Protokolldateien. Standardwert=5, Werte: 1 - 15.
  - SIZE: Die maximale Größe der rollierenden Protokolldateien. (Die Größe jeder einzelnen rollierenden Datei ist SIZE/NUM.) Standardwert=100 MB, Werte: 1 MB – 4096 MB.
  - NAME: Der Basisname für rollierende Protokolldateien. Standardwert=db2optstats.*nummer*.log, z. B. db2optstats.0.log, db2optstats.1.log usw.
  - DIR: Das Basisverzeichnis für rollierende Protokolldateien. Standardwert = $DIAGPATH/events

  Sie können eine beliebige Anzahl dieser Optionen angegeben. Sie müssen nur sicherstellen, dass Sie als ersten Wert ON angeben, wenn Sie die Statistikprotokollierung aktivieren wollen. Wenn Sie zum Beispiel die Statistikprotokollierung mit einer maximalen Anzahl von 6 Protokolldateien, einer maximalen Dateigröße von 25 MB sowie dem Basisdateinamen mystatslog und dem Verzeichnis mystats aktivieren wollen, geben Sie den folgenden Befehl ein:

  ```
  db2set DB2_OPTSTATS_LOG=ON,NUM=6,SIZE=25,NAME=mystatslog,DIR=mystats
  ```

DB2REMOTEPREG

- Betriebssystem: Windows

- Standardwert: NULL, Werte: beliebiger gültiger Windows-Maschinenname
- Definiert den Namen der fernen Maschine, die die Win32-Registrierungsdatenbankliste von DB2-Instanzprofilen und DB2-Instanzen enthält. Der Wert für **DB2REMOTEPREG** sollte nur einmal nach der Installation von DB2 definiert und anschließend nicht mehr geändert werden. Verwenden Sie diese Variable mit großer Vorsicht.

DB2_RESOLVE_CALL_CONFLICT

- Betriebssystem: AIX, HP-UX, Solaris, Linux, Windows
- Standardwert: YES, Werte: YES, NO oder ALL
- Eliminiert SQLCODE SQL0746-Laufzeitfehler beim Absetzen von CALL-Anweisungen in den beiden folgenden Situationen:
  - Wenn bei Triggern **DB2_RESOLVE_CALL_CONFLICT** auf YES (Standardwert) oder ALL gesetzt ist.
  - Wenn bei SQL-Tabellenfunktionen **DB2_RESOLVE_CALL_CONFLICT** auf ALL gesetzt ist.

  In Übereinstimmung mit den Regeln des SQL-Standards für die Ausführungsreihenfolge liest bzw. modifiziert der Datenbankmanager möglicherweise die Subjekttabelle eines Triggers erst, wenn die durch eine Triggeroperation ausgelösten Modifikationen abgeschlossen sind. Darüber hinaus liest bzw. modifiziert der Datenbankmanager möglicherweise die Subjekttabelle einer Tabellenfunktion nicht, auf die an anderer Stelle in der aufrufenden Anweisung zugegriffen wird. Wenn innerhalb derselben aufrufenden Anweisungen mehrere Tabellenzugriffsanforderungen für diese Subjekttabellen vorliegen, werden diese konkurrierenden Anforderungen als *Mutating-Table-Konflikte* bezeichnet, da die Tabellen möglicherweise geändert werden, während gleichzeitig ein Trigger oder eine SQL-Tabellenfunktion versucht, sie zu referenzieren.

  Wird diese Variable auf den Standardwert YES gesetzt, erzwingt der Datenbankmanager die strikte Ausführungsreihenfolge beim Zugriff auf Tabellen in Prozeduren, die innerhalb von Triggern aufgerufen werden, je nach Bedarf durch die Verwendung von temporären Tabellen.

  Wird diese Variable auf ALL gesetzt, erzwingt der Datenbankmanager die strikte Ausführungsreihenfolge beim Zugriff auf Tabellen in Prozeduren, die innerhalb von Triggern aufgerufen werden, sowie in Tabellenfunktionen, je nach Bedarf durch die Verwendung von temporären Tabellen. Der Wert ALL ist ab Fixpack 2 verfügbar.

  Wenn Sie diese Variable auf NO setzen, geht der Datenbankmanager davon aus, dass keine Mutating-Table-Konflikte auftreten; stellt der Datenbankmanager dennoch einen potenziellen Mutating-Table-Konflikt fest, verhindert er diesen und generiert die Fehlernachricht SQLCODE SQL0746.
- Stoppen Sie vor dem Ändern des Werts für **DB2_RESOLVE_CALL_CONFLICT** die Instanz. Starten Sie nach dem Ändern des Werts die Instanz erneut; führen Sie einen Rebind für die Pakete durch, die den Aufruf von Triggern oder Tabellenfunktionen bewirken. Verwenden Sie die folgende Anweisung für einen Rebind der SQL-Prozeduren:

  ```
  CALL SYSPROC.REBIND_ROUTINE_PACKAGE ('P','prozedurschema.prozedurname',
  'CONSERVATIVE');
  ```
- **DB2_RESOLVE_CALL_CONFLICT** hat Einfluss auf die Leistung:

– Wenn bei Triggern die Registrierdatenbankvariable auf YES oder ALL gesetzt wird. Normalerweise sind die Auswirkungen gering. Die Leistung in OLTP-Umgebungen wird normalerweise nicht beeinträchtigt, da in den meisten Fällen nur eine kleine Anzahl von Zeilen durch die auslösende Anweisung modifiziert wird. In der Regel ist der Einfluss auf die Leistung durch **DB2_RESOLVE_CALL_CONFLICT** gering, wenn die allgemeinen Empfehlungen zur Verwendung von SMS (systemverwalteter Speicherbereich, System Managed Storage) für temporäre Tabellenbereiche befolgt werden.
– Wenn bei SQL-Tabellenfunktionen die Registrierdatenbankvariable auf ALL gesetzt wird. Wie groß die Auswirkungen sind, ist vom Typ des Konflikts abhängig:
  - Wenn eine CALL-Anweisung und eine andere Anweisung in derselben SQL-Tabellenfunktion versuchen, auf dieselbe Tabelle zuzugreifen, sind die Auswirkungen gering.
  - Wenn eine CALL-Anweisung in einer SQL-Tabellenfunktion und eine SELECT-Anweisung, die dieselbe SQL-Tabellenfunktion aufruft, versuchen, auf dieselbe Tabelle zuzugreifen, sind die Auswirkungen von der Menge der Daten abhängig, die in die temporäre Tabelle eingefügt werden.

DB2ROUTINE_DEBUG

- Betriebssystem: AIX und Windows
- Standardwert: OFF, Werte: ON oder OFF
- Gibt an, ob die Fehlerbehebungsfunktion (Debug) für gespeicherte Java-Prozeduren aktiviert wird. Wenn Sie nicht gerade Fehler in gespeicherten Java-Prozeduren beheben, sollten Sie den Standardwert OFF verwenden. Die Aktivierung des Fehlerbehebungsmodus hat Auswirkungen auf die Leistung.

Anmerkung: DB2ROUTINE_DEBUG ist veraltet und wird in einem zukünftigen Release entfernt. Dieser Debugger für gespeicherte Prozeduren wurde durch den Unified Debugger ersetzt.

DB2SATELLITEID

- Betriebssystem: Alle
- Standardwert: NULL, Werte: eine gültige Satelliten-ID, die in der Satellitensteuerungsdatenbank deklariert ist
- Gibt die Satelliten-ID an, die an den Satellitensteuerungsserver übergeben wird, wenn ein Satellit eine Synchronisation durchführt. Wenn für diese Variable kein Wert angegeben ist, wird die Anmelde-ID als Satelliten-ID verwendet.

DB2_SERVER_CONTIMEOUT

- Betriebssystem: Alle
- Standardwert: 180, Werte: 0 bis 32767 Sekunden
- Diese Registrierdatenbankvariable und die Registrierdatenbankvariable **DB2_DISPATCHER_PEEKTIMEOUT** dienen beide zur Konfiguration der Behandlung eines neuen Clients während der Verbindungszeit. Mit **DB2_SERVER_CONTIMEOUT** können Sie die Zeit (in Sekunden) anpassen, die ein Agent auf die Verbindungsanforderung eines Clients wartet, bevor er die Verbindung beendet. In den meisten Fällen sollte es nicht nötig sein, diese Registrierdatenbankvariable anzupassen. Wenn DB2-Clients beim Verbindungsaufbau jedoch beständig das vom Server definierte Zeitlimit überschreiten, können Sie einen höheren Wert für

DB2_SERVER_CONTIMEOUT angeben, um das Zeitlimitintervall zu verlängern. Wenn ein ungültiger Wert definiert wird, wird der Standardwert verwendet. Diese Registrierdatenbankvariable ist nicht dynamisch.

DB2_SERVER_ENCALG

- Betriebssystem: Alle
- Standardwert: NULL, Werte: AES_CMP oder AES_ONLY
- Der Advanced Encryption Standard (AES) kann zur Verschlüsselung von Benutzer-IDs und Kennwörtern verwendet werden, wenn Verbindungen zu Datenbankservern mit DB2 für Linux, UNIX und Windows Version 9.5 Fixpack 3 (und späteren Versionen) hergestellt werden. Wenn der Konfigurationsparameter **authentication** des Datenbankmanagers auf den Wert SERVER_ENCRYPT gesetzt ist, besteht das Standardverhalten des Datenbankservers darin, jeden Verschlüsselungsalgorithmus zu akzeptieren, den der Client vorschlägt. Die Registrierdatenbankvariable **DB2_SERVER_ENCALG** kann zum Ändern dieses Verhaltens verwendet werden. Sie wirkt sich auf alle Verbindungen zur DB2-Instanz ungeachtet der Anwendungssprache aus.

  Wenn diese Variable auf den Wert AES_ONLY gesetzt wird, akzeptiert der Datenbankserver nur Verbindungen, die mit AES-Verschlüsselung arbeiten. Wenn der Client die AES-Verschlüsselung nicht unterstützt, wird die Verbindung zurückgewiesen.

  Wenn diese Variable auf den Wert AES_CMP gesetzt wird, akzeptiert der Datenbankserver Benutzer-IDs und Kennwörter, die entweder mit AES oder DES verschlüsselt wurden, vereinbart jedoch AES, wenn der Client die AES-Verschlüsselung unterstützt.

  Sie müssen die Registrierdatenbankvariable **DB2_SERVER_ENCALG** nicht definieren, wenn Ihre Anwendung programmgesteuert zur Verwendung der AES-Verschlüsselung eingerichtet ist.

DB2SORT

- Betriebssystem: Alle, nur Server
- Standardwert: NULL.
- Diese Variable gibt die Speicherposition einer während der Laufzeit durch das Dienstprogramm LOAD zu ladenden Bibliothek an. Die Bibliothek enthält den Eingangspunkt für Funktionen, die beim Sortieren von Indexdaten verwendet werden. Verwenden Sie **DB2SORT**, um Sortierprogrammprodukte anderer Lieferanten mit dem Dienstprogramm LOAD zur Generierung von Tabellenindizes zu nutzen. Der angegebene Pfad muss relativ zum Datenbankserver definiert werden.

DB2_THREAD_SUSPENSION

- Betriebssystem: AIX mit Speicherschlüsselunterstützung
- Standardwert: OFF, Werte: ON oder OFF
- Diese Registrierdatenbankvariable aktiviert oder inaktiviert die Aussetzfunktion für DB2-Threads. Mithilfe dieser Funktion können Sie steuern, ob eine DB2-Instanz einen Trap auffangen kann, indem sie einen fehlhaften Thread der Steuerkomponente (d. h. einen Thread, der in unzulässiger Weise versucht hat, auf den durch Speicherschlüssel geschützten Speicher zuzugreifen) aussetzt.

  Anmerkung: Die Registrierdatenbankvariable **DB2_THREAD_SUSPENSION** kann nur aktiviert werden, wenn Registrierdatenbankvariable **DB2_MEMORY_PROTECT** auf YES gesetzt ist.

DB2_TRUNCATE_REUSESTORAGE

- Betriebssystem: Alle
- Standardwert: NULL (nicht definiert), Werte: IMPORT, import
- Mithilfe dieser Variablen können Sie Sperrenkonflikte zwischen dem Befehl IMPORT mit REPLACE und dem Befehl BACKUP ... ONLINE auflösen. In einigen Fällen können ein Online-Backup und Abschneideoperationen (TRUNCATE) nicht gleichzeitig ausgeführt werden. Wenn dieser Fall eintritt, können Sie die Variable **DB2_TRUNCATE_REUSESTORAGE** auf den Wert IMPORT bzw. import setzen. Dadurch wird das physische Abschneiden des Objekts, einschließlich Daten, Indizes, Langfelddaten, große Objekte (LOBs) und Blockzuordnungen (für MDC-Tabellen) übersprungen und nur ein logisches Abschneiden ausgeführt. Das heißt, der Befehl IMPORT mit REPLACE leert die Tabelle, sodass sich die logische Größe des Objekts verringert, der Plattenspeicherplatz jedoch zugeordnet bleibt.

  Diese Registrierdatenbankvariable ist dynamisch. Sie können sie definieren bzw. ihren Wert löschen, ohne die Instanz stoppen und starten zu müssen. Sie können die Variable **DB2_TRUNCATE_REUSESTORAGE** vor dem Start eines Online-Backups definieren und nach Abschluss des Online-Backups ihren Wert wieder löschen. In Umgebungen mit mehreren Datenbankpartitionen ist die Registrierdatenbankvariable nur auf den Knoten aktiv, auf denen sie definiert ist. Die Variable **DB2_TRUNCATE_REUSESTORAGE** ist nur auf permanenten DMS-Objekten wirksam.

  Wenn in SAP-Umgebungen **DB2_WORKLOAD**=SAP definiert ist, hat diese Registrierdatenbankvariable den Standardwert IMPORT.

DB2_USE_DB2JCCT2_JROUTINE

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: ON/YES/1/TRUE oder OFF/NO/0/FALSE
- Der Standardtreiber für gespeicherte Java-Prozeduren und benutzerdefinierte Java-Funktionen ist IBM Data Server Driver for JDBC and SQLJ. Wenn Sie den veralteten Treiber DB2 JDBC Type 2 Driver für Linux, UNIX und Windows verwenden wollen, um SQL-Anforderungen für Java-Routinen zu bedienen, setzen Sie **DB2_USE_DB2JCCT2_JROUTINE** auf einen der Werte OFF, NO, 0 oder FALSE.

DB2_UTIL_MSGPATH

- Betriebssystem: Alle
- Standardwert: Verzeichnis *instanzname*/tmp
- Die Registrierdatenbankvariable **DB2_UTIL_MSGPATH** wird in Verbindung mit der Prozedur SYSPROC.ADMIN_CMD, der Prozedur SYSPROC.ADMIN_REMOVE_MSGS und der benutzerdefinierten Funktion (UDF) SYSPROC.ADMIN_GET_MSGS verwendet. Sie gilt für die Instanzebene. Die Variable **DB2_UTIL_MSGPATH** kann definiert werden, um einen Verzeichnispfad auf dem Server anzugeben, in dem die abgeschirmte Benutzer-ID Dateien lesen, schreiben und löschen kann. Dieses Verzeichnis muss von allen Koordinatorpartitionen aus zugänglich sein und genügend Speicherplatz besitzen, um die Nachrichtendateien von Dienstprogrammen aufzunehmen.

Wenn dieser Pfad nicht festgelegt ist, wird standardmäßig das Verzeichnis *instanzname*/tmp verwendet. (Beachten Sie, dass das Verzeichnis *instanzname*/tmp bei der Deinstallation von DB2 bereinigt wird.)

Wenn dieser Pfad geändert wird, werden die Dateien, die sich in dem durch die vorige Einstellung der Variablen bezeichneten Verzeichnis befanden, nicht automatisch versetzt oder gelöscht. Wenn Sie den Inhalt der Nachrichten, die unter dem alten Pfad erstellt wurden, abrufen wollen, müssen Sie diese Nachrichten (die den Dienstprogrammnamen als Präfix und die Benutzer-ID als Suffix haben) manuell in das neue Verzeichnis versetzen, das durch **DB2_UTIL_MSGPATH** angegeben wird. Die nächste Nachrichtendatei eines Dienstprogramms wird an der neuen Position erstellt, gelesen und bereinigt.

Die Dateien unter dem durch **DB2_UTIL_MSGPATH** angegebenen Verzeichnis sind für die jeweiligen Dienstprogramme spezifisch und nicht transaktionsabhängig. Sie sind kein Teil des Backup-Images. Die Dateien unter dem durch **DB2_UTIL_MSGPATH** angegebenen Verzeichnis werden vom Benutzer verwaltet. Dies bedeutet, dass ein Benutzer die Nachrichtendateien mithilfe der Prozedur SYSPROC.ADMIN_REMOVE_UTILMSG löschen kann. Diese Dateien werden bei der Deinstallation von DB2 nicht bereinigt.

DB2_VENDOR_INI

- Betriebssystem: AIX, HP-UX, Solaris und Windows
- Standardwert: NULL, Werte: eine beliebige gültige Angabe für Pfad und Datei.
- Verweist auf eine Datei, die alle herstellerspezifischen Umgebungseinstellungen enthält. Der Wert wird beim Start des Datenbankmanagers gelesen.

Anmerkung: **DB2_VENDOR_INI** ist in Version 9.5 veraltet und wird in einem zukünftigen Release möglicherweise entfernt. Sie können die in der Umgebungsvariablen enthaltenen Einstellungen stattdessen in die Datei einfügen, die durch die Variable **DB2_DJ_INI** angegeben wird.

DB2_XBSA_LIBRARY

- Betriebssystem: AIX, HP-UX, Solaris und Windows
- Standardwert: NULL, Werte: eine beliebige gültige Angabe für Pfad und Datei.
- Diese Registrierdatenbankvariable verweist auf die vom Hersteller bereitgestellte XBSA-Bibliothek. Unter AIX muss die Einstellung das gemeinsam genutzte Objekt enthalten, sofern es nicht den Namen shr.o hat. Für HP-UX, Solaris und Windows wird der Name des gemeinsam genutzten Objekts nicht benötigt. Wenn zum Beispiel NetWorker Business Suite Module für DB2 von Legato verwendet werden soll, muss diese Registrierdatenbankvariable wie folgt eingestellt werden:

  ```
  db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"
  ```

  Die XBSA-Schnittstelle kann mit dem Befehl BACKUP DATABASE oder RESTORE DATABASE aufgerufen werden. Beispiel:

  ```
  db2 backup db sample use XBSA
  db2 restore db sample use XBSA
  ```

Kapitel 20. Konfigurationsparameter

Bei der Erstellung einer DB2-Datenbankinstanz oder -Datenbank wird eine entsprechende Konfigurationsdatei mit Standardparameterwerten erstellt. Diese Parameterwerte können zur Verbesserung der Leistung und anderer Merkmale der Instanz oder der Datenbank geändert werden.

Der vom Datenbankmanager auf der Basis von Standardwerten für die Parameter zugeordnete Plattenspeicherplatz und der Hauptspeicher können für Ihre Anforderungen in einigen Fällen ausreichend sein. In einigen Situationen wird jedoch die maximale Leistung bei Verwendung dieser Standardwerte möglicherweise nicht erreicht.

Konfigurationsdateien enthalten Parameter, die Werte definieren, wie zum Beispiel die den DB2-Datenbankprodukten und einzelnen Datenbanken zugeordneten Ressourcen und die Diagnoseebene. Es gibt zwei Arten von Konfigurationsdateien:

- Die Konfigurationsdatei des Datenbankmanagers für jede DB2-Instanz
- Die Datenbankkonfigurationsdatei für jede einzelne Datenbank

Eine *Konfigurationsdatei des Datenbankmanagers* wird erstellt, wenn eine DB2-Instanz erstellt wird. Die in ihr enthaltenen Parameter betreffen Systemressourcen auf Instanzebene, die von keiner Datenbank abhängig sind, die zu dieser Instanz gehört. Die Werte vieler dieser Parameter können von den Systemstandardwerten zur Leistungsoptimierung oder Kapazitätserhöhung abhängig von der Konfiguration des Systems geändert werden.

Darüber hinaus gibt es auch für jede Clientinstallation eine Konfigurationsdatei des Datenbankmanagers. Diese Datei enthält Informationen über den Client Enabler für eine bestimmte Workstation. Eine Untergruppe der für einen Server verfügbaren Parameter gilt für den Client.

Konfigurationsparameter des Datenbankmanagers werden in einer Datei mit dem Namen `db2systm` gespeichert. Diese Datei wird erstellt, wenn die Instanz des Datenbankmanagers erstellt wird. In Linux- und UNIX-Umgebungen befindet sich diese Datei im Unterverzeichnis `sqllib` für die Instanz des Datenbankmanagers. Unter Windows ist die Standardposition dieser Datei das Instanzunterverzeichnis des Verzeichnisses `sqllib`. Wenn die Variable DB2INSTPROF definiert ist, befindet sich die Datei im Unterverzeichnis `instance` des Verzeichnisses, das durch die Variable DB2INSTPROF angegeben wird.

In Version 9.5 wird der implizite Standardwert von DB2INSTPROF auf der globalen Ebene (db2set -g) an der neuen Position (siehe unten) gespeichert, auch wenn DB2INSTPROF nicht definiert ist:

- In Vista-Umgebungen: `ProgramData\IBM\DB2\<DB2COPYNAME>`
- In Windows 2003/XP-Umgebungen: `Dokumente und Einstellungen\All Users\Application Data\IBM\DB2\<Name_der_Kopie>`

Andere Profilregistrierdatenbankvariablen, die angeben, wo sich Laufzeitdatendateien befinden sollen, müssen den Wert von DB2INSTPROF abfragen. Dabei handelt es sich um die folgenden Variablen:

- DB2CLINIPATH
- DIAGPATH

- SPM_LOG_PATH

Für Datenbanken, die mit einer Version vor Version 8.2 erstellt werden, werden Datenbankkonfigurationsparameter in einer Datei mit dem Namen SQLDBCON gespeichert. Für Datenbanken, die mit einer Version ab Version 8.2 erstellt werden, werden alle Datenbankkonfigurationsparameter in einer Datei mit dem Namen SQLDBCONF gespeichert. Diese Dateien können nicht direkt editiert werden, sondern lediglich mithilfe einer bereitgestellten Anwendungsprogrammierschnittstelle (API) oder mit einem Tool, das diese API aufruft, geändert oder angezeigt werden.

In einer Umgebung mit partitionierten Datenbanken befindet sich diese Datei in einem gemeinsamen Dateisystem, sodass alle Datenbankpartitionsserver Zugriff auf dieselbe Datei haben. Die Konfiguration des Datenbankmanagers ist auf allen Datenbankpartitionsservern identisch.

Die meisten Parameter haben entweder Einfluss darauf, wie viel Systemressourcen einer einzelnen Datenbankmanagerinstanz zugeordnet werden, oder sie konfigurieren die Einrichtung des Datenbankmanagers und der verschiedenen Kommunikationssubsysteme nach umgebungsspezifischen Aspekten. Darüber hinaus gibt es noch weitere Parameter, die nur der Information dienen und deren Werte nicht geändert werden können. Alle diese Parameter sind allgemein gültig und unabhängig von einzelnen Datenbanken, die unter dieser Datenbankmanagerinstanz gespeichert sind.

Eine *Konfigurationsdatei der Datenbank* wird erstellt, wenn eine Datenbank erstellt wird. Sie befindet sich dort, wo sich die Datenbank befindet. Pro Datenbank gibt es eine Konfigurationsdatei. Die in ihr enthaltenen Parameter geben neben anderen Merkmalen die Menge der Ressourcen an, die der zugehörigen Datenbank zugeordnet sind. Die Werte vieler dieser Parameter können zur Leistungsoptimierung und Kapazitätserhöhung geändert werden. Abhängig von der Art der Aktivitäten in einer bestimmten Datenbank können unterschiedliche Änderungen erforderlich sein.

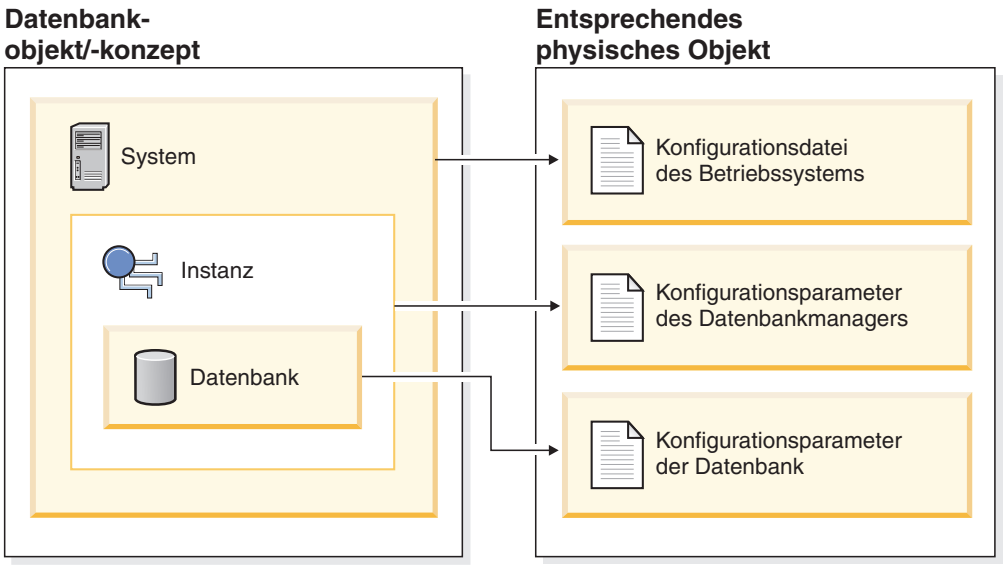


Abbildung 29. Beziehung zwischen Datenbankobjekten und Konfigurationsdateien

Konfigurieren des DB2-Datenbankmanagers mit Konfigurationsparametern

Der vom Datenbankmanager auf der Basis von Standardwerten für die Parameter zugeordnete Plattenspeicherplatz und der Hauptspeicher können für Ihre Anforderungen in einigen Fällen ausreichend sein. In einigen Situationen wird jedoch die maximale Leistung bei Verwendung dieser Standardwerte möglicherweise nicht erreicht.

Da sich die Standardwerte an Systemen orientieren, die über relativ kleine Hauptspeicherressourcen verfügen und als dedizierte Datenbankserver eingesetzt werden, müssen Sie diese Werte möglicherweise ändern, wenn Ihre Umgebung folgende Merkmale aufweist:

- Umfangreiche Datenbanken
- Große Anzahl von Verbindungen
- Hohe Leistungsanforderungen für eine bestimmte Anwendung
- Spezielle Abfrage-/Transaktionsauslastungen bzw. -typen

Jede Umgebung, die Transaktionen verarbeitet, hat einen oder mehrere spezifische, nur für sie geltende Aspekte. Diese Unterschiede können sich tief greifend auf die Leistung des Datenbankmanagers auswirken, wenn die Standardkonfiguration verwendet wird. Aus diesem Grund wird ausdrücklich empfohlen, die Konfiguration für die Umgebung zu optimieren.

Ein guter Ausgangspunkt für die Optimierung Ihrer Konfiguration ist die Verwendung des Konfigurationsadvisors bzw. des Befehls AUTOCONFIGURE, mit dessen Hilfe Parameterwerte auf der Basis Ihrer Antworten auf Fragen zu Auslastungsmerkmalen generiert werden.

Einige Konfigurationsparameter können auf AUTOMATIC gesetzt werden, sodass der Datenbankmanager diese Parameter automatisch so verwalten kann, dass sie den jeweils aktuellen Ressourcenbedarf berücksichtigen. Wenn Sie die Einstellung AUTOMATIC eines Konfigurationsparameters inaktivieren und gleichzeitig die aktuelle interne Einstellung beibehalten wollen, verwenden Sie das Schlüsselwort MANUAL im Befehl UPDATE DATABASE CONFIGURATION. Wenn der Datenbankmanager den Wert dieser Parameter aktualisiert, wird über die Befehle get db/dbm cfg show detail der neue Wert angezeigt.

Parameter für eine einzelne Datenbank werden in einer Konfigurationsdatei mit dem Namen `SQLDBCONF` gespeichert. Diese Datei wird zusammen mit anderen Steuerdateien für die Datenbank im Verzeichnis `SQLnnnnn` gespeichert, wobei `nnnnn` eine Nummer ist, die bei der Erstellung der Datenbank zugeordnet wurde. Jede Datenbank hat eine eigene Konfigurationsdatei, und die meisten Parameter in der Datei geben an, wie viele Ressourcen der betreffenden Datenbank zugeordnet werden. Die Datei enthält außerdem beschreibende Informationen sowie Markierungen, die den Status der Datenbank angeben.

Achtung: Wenn Sie die Datei `db2systm`, `SQLDBCON` oder `SQLDBCONF` mit anderen als den vom Datenbankmanager vorgesehenen Methoden editieren, wird die Datenbank möglicherweise unbrauchbar. Ändern Sie diese Dateien nicht mit anderen als den dokumentierten und vom Datenbankmanager unterstützten Methoden.

In einer Umgebung mit partitionierten Datenbanken ist für jede Datenbankpartition eine separate Datei `SQLDBCONF` vorhanden. Die Werte in der Datei

SQLDBCONF können in den einzelnen Datenbankpartitionen gleich oder unterschiedlich sein. Für eine homogene Umgebung ist es allerdings empfehlenswert, in allen Datenbankpartitionen die gleichen Werte für die Datenbankkonfigurationsparameter zu verwenden. In der Regel könnte ein Katalogknoten andere Einstellungen für die Datenbankkonfigurationsparameter benötigen, während die anderen Datenpartitionen abhängig vom Systemtyp und anderen Informationen wiederum andere Werte haben.

Aktualisieren von Konfigurationsparametern über den Befehlszeilenprozessor

Befehle zum Ändern der Einstellungen können wie folgt eingegeben werden:

Für Konfigurationsparameter des Datenbankmanagers:

- GET DATABASE MANAGER CONFIGURATION (oder GET DBM CFG)
- UPDATE DATABASE MANAGER CONFIGURATION (oder UPDATE DBM CFG)
- RESET DATABASE MANAGER CONFIGURATION (oder RESET DBM CFG) zum Zurücksetzen *aller* Parameter des Datenbankmanagers auf ihre Standardwerte
- AUTOCONFIGURE

Für Konfigurationsparameter der Datenbank:

- GET DATABASE CONFIGURATION (oder GET DB CFG)
- UPDATE DATABASE CONFIGURATION (oder UPDATE DB CFG)
- RESET DATABASE CONFIGURATION (oder RESET DB CFG) zum Zurücksetzen *aller* Datenbankparameter auf ihre Standardwerte
- AUTOCONFIGURE

Aktualisieren von Konfigurationsparametern mithilfe von Anwendungsprogrammierschnittstellen (APIs)

APIs können in einer Anwendung oder einem Programm in einer Hostprogrammiersprache aufgerufen werden. Rufen Sie die folgenden DB2-APIs auf, um Konfigurationsparameter anzuzeigen oder zu aktualisieren:

- db2AutoConfig - Auf den Konfigurationsadvisor zugreifen
- db2CfgGet - Konfigurationsparameter für den Datenbankmanager oder die Datenbank abrufen
- db2CfgSet - Konfigurationsparameter für den Datenbankmanager oder die Datenbank definieren

Aktualisieren von Konfigurationsparametern über den Konfigurationsassistenten

Der Konfigurationsassistent kann ebenfalls zur Einstellung der Konfigurationsparameter des Datenbankmanagers auf einem Client verwendet werden. Andere Parameter können online geändert werden. Diese werden als *online konfigurierbare Konfigurationsparameter* bezeichnet.

Anzeigen aktualisierter Konfigurationswerte

Für einige Konfigurationsparameter des Datenbankmanagers muss der Datenbankmanager gestoppt (db2stop) und anschließend erneut gestartet (db2start) werden, damit die neuen Parameterwerte in Kraft treten.

Für einige Datenbankparameter werden Änderungen erst wirksam, wenn die Datenbank erneut aktiviert bzw. vom Offlinestatus in den Onlinestatus versetzt wird. Dazu müssen zunächst alle Anwendungen ihre Verbindung zur Datenbank trennen. (Wenn die Datenbank aktiviert war bzw. vom

Offlinestatus in den Onlinestatus versetzt wurde, muss sie inaktiviert und anschließend erneut aktiviert werden.) Bei der Herstellung der ersten neuen Verbindung zur Datenbank werden die Änderungen wirksam.

Wenn Sie die Einstellung eines online konfigurierbaren Konfigurationsparameters des Datenbankmanagers ändern, während Sie mit einer Instanz verbunden sind, wendet der Befehl UPDATE DBM CFG die Änderung standardmäßig sofort an. Wenn die Änderung nicht sofort angewendet werden soll, verwenden Sie die Option DEFERRED im Befehl UPDATE DBM CFG.

Geben Sie folgende Befehle ein, um einen Konfigurationsparameter des Datenbankmanagers online zu ändern:

```
db2 attach to <instanzname>
db2 update dbm cfg using <parametername> <wert>
db2 detach
```

Bei Clients werden Änderungen an den Konfigurationsparametern des Datenbankmanagers wirksam, wenn der Client das nächste Mal die Verbindung zu einem Server herstellt.

Wenn Sie einen online konfigurierbaren Datenbankkonfigurationsparameter ändern, während Sie mit einer Datenbank verbunden sind, wird die Änderung standardmäßig, soweit möglich, online angewendet. Sie sollten jedoch beachten, dass einige Parameteränderungen aufgrund des Systemaufwands in Bezug auf die Speicherzuordnung möglicherweise eine relativ lange Zeit benötigen, um in Kraft zu treten. Zur Änderung von Konfigurationsparametern online über den Befehlszeilenprozessor ist eine Verbindung zur Datenbank erforderlich. Geben Sie folgende Befehle ein, um einen Datenbankkonfigurationsparameter online zu ändern:

```
db2 connect to <datenbankname>
db2 update db cfg using <parametername> <parameterwert>
db2 connect reset
```

Jedem online konfigurierbaren Konfigurationsparameter ist eine *Weitergabeklasse* zugeordnet. Die Weitergabeklasse gibt an, wann Sie damit rechnen können, dass eine Änderung an dem Konfigurationsparameter in Kraft tritt. Es gibt drei Weitergabeklassen:

- **Sofort:** Gibt Parameterwerte an, die sich sofort beim Aufruf des Befehls oder der API ändern. Zum Beispiel hat der Parameter *diaglevel* die Weitergabeklasse 'Sofort'.
- **Anweisungsgrenzwert:** Gibt Parameter an, die sich bei Anweisungsgrenzen und anweisungsähnlichen Grenzen ändern. Wenn Sie zum Beispiel den Wert des Parameters *sortheap* ändern, beginnen alle neuen Anforderungen mit der Verwendung des neuen Werts.
- **Transaktionsgrenzwert:** Gibt Parameter an, die sich bei Transaktionsgrenzen ändern. Zum Beispiel wird ein neuer Wert für den Parameter *dl_expint* nach einer Anweisung COMMIT aktualisiert.

Obwohl die neuen Parameterwerte möglicherweise nicht sofort in Kraft treten, werden beim Anzeigen der Parametereinstellungen (mithilfe des Befehls GET DATABASE MANAGER CONFIGURATION bzw. GET DATABASE CONFIGURATION) stets die zuletzt aktualisierten Werte angezeigt. Wenn Sie die Klausel SHOW DETAIL in diesen Befehlen zum Anzeigen der Parametereinstellungen angeben, werden sowohl die zuletzt aktualisierten Werte als auch die im Hauptspeicher befindlichen Werte angezeigt.

Rebind von Anwendungen nach dem Aktualisieren von Datenbankkonfigurationsparametern

Das Ändern einiger Konfigurationsparameter der Datenbank kann den Zugriffsplan beeinflussen, der vom SQL- und XQuery-Optimierungsprogramm ausgewählt wird. Nach der Änderung eines solchen Parameters sollten Sie in Betracht ziehen, einen Rebind für die Anwendungen durchzuführen, um sicherzustellen, dass der beste Zugriffsplan für die SQL- und XQuery-Anweisungen verwendet wird. Alle Parameter, die online modifiziert werden (z. B. durch den Befehl UPDATE DATABASE CONFIGURATION IMMEDIATE) veranlassen das SQL- und XQuery-Optimierungsprogramm, neue Zugriffspläne für neue Abfrageanweisungen auszuwählen. Allerdings wird der Abfrageanweisungscache nicht von vorhandenen Einträgen bereinigt. Zur Bereinigung des Inhalts des Abfragecache verwenden Sie die Anweisung FLUSH PACKAGE CACHE.

Anmerkung: Eine Reihe von Konfigurationsparametern (z. B. *userexit*) verfügen laut Beschreibung in der Hilfe und anderer DB2-Dokumentation über die zulässigen Werte „Yes" oder „No" bzw. „On" oder „Off". Zur Vermeidung von Unklarheiten sei hier angemerkt, dass „Yes" als äquivalent zu „On" und „No" als äquivalent zu „Off" anzusehen sind.

Konfigurationsparameter - Zusammenfassung

In den folgenden Tabellen sind die Parameter der Konfigurationsdateien des Datenbankmanagers und der Datenbank für Datenbankserver aufgeführt. Beachten Sie beim Ändern der Konfigurationsparameter des Datenbankmanagers und der Datenbank die detaillierten Informationen zu den einzelnen Parametern. Informationen zu spezifischen Betriebsumgebungen mit Standardwerten sind in jeder Parameterbeschreibung enthalten.

Übersicht über die Konfigurationsparameter des Datenbankmanagers

Für einige Konfigurationsparameter des Datenbankmanagers muss der Datenbankmanager gestoppt (db2stop) und erneut gestartet (db2start) werden, um die neuen Parameterwerte in Kraft zu setzen. Andere Parameter können online geändert werden. Diese werden als *online konfigurierbare Konfigurationsparameter* (Onl. kfg.) bezeichnet. Wenn Sie die Einstellung eines online konfigurierbaren Konfigurationsparameters des Datenbankmanagers ändern, während Sie mit einer Instanz verbunden sind, wird durch das Standardverhalten des Befehls UPDATE DBM CFG die Änderung unverzüglich angewendet. Wenn die Änderung nicht sofort angewendet werden soll, verwenden Sie die Option **DEFERRED** im Befehl UPDATE DBM CFG.

Die Spalte „Auto." in der folgenden Tabelle gibt an, ob der Parameter das Schlüsselwort AUTOMATIC im Befehl UPDATE DBM CFG unterstützt.

Bei der Aktualisierung eines Parameters auf den Wert AUTOMATIC ist es außerdem möglich, einen Anfangswert und das Schlüsselwort AUTOMATIC anzugeben. Beachten Sie hierbei, dass der Wert bei jedem Parameter eine andere Bedeutung haben kann und dass dieser in bestimmten Fällen nicht gilt. Vor der Angabe eines Werts sollten Sie die Dokumentation zu dem jeweiligen Parameter lesen, um festzustellen, was dieser Wert bedeutet. Im folgenden Beispiel wird **num_poolagents**

auf den Wert AUTOMATIC aktualisiert, und der Datenbankmanager verwendet den Wert 20 als minimale Anzahl für inaktive Agenten, die in einem Pool zusammengefasst werden sollen:

```
db2 update dbm cfg using num_poolagents 20 automatic
```

Zur Inaktivierung der AUTOMATIC-Funktion kann der Parameter auf einem bestimmten Wert aktualisiert oder das Schlüsselwort MANUAL verwendet werden. Wenn ein Parameter auf MANUAL aktualisiert wird, wird der Parameter nicht mehr automatisch festgelegt, sondern auf den aktuellen Wert gesetzt (wie in der Spalte `Aktueller Wert` der Ausgabe des Befehls GET DBM CFG SHOW DETAIL oder GET DB CFG SHOW DETAIL angezeigt).

In der Spalte „Leistungsrelevanz" (Leist.-Relev.) ist vermerkt, in welchem relativen Ausmaß sich jeder Parameter in Bezug auf die Systemleistung auswirken kann. Diese Spalte kann allerdings nicht für alle Arten von Umgebungen gültige Angaben enthalten. Sie sollten die Informationen als allgemeine Hinweise betrachten.

- **Hoch:** Gibt an, dass ein Parameter wesentliche Auswirkungen auf die Leistung haben kann. Die Werte für diese Parameter müssen sehr eingehend überlegt werden. In einigen Fällen kann das bedeuten, dass Sie die vorgegebenen Standardwerte übernehmen sollten.
- **Mittel:** Gibt an, dass der Parameter einige Auswirkungen auf die Leistung haben kann. Sie sollten von Ihrer spezifischen Umgebung und Ihren Anforderungen ausgehend entscheiden, wie viel Aufwand in die Optimierung dieser Parameter investiert werden sollte.
- **Niedrig:** Gibt an, dass der Parameter weniger allgemeine bzw. weniger bedeutende Auswirkungen auf die Leistung hat.
- **Keine:** Gibt an, dass der Parameter keine direkten Auswirkungen auf die Leistung hat. Da diese Parameter nicht leistungsrelevant sind, müssen sie nicht optimiert werden. Sie können jedoch in anderer Hinsicht für die Systemkonfiguration von Bedeutung sein, zum Beispiel zur Aktivierung der Kommunikationsunterstützung.

Die Spalten „Token", „Tokenwert" und „Datentyp" stellen Informationen bereit, die Sie beim Aufrufen der API db2CfgGet oder db2CfgSet benötigen. Diese Informationen enthalten Kennungen für Konfigurationsparameter, Einträge für das Element **token** in der Datenstruktur db2CfgParam und Datentypen für Werte, die an die Struktur übergeben werden.

Tabelle 66. Konfigurierbare Konfigurationsparameter des Datenbankmanagers

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
agent_stack_sz	Nein	Nein	Niedrig	SQLF_KTN_AGENT_STACK_SZ	61	Uint16	„agent_stack_sz - Größe des Agentenstacks" auf Seite 529
agentpri	Nein	Nein	Hoch	SQLF_KTN_AGENTPRI	26	Sint16	„agentpri - Agentenpriorität" auf Seite 531
aslheapsz	Nein	Nein	Hoch	SQLF_KTN_ASLHEAPSZ	15	Uint32	„aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 532
audit_buf_sz	Nein	Nein	Hoch	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32	„audit_buf_sz - Prüfpuffergröße" auf Seite 534
authentication[1]	Nein	Nein	Niedrig	SQLF_KTN_AUTHENTICATION	78	Uint16	„authentication - Authentifizierungstyp" auf Seite 535
catalog_noauth	Ja	Nein	Keine	SQLF_KTN_CATALOG_NOAUTH	314	Uint16	„catalog_noauth - Katalogisieren ohne Berechtigung zulässig" auf Seite 536

Tabelle 66. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
clnt_krb_plugin	Nein	Nein	Keine	SQLF_KTN_CLNT_KRB_ PLUGIN	812	char(33)	„clnt_krb_plugin - Client-Kerberos-Plug-in" auf Seite 537
clnt_pw_plugin	Nein	Nein	Keine	SQLF_KTN_CLNT_PW_ PLUGIN	811	char(33)	„clnt_pw_plugin - Client-Plug-in für Benutzer-ID und Kennwort" auf Seite 537
cluster_mgr	Nein	Nein	Keine	SQLF_KTN_CLUSTER_MGR	920	char(262)	„cluster_mgr - Name des Cluster-Managers" auf Seite 538
comm_bandwidth	Ja	Nein	Mittel	SQLF_KTN_COMM_ BANDWIDTH	307	float	„comm_bandwidth - Kommunikationsbandbreite" auf Seite 538
conn_elapse	Ja	Nein	Mittel	SQLF_KTN_CONN_ELAPSE	508	Uint16	„conn_elapse - Antwortzeit für Verbindung" auf Seite 539
cpuspeed	Ja	Nein	Hoch	SQLF_KTN_CPUSPEED	42	float	„cpuspeed - CPU-Geschwindigkeit" auf Seite 539
dft_account_str	Ja	Nein	Keine	SQLF_KTN_DFT_ ACCOUNT_STR	28	char(25)	„dft_account_str - Standardzeichen-folge für Abrechnung" auf Seite 540
dft_monswitches • **dft_mon_bufpool** • **dft_mon_lock** • **dft_mon_sort** • **dft_mon_stmt** • **dft_mon_table** • **dft_mon_timestamp** • **dft_mon_uow**	Ja	Nein	Mittel	SQLF_KTN_DFT_ MONSWITCHES[2] • SQLF_KTN_DFT_MON_ BUFPOOL • SQLF_KTN_DFT_MON_LOCK • SQLF_KTN_DFT_MON_SORT • SQLF_KTN_DFT_MON_STMT • SQLF_KTN_DFT_MON_ TABLE • SQLF_KTN_DFT_MON_ TIMESTAMP • SQLF_KTN_DFT_MON_ UOW	29 • 33 • 34 • 35 • 31 • 32 • 36 • 30	Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16	„dft_monswitches - Monitorschalter des Standarddatenbanksystems" auf Seite 541
dftdbpath	Ja	Nein	Keine	SQLF_KTN_DFTDBPATH	27	char(215)	„dftdbpath - Standarddatenbankpfad" auf Seite 542
diaglevel	Ja	Nein	Niedrig	SQLF_KTN_DIAGLEVEL	64	Uint16	„diaglevel - Aufzeichnungsebene bei Fehlerdiagnose" auf Seite 543
diagpath	Ja	Nein	Keine	SQLF_KTN_DIAGPATH	65	char(215)	„diagpath - Verzeichnispfad für Diagnosedaten" auf Seite 544
dir_cache	Nein	Nein	Mittel	SQLF_KTN_DIR_CACHE	40	Uint16	„dir_cache - Verzeichniscacheunterstützung" auf Seite 545
discover[3]	Nein	Nein	Mittel	SQLF_KTN_DISCOVER	304	Uint16	„discover - Discovery-Modus" auf Seite 546
discover_inst	Ja	Nein	Niedrig	SQLF_KTN_DISCOVER_INST	308	Uint16	„discover_inst - Discovery-Serverinstanz" auf Seite 547
fcm_num_buffers	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_ BUFFERS	503	Uint32	„fcm_num_buffers - Anzahl FCM-Puffer" auf Seite 548
fcm_num_channels	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_ CHANNELS	902	Uint32	„fcm_num_channels - Anzahl FCM-Kanäle" auf Seite 549
fed_noauth	Ja	Nein	Keine	SQLF_KTN_FED_NOAUTH	806	Uint16	„fed_noauth - Authentifizierung bei Servern mit föderierten Datenbanken umgehen" auf Seite 550
federated	Ja	Nein	Mittel	SQLF_KTN_FEDERATED	604	Sint16	„federated - Unterstützung für Systeme föderierter Datenbanken" auf Seite 550
federated_async	Ja	Ja	Mittel	SQLF_KTN_FEDERATED_ ASYNC	849	Sint32	„federated_async - Maximale ATQs pro Abfrage (Konfigurationsparameter)" auf Seite 550

Tabelle 66. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
fenced_pool	Ja	Ja	Mittel	SQLF_KTN_FENCED_POOL	80	Sint32	„fenced_pool - Maximale Anzahl abgeschirmter Prozesse" auf Seite 551
group_plugin	Nein	Nein	Keine	SQLF_KTN_GROUP_PLUGIN	810	char(33)	„group_plugin - Gruppen-Plug-in" auf Seite 553
health_mon	Ja	Nein	Niedrig	SQLF_KTN_HEALTH_MON	804	Uint16	„health_mon - Überwachung mit dem Diagnosemonitor" auf Seite 553
indexrec[4]	Ja	Nein	Mittel	SQLF_KTN_INDEXREC	20	Uint16	„indexrec - Zeitpunkt für Indexneuerstellung" auf Seite 554
instance_memory	Ja	Ja	Mittel	SQLF_KTN_INSTANCE_ MEMORY	803	Uint64	„instance_memory - Instanzspeicher" auf Seite 556
intra_parallel	Nein	Nein	Hoch	SQLF_KTN_INTRA_PARALLEL	306	Sint16	„intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 559
java_heap_sz	Nein	Nein	Hoch	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32	„java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter" auf Seite 560
jdk_path	Nein	Nein	Keine	SQLF_KTN_JDK_PATH	311	char(255)	„jdk_path - Installationspfad für Software Developer's Kit für Java" auf Seite 561
keepfenced	Nein	Nein	Mittel	SQLF_KTN_KEEPFENCED	81	Uint16	„keepfenced - Abgeschirmten Prozess beibehalten" auf Seite 561
local_gssplugin	Nein	Nein	Keine	SQLF_KTN_LOCAL_ GSSPLUGIN	816	char(33)	„local_gssplugin - GSS-API-Plug-in für lokale Berechtigung auf Instanzebene" auf Seite 562
max_connections	Ja	Ja	Mittel	SQLF_KTN_MAX_ CONNECTIONS	802	Sint32	„max_connections - Maximale Anzahl von Clientverbindungen" auf Seite 563
max_connretries	Ja	Nein	Mittel	SQLF_KTN_MAX_ CONNRETRIES	509	Uint16	„max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten" auf Seite 564
max_coordagents	Ja	Ja	Mittel	SQLF_KTN_MAX_ COORDAGENTS	501	Sint32	„max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 564
max_querydegree	Ja	Nein	Hoch	SQLF_KTN_MAX_ QUERYDEGREE	303	Sint32	„max_querydegree - max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 565
max_time_diff	Nein	Nein	Mittel	SQLF_KTN_MAX_TIME_DIFF	510	Uint16	„max_time_diff - Maximale Zeitdifferenz zwischen Knoten" auf Seite 566
mon_heap_sz	Ja	Ja	Niedrig	SQLF_KTN_MON_HEAP_SZ	79	Uint16	„mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor" auf Seite 568
notifylevel	Ja	Nein	Niedrig	SQLF_KTN_NOTIFYLEVEL	605	Sint16	„notifylevel - Benachrichtigungsstufe" auf Seite 570
num_initagents	Nein	Nein	Mittel	SQLF_KTN_NUM_ INITAGENTS	500	Uint32	„num_initagents - Anfangswert für die Anzahl Agenten im Pool" auf Seite 571
num_initfenced	Nein	Nein	Mittel	SQLF_KTN_NUM_ INITFENCED	601	Sint32	„num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse" auf Seite 571
num_poolagents	Ja	Ja	Hoch	SQLF_KTN_NUM_ POOLAGENTS	502	Sint32	„num_poolagents - Agentenpoolgröße" auf Seite 572
numdb	Nein	Nein	Niedrig	SQLF_KTN_NUMDB	6	Uint16	„numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und System i-Datenbanken" auf Seite 573
query_heap_sz	Nein	Nein	Mittel	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32	„query_heap_sz - Größe des Abfragezwischenspeichers" auf Seite 574

Tabelle 66. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
resync_interval	Nein	Nein	Keine	SQLF_KTN_RESYNC_ INTERVAL	68	Uint16	„resync_interval - Intervall für Transaktionsresynchronisation" auf Seite 575
rqrioblk	Nein	Nein	Hoch	SQLF_KTN_RQRIOBLK	1	Uint16	„rqrioblk - E/A-Blockgröße für Clients" auf Seite 576
sheapthres	Nein	Nein	Hoch	SQLF_KTN_SHEAPTHRES	21	Uint32	„sheapthres - Schwellenwert für Sortierspeicher" auf Seite 577
spm_log_file_sz	Nein	Nein	Niedrig	SQLF_KTN_SPM_LOG_FILE_ SZ	90	Sint32	„spm_log_file_sz - Protokolldatei-größe für SPM" auf Seite 579
spm_log_path	Nein	Nein	Mittel	SQLF_KTN_SPM_LOG_PATH	313	char(226)	„spm_log_path - Protokolldateipfad für SPM" auf Seite 579
spm_max_resync	Nein	Nein	Niedrig	SQLF_KTN_SPM_MAX_ RESYNC	91	Sint32	„spm_max_resync - Maximale Anzahl von SPM-Resynchronisationsagenten" auf Seite 580
spm_name	Nein	Nein	Keine	SQLF_KTN_SPM_NAME	92	char(8)	„spm_name - Name des Synchronisationspunktmanagers" auf Seite 580
srvcon_auth	Nein	Nein	Keine	SQLF_KTN_SRVCON_AUTH	815	Uint16	„srvcon_auth - Authentifizierungstyp für ankom-mende Verbindungen auf dem Server" auf Seite 581
srvcon_gssplugin_list	Nein	Nein	Keine	SQLF_KTN_SRVCON_ GSSPLUGIN_ LIST	814	char(256)	„srvcon_gssplugin_list - Liste der GSS-API-Plug-ins für ankommende Verbindungen auf dem Server" auf Seite 581
srv_plugin_mode	Nein	Nein	Keine	SQLF_KTN_SRV_PLUGIN_ MODE	809	Uint16	„srv_plugin_mode - Server-Plug-in-Modus" auf Seite 582
srvcon_pw_plugin	Nein	Nein	Keine	SQLF_KTN_SRVCON_PW_ PLUGIN	813	char(33)	„srvcon_pw_plugin - Plug-in für Benutzer-ID/Kennwort für ankom-mende Verbindungen auf dem Server" auf Seite 582
start_stop_time	Ja	Nein	Niedrig	SQLF_KTN_START_STOP_ TIME	511	Uint16	„start_stop_time - Zeitlimit für DB2START und DB2STOP" auf Seite 583
svcename	Nein	Nein	Keine	SQLF_KTN_SVCENAME	24	char(14)	„svcename - TCP/IP-Servicename" auf Seite 584
sysadm_group	Nein	Nein	Keine	SQLF_KTN_SYSADM_ GROUP	39	char(128)	„sysadm_group - SYSADM-Gruppenname" auf Seite 584
sysctrl_group	Nein	Nein	Keine	SQLF_KTN_SYSCTRL_ GROUP	63	char(128)	„sysctrl_group - SYSCTRL-Gruppenname" auf Seite 585
sysmaint_group	Nein	Nein	Keine	SQLF_KTN_SYSMAINT_ GROUP	62	char(128)	„sysmaint_group - SYSMAINT-Gruppenname" auf Seite 586
sysmon_group	Nein	Nein	Keine	SQLF_KTN_SYSMON GROUP	808	char(128)	„sysmon_group - Berechtigungs-gruppenname für Systemmonitor" auf Seite 586
tm_database	Nein	Nein	Keine	SQLF_KTN_TM_DATABASE	67	char(8)	„tm_database - Name für Transaktionsmanagerdatenbank" auf Seite 587
tp_mon_name	Nein	Nein	Keine	SQLF_KTN_TP_MON_NAME	66	char(19)	„tp_mon_name - Name des Transaktionsprozessormonitors" auf Seite 588
trust_allclnts[5]	Nein	Nein	Keine	SQLF_KTN_TRUST_ALLCLNTS	301	Uint16	„trust_allclnts - Alle Clients akzeptieren" auf Seite 589
trust_clntauth	Nein	Nein	Keine	SQLF_KTN_TRUST_ CLNTAUTH	302	Uint16	„trust_clntauth - Authentifizierung gesicherter Clients" auf Seite 590
util_impact_lim	Ja	Nein	Hoch	SQLF_KTN_UTIL_IMPACT_ LIM	807	Uint32	„util_impact_lim - Richtlinie für Instanzauslastungswirkung" auf Seite 591

Tabelle 66. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen

Anmerkung:

1. Die gültigen Werte sind in sqlenv.h definiert.
2.

```
Bit 1 (xxxx xxx1): dft_mon_uow
Bit 2 (xxxx xx1x): dft_mon_stmt
Bit 3 (xxxx x1xx): dft_mon_table
Bit 4 (xxxx 1xxx): dft_mon_buffpool
Bit 5 (xxx1 xxxx): dft_mon_lock
Bit 6 (xx1x xxxx): dft_mon_sort
Bit 7 (x1xx xxxx): dft_mon_timestamp
```

3. Gültige Werte (in sqlutil.h definiert) sind:

```
SQLF_DSCVR_KNOWN (1)
SQLF_DSCVR_SEARCH (2)
```

4. Gültige Werte (in sqlutil.h definiert) sind:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
```

5. Gültige Werte (in sqlutil.h definiert) sind:

```
SQLF_TRUST_ALLCLNTS_NO (0)
SQLF_TRUST_ALLCLNTS_YES (1)
SQLF_TRUST_ALLCLNTS_DRDAONLY (2)
```

Tabelle 67. Informative Konfigurationsparameter des Datenbankmanagers

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
nodetype[1]	SQLF_KTN_NODETYPE	100	Uint16	„nodetype - Knotenart des Systems" auf Seite 569
release	SQLF_KTN_RELEASE	101	Uint16	„release - Release-Level der Datenbankkonfiguration" auf Seite 575

Anmerkung:

1. Gültige Werte (in sqlutil.h definiert) sind:

```
SQLF_NT_STANDALONE (0)
SQLF_NT_SERVER (1)
SQLF_NT_REQUESTOR (2)
SQLF_NT_STAND_REQ (3)
SQLF_NT_MPP (4)
SQLF_NT_SATELLITE (5)
```

Übersicht über die Konfigurationsparameter der Datenbank

In der folgenden Tabelle sind die Parameter der Konfigurationsdatei für die Datenbank aufgeführt. Wenn Sie Konfigurationsparameter der Datenbank ändern möchten, lesen Sie die detaillierten Informationen zu den Parametern.

Für einige Datenbankkonfigurationsparameter werden Änderungen erst dann wirksam, wenn die Datenbank erneut aktiviert wird. Dazu müssen zunächst alle Anwendungen ihre Verbindung zur Datenbank trennen. (Wenn die Datenbank aktiviert war, muss sie inaktiviert und anschließend erneut aktiviert werden.) Die Änderungen werden wirksam, sobald das nächste Mal eine Verbindung zur Datenbank hergestellt wird. Andere Parameter können online geändert werden. Diese werden als *online konfigurierbare Konfigurationsparameter* bezeichnet.

Eine Beschreibung der Spalten „Auto.", „Leist.-Relev.", „Token", „Tokenwert" und „Datentyp" finden Sie im Abschnitt mit der Übersicht über die Konfigurationsparameter des Datenbankmanagers.

Das Schlüsselwort AUTOMATIC wird auch im Befehl UPDATE DB CFG unterstützt. Im folgenden Beispiel wird der Parameter **database_memory** auf den Wert AUTOMATIC aktualisiert, und der Datenbankmanager verwendet den Wert 20000 als Anfangswert, wenn weitere Änderungen an diesem Parameter vorgenommen werden:

```
db2 update db cfg using for sample using database_memory 20000 automatic
```

Ab Version 9.5 können Sie Datenbankkonfigurationsparameterwerte auf einigen oder allen Plattformen aktualisieren und zurücksetzen. Dazu müssen Sie nicht den Befehl db2_all absetzen oder jede Partition einzeln aktualisieren oder zurücksetzen. Detaillierte Informationen finden Sie in Konfigurieren von Datenbanken über mehrere Partitionen.

Tabelle 68. Konfigurierbare Konfigurationsparameter für die Datenbank

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
alt_collate	Nein	Nein	Keine	SQLF_DBTN_ALT_COLLATE	809	Uint32	„alt_collate - Alternative Sortierfolge" auf Seite 592
applheapsz	Ja	Ja	Mittel	SQLF_DBTN_APPLHEAPSZ	51	Uint16	„applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 595
appl_memory	Ja	Ja	Mittel	SQLF_DBTN_APPL_MEMORY	904	Uint64	„appl_memory - Anwendungsspeicher (Konfigurationsparameter)" auf Seite 595
archretrydelay	Ja	Nein	Keine	SQLF_DBTN_ARCHRETRYDELAY	828	Uint16	„archretrydelay - Wiederholungsintervall für Archivierung bei Fehler" auf Seite 596
• **auto_maint** • **auto_db_backup** • **auto_tbl_maint** • **auto_runstats** • **auto_stats_prof** • **auto_stmt_stats** • **auto_prof_upd** • **auto_reorg**	Ja	Nein	Mittel	• SQLF_DBTN_AUTO_MAINT • SQLF_DBTN_AUTO_DB_BACKUP • SQLF_DBTN_AUTO_TBL_MAINT • SQLF_DBTN_AUTO_RUNSTATS • SQLF_DBTN_AUTO_STATS_PROF • SQLF_DBTN_AUTO_STMT_STATS • SQLF_DBTN_AUTO_PROF_UPD • SQLF_DBTN_AUTO_REORG	• 831 • 833 • 835 • 837 • 839 • 905 • 844 • 841	Uint16	„auto_maint - Automatische Verwaltung" auf Seite 597
auto_del_rec_obj	Ja	Nein	Mittel	SQLF_DBTN_AUTO_DEL_REC_OBJ	912	Uint16	„auto_del_rec_obj - Automatisches Löschen von Recovery-Objekten (Konfigurationsparameter)" auf Seite 597
autorestart	Ja	Nein	Niedrig	SQLF_DBTN_AUTO_RESTART	25	Uint16	„autorestart - Automatischer Neustart aktiviert" auf Seite 600
avg_appls	Ja	Ja	Hoch	SQLF_DBTN_AVG_APPLS	47	Uint16	„avg_appls - Durchschnittliche Anzahl aktiver Anwendungen" auf Seite 600
blk_log_dsk_ful	Ja	Nein	Keine	SQLF_DBTN_BLK_LOG_DSK_FUL	804	Uint16	„blk_log_dsk_ful - Bei voller Protokollplatte blockieren" auf Seite 601
catalogcache_sz	Ja	Nein	Mittel	SQLF_DBTN_CATALOGCACHE_SZ	56	Sint32	„catalogcache_sz - Katalogcachegröße" auf Seite 602
chngpgs_thresh	Nein	Nein	Hoch	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16	„chngpgs_thresh - Schwellenwert für geänderte Seiten" auf Seite 604

Tabelle 68. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
database_memory	Ja	Ja	Mittel	SQLF_DBTN_DATABASE_ MEMORY	803	Uint64	„database_memory - Größe des gemeinsam genutzten Datenbankspeichers" auf Seite 607
dbheap	Ja	Ja	Mittel	SQLF_DBTN_DB_HEAP	58	Uint64	„dbheap - Zwischenspeicher für Datenbank" auf Seite 610
db_mem_thresh	Ja	Nein	Niedrig	SQLF_DBTN_DB_MEM_ THRESH	849	Uint16	„db_mem_thresh - Schwellenwert für Datenbankspeicher" auf Seite 609
decflt_rounding	Nein	Nein	Keine	SQLF_DBTN_DECFLT_ ROUNDING	913	Unit16	„decflt_rounding - Rundungsmodus für dezimale Gleitkommawerte (Konfigurationsparameter)" auf Seite 612
dft_degree	Ja	Nein	Hoch	SQLF_DBTN_DFT_DEGREE	301	Sint32	„dft_degree - Grad der Parallelität" auf Seite 613
dft_extent_sz	Ja	Nein	Mittel	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32	„dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen" auf Seite 614
dft_loadrec_ses	Ja	Nein	Mittel	SQLF_DBTN_DFT_LOADREC_ SES	42	Sint16	„dft_loadrec_ses - Standardanzahl von Sitzungen für Recovery" auf Seite 615
dft_mttb_types	Nein	Nein	Keine	SQLF_DBTN_DFT_MTTB_ TYPES	843	Uint32	„dft_mttb_types - Standardtypen von verwalteten Tabellen zur Optimierung" auf Seite 615
dft_prefetch_sz	Ja	Ja	Mittel	SQLF_DBTN_DFT_PREFETCH_ SZ	40	Sint16	„dft_prefetch_sz - Standardwert für PREFETCHSIZE" auf Seite 616
dft_queryopt	Ja	Nein	Mittel	SQLF_DBTN_DFT_QUERYOPT	57	Sint32	„dft_queryopt - Standardabfrageoptimierungsklasse" auf Seite 617
dft_refresh_age	Nein	Nein	Mittel	SQLF_DBTN_DFT_REFRESH_ AGE	702	char(22)	„dft_refresh_age - Standardaktualisierungsalter" auf Seite 617
dft_sqlmathwarn	Nein	Nein	Keine	SQLF_DBTN_DFT_ SQLMATHWARN	309	Sint16	„dft_sqlmathwarn - Bei arithmetischen Ausnahmebedingungen fortsetzen" auf Seite 618
discover_db	Ja	Nein	Mittel	SQLF_DBTN_DISCOVER	308	Uint16	„discover_db - Discovery-Unterstützung für diese Datenbank" auf Seite 619
dlchktime	Ja	Nein	Mittel	SQLF_DBTN_DLCHKTIME	9	Uint32	„dlchktime - Zeitintervall für Prüfung von Deadlocks" auf Seite 620
dyn_query_mgmt	Nein	Nein	Niedrig	SQLF_DBTN_DYN_QUERY_ MGMT	604	Uint16	„dyn_query_mgmt - Dynamische SQL- und XQuery-Abfrageverwaltung" auf Seite 621
enable_xmlchar	Ja	Nein	Keine	SQLF_DBTN_ENABLE_ XMLCHAR	853	Uint32	„enable_xmlchar - Ermöglichen der Konvertierung in XML (Konfigurationsparameter)" auf Seite 621
failarchpath	Ja	Nein	Keine	SQLF_DBTN_FAILARCHPATH	826	char(243)	„failarchpath - Protokollarchivpfad für Funktionsübernahme" auf Seite 622
hadr_local_host	Nein	Nein	Keine	SQLF_DBTN_HADR_LOCAL_ HOST	811	char(256)	„hadr_local_host - Name des lokalen Hosts für HADR" auf Seite 623
hadr_local_svc	Nein	Nein	Keine	SQLF_DBTN_HADR_LOCAL_ SVC	812	char(41)	„hadr_local_svc - Lokaler HADR-Servicename" auf Seite 624

Tabelle 68. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
hadr_peer_window	Nein	Nein	Niedrig (siehe Anmerkung 4)	SQLF_DBTN_HADR_PEER_WINDOW	914	Uint32	„hadr_peer_window - HADR-Peerfenster (Konfigurationsparameter)" auf Seite 624
hadr_remote_host	Nein	Nein	Keine	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)	„hadr_remote_host - Name des fernen HADR-Hosts" auf Seite 625
hadr_remote_inst	Nein	Nein	Keine	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)	„hadr_remote_inst - HADR-Instanzname des fernen Servers" auf Seite 625
hadr_remote_svc	Nein	Nein	Keine	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)	„hadr_remote_svc - Ferner HADR-Servicename" auf Seite 626
hadr_syncmode	Nein	Nein	Keine	SQLF_DBTN_HADR_SYNCMODE	817	Uint32	„hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus" auf Seite 626
hadr_timeout	Nein	Nein	Keine	SQLF_DBTN_HADR_TIMEOUT	816	Uint32	„hadr_timeout - HADR-Zeitlimitwert" auf Seite 627
indexrec[2]	Ja	Nein	Mittel	SQLF_DBTN_INDEXREC	30	Uint16	„indexrec - Zeitpunkt für Indexneuerstellung" auf Seite 554
locklist	Ja	Ja	Hoch bei Einfluss auf die Eskalation	SQLF_DBTN_LOCK_LIST	704	Uint64	„locklist - Maximaler Speicher für Sperrenliste" auf Seite 630
locktimeout	Nein	Nein	Mittel	SQLF_DBTN_LOCKTIMEOUT	34	Sint16	„locktimeout - Zeitlimit für Sperren" auf Seite 633
logarchmeth1	Ja	Nein	Keine	SQLF_DBTN_LOGARCHMETH1	822	char(252)	„logarchmeth1 - Primäre Protokollarchivierungsmethode" auf Seite 635
logarchmeth2	Ja	Nein	Keine	SQLF_DBTN_LOGARCHMETH2	823	char(252)	„logarchmeth2 - Sekundäre Protokollarchivierungsmethode" auf Seite 636
logarchopt1	Ja	Nein	Keine	SQLF_DBTN_LOGARCHOPT1	824	char(243)	„logarchopt1 - Optionen für primäres Protokollarchiv" auf Seite 637
logarchopt2	Ja	Nein	Keine	SQLF_DBTN_LOGARCHOPT2	825	char(243)	„logarchopt2 - Optionen für sekundäres Protokollarchiv" auf Seite 638
logbufsz	Nein	Nein	Hoch	SQLF_DBTN_LOGBUFSZ	33	Uint16	„logbufsz - Protokollpuffergröße" auf Seite 638
logfilsiz	Nein	Nein	Mittel	SQLF_DBTN_LOGFIL_SIZ	92	Uint32	„logfilsiz - Protokolldateigröße" auf Seite 639
logindexbuild	Ja	Ja	Keine	SQLF_DBTN_LOGINDEXBUILD	818	Uint32	„logindexbuild - Erstellte Indexseiten protokollieren" auf Seite 640
logprimary	Nein	Nein	Mittel	SQLF_DBTN_LOGPRIMARY	16	Uint16	„logprimary - Anzahl primärer Protokolldateien" auf Seite 641
logretain[3]	Nein	Nein	Niedrig	SQLF_DBTN_LOG_RETAIN	23	Uint16	„logretain - Beibehalten von Protokollen aktivieren" auf Seite 643
logsecond	Ja	Nein	Mittel	SQLF_DBTN_LOGSECOND	17	Uint16	„logsecond - Anzahl sekundärer Protokolldateien" auf Seite 644
max_log	Ja	Ja		SQLF_DBTN_MAX_LOG	807	Uint16	„max_log - Maximale Protokollgröße pro Transaktion" auf Seite 645

Tabelle 68. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
maxappls	Ja	Ja	Mittel	SQLF_DBTN_MAXAPPLS	6	Uint16	„maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 645
maxfilop	Ja	Nein	Mittel	SQLF_DBTN_MAXFILOP	3	Uint16	„maxfilop - Maximale Anzahl offener Datenbankdateien pro Anwendung" auf Seite 646
maxlocks	Ja	Ja	Hoch bei Einfluss auf die Eskala-tion	SQLF_DBTN_MAXLOCKS	15	Uint16	„maxlocks - Maximale Anzahl von Sperren vor Eskalation" auf Seite 647
min_dec_div_3	Nein	Nein	Hoch	SQLF_DBTN_MIN_DEC_DIV_3	605	Sint32	„min_dec_div_3 - 3 Komma-stellen bei Dezimaldivision" auf Seite 649
mincommit	Ja	Nein	Hoch	SQLF_DBTN_MINCOMMIT	32	Uint16	„mincommit - Anzahl der Gruppencommits" auf Seite 651
mirrorlogpath	Nein	Nein	Niedrig	SQLF_DBTN_ MIRRORLOGPATH	806	char(242)	„mirrorlogpath - Pfad für Protokollspiegelung" auf Seite 652
newlogpath	Nein	Nein	Niedrig	SQLF_DBTN_NEWLOGPATH	20	char(242)	„newlogpath - Datenbank-protokollpfad ändern" auf Seite 653
num_db_backups	Ja	Nein	Keine	SQLF_DBTN_NUM_DB_ BACKUPS	601	Uint16	„num_db_backups - Anzahl der Datenbank-Backups" auf Seite 655
num_freqvalues	Ja	Nein	Niedrig	SQLF_DBTN_NUM_ FREQVALUES	36	Uint16	„num_freqvalues - Anzahl der häufigsten Werte" auf Seite 656
num_iocleaners	Nein	Ja	Hoch	SQLF_DBTN_NUM_ IOCLEANERS	37	Uint16	„num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen" auf Seite 657
num_ioservers	Nein	Ja	Hoch	SQLF_DBTN_NUM_ IOSERVERS	39	Uint16	„num_ioservers - Anzahl von E/A-Servern" auf Seite 658
num_log_span	Ja	Ja		SQLF_DBTN_NUM_LOG_ SPAN	808	Uint16	„num_log_span - Anzahl ver-wendeter Protokolldateien" auf Seite 659
num_quantiles	Ja	Nein	Niedrig	SQLF_DBTN_NUM_ QUANTILES	48	Uint16	„num_quantiles - Anzahl der Quantile für Spalten" auf Seite 660
numarchretry	Ja	Nein	Keine	SQLF_DBTN_ NUMARCHRETRY	827	Uint16	„numarchretry - Anzahl der Wiederholungen bei Fehler" auf Seite 661
overflowlogpath	Nein	Nein	Mittel	SQLF_DBTN_ OVERFLOWLOGPATH	805	char(242)	„overflowlogpath - Überlaufprotokollpfad" auf Seite 662
pckcachesz	Ja	Ja	Hoch	SQLF_DBTN_PCKCACHE_SZ	505	Uint32	„pckcachesz - Größe des Paketcache" auf Seite 663
rec_his_retentn	Nein	Nein	Keine	SQLF_DBTN_REC_HIS_ RETENTN	43	Sint16	„rec_his_retentn - Auf-bewahrungszeitraum für Recoveryprotokoll" auf Seite 666
self_tuning_mem	Ja	Nein	Hoch	SQLF_DBTN_SELF_TUNING_ MEM	848	Uint16	„self_tuning_mem - Speicher mit automatischer Leistungsoptimierung" auf Seite 668
seqdetect	Ja	Nein	Hoch	SQLF_DBTN_SEQDETECT	41	Uint16	„seqdetect - Markierung für Sequenzerkennung" auf Seite 669
sheapthres_shr	Ja	Ja	Hoch	SQLF_DBTN_SHEAPTHRES_ SHR	802	Uint32	„sheapthres_shr - Schwellen-wert für Sortierspeicher für gemeinsame Sortiervorgänge" auf Seite 670

Tabelle 68. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
softmax	Nein	Nein	Mittel	SQLF_DBTN_SOFTMAX	5	Uint16	„softmax - Recoverybereich und Intervall für bedingte Prüfpunkte" auf Seite 672
sortheap	Ja	Ja	Hoch	SQLF_DBTN_SORT_HEAP	52	Uint32	„sortheap - Sortierspeichergröße" auf Seite 673
stat_heap_sz	Ja	Ja	Niedrig	SQLF_DBTN_STAT_HEAP_SZ	45	Uint32	„stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 675
stmtheap	Ja	Ja	Mittel	SQLF_DBTN_STMT_HEAP	821	Uint32	„stmtheap - Größe des Anweisungszwischenspeichers" auf Seite 675
trackmod	Nein	Nein	Niedrig	SQLF_DBTN_TRACKMOD	703	Uint16	„trackmod - Geänderte Seiten protokollieren" auf Seite 677
tsm_mgmtclass	Ja	Nein	Keine	SQLF_DBTN_TSM_ MGMTCLASS	307	char(30)	„tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse" auf Seite 677
tsm_nodename	Ja	Nein	Keine	SQLF_DBTN_TSM_ NODENAME	306	char(64)	„tsm_nodename - TSM-Knotenname" auf Seite 677
tsm_owner	Ja	Nein	Keine	SQLF_DBTN_TSM_OWNER	305	char(64)	„tsm_owner - TSM-Eignername" auf Seite 678
tsm_password	Ja	Nein	Keine	SQLF_DBTN_TSM_PASSWORD	501	char(64)	„tsm_password - TSM-Kennwort" auf Seite 678
userexit	Nein	Nein	Niedrig	SQLF_DBTN_USER_EXIT	24	Uint16	„userexit - Benutzerexit aktivieren" auf Seite 679
util_heap_sz	Ja	Nein	Niedrig	SQLF_DBTN_UTIL_HEAP_SZ	55	Uint32	„util_heap_sz - Zwischenspeichergröße für Dienstprogramme" auf Seite 680
vendoropt	Ja	Nein	Keine	SQLF_DBTN_VENDOROPT	829	char(242)	„vendoropt - Lieferantenoptionen" auf Seite 680<
wlm_collect_int	Ja	Nein	Niedrig	SQLF_DBTN_WLM_COLLECT_ INT	907	Sint32	„wlm_collect_int - Workload-Management-Erfassungsintervall (Konfigurationsparameter)" auf Seite 681

Tabelle 68. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen

Anmerkung: Die Bits des Tokens SQLF_DBTN_AUTONOMIC_SWITCHES geben die Standardeinstellungen für eine Reihe von Konfigurationsparametern zur automatischen Verwaltung an. Im Folgenden sehen Sie die einzelnen Bits, aus denen dieser zusammengesetzte Parameter besteht:

1.
```
Default => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
Bit 2 off (xxxx xxxx xxxx xx0x): auto_db_backup
Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
Bit 5 off (xxxx xxxx xxx0 xxxx): auto_stats_prof
Bit 6 off (xxxx xxxx xx0x xxxx): auto_prof_upd
Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg
Bit 8 off (xxxx xxxx 0xxx xxxx): auto_storage
Bit 9 off (xxxx xxx0 xxxx xxxx): auto_stmt_stats
0 0 0 D

Maximum => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
Bit 2 off (xxxx xxxx xxxx xx1x): auto_db_backup
Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
Bit 6 off (xxxx xxxx xx1x xxxx): auto_prof_upd
Bit 7 off (xxxx xxxx x1xx xxxx): auto_reorg
Bit 8 off (xxxx xxxx 1xxx xxxx): auto_storage
Bit 9 off (xxxx xxx1 xxxx xxxx): auto_stmt_stats
0 1 F F
```

2. Gültige Werte (in sqlutil.h definiert) sind:
```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
SQLF_INX_REC_RESTART (2)
```

3. Gültige Werte (in sqlutil.h definiert) sind:
```
SQLF_LOGRETAIN_NO (0)
SQLF_LOGRETAIN_RECOVERY (1)
SQLF_LOGRETAIN_CAPTURE (2)
```

4. Wenn Sie den Parameter **hadr_peer_window** auf einen anderen Zeitwert als null setzen, kann die Primärdatenbank so aussehen, als ob sie bei Transaktionen blockiert, wenn sie sich im getrennten Peerzustand befindet, da sie auf eine Bestätigung von der Bereitschaftsdatenbank wartet, selbst wenn sie nicht mit der Bereitschaftsdatenbank verbunden ist.

Tabelle 69. Informative Konfigurationsparameter für die Datenbank

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
backup_pending	SQLF_DBTN_BACKUP_PENDING	112	Uint16	„backup_pending - Backup anstehend" auf Seite 601
codepage	SQLF_DBTN_CODEPAGE	101	Uint16	„codepage - Codepage für die Datenbank" auf Seite 604
codeset	SQLF_DBTN_CODESET	120	char(9)[1]	„codeset - Codierter Zeichensatz für die Datenbank" auf Seite 605
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)	„collate_info - Informationen zur Sortierfolge" auf Seite 605
country/region	SQLF_DBTN_COUNTRY	100	Uint16	„country/region - Gebietscode der Datenbank" auf Seite 606
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16	„database_consistent - Datenbank ist konsistent" auf Seite 606
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16	„database_level - Release-Level der Datenbank" auf Seite 606
hadr_db_role	SQLF_DBTN_HADR_DB_ROLE	810	Uint32	„hadr_db_role - Rolle der HADR-Datenbank" auf Seite 623
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16	„log_retain_status - Statusanzeiger für Beibehalten der Protokolle" auf Seite 634
loghead	SQLF_DBTN_LOGHEAD	105	char(12)	„loghead - Erste aktive Protokolldatei" auf Seite 640
logpath	SQLF_DBTN_LOGPATH	103	char(242)	„logpath - Pfad zu Protokolldateien" auf Seite 641
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16	„multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv" auf Seite 653

Tabelle 69. Informative Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
numsegs	SQLF_DBTN_NUMSEGS	122	Uint16	„numsegs - Standardanzahl von SMS-Containern" auf Seite 662
pagesize	SQLF_DBTN_PAGESIZE	846	Uint32	„pagesize - Standardseitengröße für die Datenbank" auf Seite 663
release	SQLF_DBTN_RELEASE	102	Uint16	„release - Release-Level der Datenbankkonfiguration" auf Seite 575
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16	„restore_pending - Restore anstehend" auf Seite 667
restrict_access	SQLF_DBTN_RESTRICT_ACCESS	852	Sint32	„restrict_access - Eingeschränkter Datenbankzugriff (Konfigurationsparameter)" auf Seite 667
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	Uint16	„rollfwd_pending - Aktualisierende Recovery anstehend" auf Seite 667
territory	SQLF_DBTN_TERRITORY	121	char(5)[2]	„territory - Datenbankgebiet" auf Seite 676
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16	„user_exit_status - Statusanzeiger für Benutzerexit" auf Seite 679

Anmerkung:

1. char(17) unter HP-UX, Linux und Solaris Operating Environment.
2. char(33) unter HP-UX, Linux und Solaris.

Übersicht über die Konfigurationsparameter für den DB2-Verwaltungsserver (DAS)

Tabelle 70. DAS-Konfigurationsparameter

Parameter	Parametertyp	Zusätzliche Informationen
authentication	Konfigurierbar	„authentication - DAS-Authentifizierungstyp" auf Seite 682
contact_host	Online konfigurierbar	„contact_host - Speicherposition der Liste mit Ansprechpartnern" auf Seite 682
das_codepage	Online konfigurierbar	„das_codepage - DAS-Codepage" auf Seite 683
das_territory	Online konfigurierbar	„das_territory - DAS-Gebiet" auf Seite 683
dasadm_group	Konfigurierbar	„dasadm_group - DASADM-Gruppenname" auf Seite 684
db2system	Online konfigurierbar	„db2system - Name des DB2-Serversystems" auf Seite 684
discover	Online konfigurierbar	„discover - DAS-Discovery-Modus" auf Seite 685
exec_exp_task	Konfigurierbar	„exec_exp_task - Verfallene Tasks ausführen" auf Seite 685
jdk_64_path	Online konfigurierbar	„jdk_64_path - Installationspfad für 64-Bit Software Developer's Kit für Java auf DAS" auf Seite 630
jdk_path	Online konfigurierbar	„jdk_path - Installationspfad für Software Developer's Kit für Java auf DAS" auf Seite 686
sched_enable	Konfigurierbar	„sched_enable - Schedulermodus" auf Seite 686
sched_userid	Informativ	„sched_userid - Benutzer-ID für Scheduler" auf Seite 687
smtp_server	Online konfigurierbar	„smtp_server - SMTP-Server" auf Seite 687
toolscat_db	Konfigurierbar	„toolscat_db - Toolskatalogdatenbank" auf Seite 688
toolscat_inst	Konfigurierbar	„toolscat_inst - Instanz der Toolskatalogdatenbank" auf Seite 688
toolscat_schema	Konfigurierbar	„toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 689

Abschnittsüberschriften für Konfigurationsparameter

Jede der Beschreibungen von Konfigurationsparametern enthält je nach Bedarf einige oder alle der folgenden Abschnittsüberschriften. In einigen Fällen schließen sie sich gegenseitig aus. Zum Beispiel müssen keine gültigen Werte angegeben werden, wenn der '[Bereich]' angegeben wird. In den meisten Fällen sind diese Überschriften selbsterklärend.

Tabelle 71. Beschreibung der Abschnittsüberschriften für Konfigurationsparameter

Abschnittsüberschrift	**Beschreibung und mögliche Werte**
Konfigurationstyp	Mögliche Werte: • Datenbankmanager • Datenbank • DB2-Verwaltungsserver
Gilt für	Falls zutreffend, werden unter dieser Überschrift die Datenservertypen aufgelistet, für die der Konfigurationsparameter gilt. Mögliche Werte: • Client • Datenbankserver mit lokalen und fernen Clients • Datenbankserver mit lokalen Clients • DB2-Verwaltungsserver • OLAP-Funktionen • Partitionierten Datenbankserver mit lokalen und fernen Clients • Partitionierten Datenbankserver mit lokalen und fernen Clients, wenn die Föderation aktiviert ist. • Satellitendatenbankserver mit lokalen Clients
Parametertyp	Mögliche Werte: • Konfigurierbar (Der Datenbankmanager muss erneut gestartet werden, um die Änderungen in Kraft zu setzen.) • Online konfigurierbar (Der Konfigurationsparameter kann dynamisch online aktualisiert werden, ohne den Datenbankmanager erneut zu starten.) • Informativ (Werte dienen nur der Information und können nicht aktualisiert werden.)
Standardwert [Bereich]	Falls zutreffend, werden unter dieser Überschrift der Standardwert und die möglichen Wertebereiche, einschließlich Nullwerten (NULL) oder automatische Einstellungen (AUTOMATIC), angegeben. Wenn sich der Bereich je nach Plattform unterscheidet, werden die Werte nach Plattform bzw. Plattformtyp (z. B. 32-Bit-Plattformen oder 64-Bit-Plattformen) getrennt aufgeführt. Beachten Sie, dass der Standardwert in den meisten Fällen nicht als Teil des Bereichs angegeben wird.
Maßeinheit	Falls zutreffend, wird unter dieser Überschrift die Maßeinheit der Werte angegeben. Mögliche Werte: • Byte • Zähler • Megabyte pro Sekunde • Millisekunden • Minuten • Seiten (4 KB) • Prozent • Sekunden
Gültige Werte	Falls zutreffend, werden unter dieser Überschrift gültige Werte angegeben. Diese Überschrift schließt sich mit der Überschrift 'Standardwert [Bereich]' gegenseitig aus.
Beispiele	Falls zutreffend, werden unter dieser Überschrift Beispiele angegeben.
Weitergabeklasse	Falls zutreffend, sind die möglichen Werte folgende: • Sofort • Anweisungsgrenzwert
Zuordnung	Falls zutreffend, wird unter dieser Überschrift angegeben, wann der Konfigurationsparameter vom Datenbankmanager zugeordnet wird.
Freigabe	Falls zutreffend, wird unter dieser Überschrift angegeben, wann der Konfigurationsparameter vom Datenbankmanager freigegeben wird.
Einschränkungen	Falls zutreffend, werden unter dieser Überschrift Einschränkungen aufgeführt, die für den Konfigurationsparameter gelten.
Begrenzungen	Falls zutreffend, werden unter dieser Überschrift Begrenzungen aufgeführt, die für den Konfigurationsparameter gelten.
Empfehlungen	Falls zutreffend, werden unter dieser Überschrift Empfehlungen gegeben, die für den Konfigurationsparameter gelten.
Hinweise	Falls zutreffend, werden unter dieser Überschrift Hinweise gegeben, die für den Konfigurationsparameter gelten.

Konfigurationsparameter mit Auswirkung auf die Anzahl von Agenten

Es gibt eine Reihe von Datenbankmanagerkonfigurationsparametern, die sich auf Datenbankagenten und deren Verwaltung beziehen.

Die folgenden Konfigurationsparameter des Datenbankmanagers legen fest, wie viele Datenbankagenten erstellt werden und wie sie verwaltet werden:

- Agentenpoolgröße (*num_poolagents*): Die Gesamtanzahl an inaktiven Agenten, die in einem Pool zusammengefasst werden sollen, die im System verfügbar gehalten werden. Der Standardwert für diesen Parameter ist 100, AUTOMATIC.
- Anfangswert für die Anzahl von Agenten im Pool (*num_initagents*): Wenn der Datenbankmanager gestartet wird, wird ein Pool von Verarbeitungsagenten nach diesem Wert erstellt. Dadurch wird die Leistung für Erstabfragen erhöht. Die Verarbeitungsagenten beginnen alle als Agenten im Bereitschaftsmodus.
- Maximale Anzahl von Verbindungen (*max_connections*): Gibt die maximale Anzahl von Verbindungen zum Datenbankmanagersystem an, die in jeder Datenbankpartition zulässig sind.
- Maximale Anzahl koordinierender Agenten (*max_coordagents*): In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität und aktiviertem **Verbindungskonzentrator** begrenzt dieser Wert die Anzahl koordinierender Agenten.

Konfigurationsparameter mit Einfluss auf die Abfrageoptimierung

Verschiedene Konfigurationsparameter wirken sich auf die Auswahl des Zugriffsplans durch den SQL- oder XQuery-Compiler aus. Viele von ihnen gelten für eine Umgebung mit Einzelpartitionsdatenbanken, während einige nur für eine Umgebung mit partitionierten Datenbanken gelten. In einer homogenen Umgebung mit partitionierten Datenbanken, in der identische Hardware eingesetzt wird, sollten die Werte, die für die einzelnen Parameter verwendet werden, für alle Datenbankpartitionen gleich sein.

Anmerkung: Wenn Sie einen Konfigurationsparameter dynamisch ändern, liest das Optimierungsprogramm die geänderten Parameterwerte möglicherweise nicht sofort, da vielleicht noch ältere Zugriffspläne im Paketcache vorhanden sind. Führen Sie zum Zurücksetzen des Paketcache den Befehl FLUSH PACKAGE CACHE aus.

Wenn in einem föderierten System die Mehrzahl der Abfragen auf Kurznamen zugreift, sollten Sie die Art der Abfragen, die Sie senden, auswerten, bevor Sie Ihre Umgebung ändern. Zum Beispiel speichert der Pufferpool in einer föderierten Datenbank keine Seiten aus Datenquellen im Cache zwischen. Datenquellen sind die Datenbankverwaltungssysteme (DBMSs) und Daten in einem föderierten System. Aus diesem Grund kann eine Erhöhung der Puffergröße nicht garantieren, dass das Optimierungsprogramm weitere Alternativen für Zugriffspläne in Betracht zieht, wenn es einen Zugriffsplan für Abfragen wählt, die Kurznamen enthalten. Allerdings stellt das Optimierungsprogramm möglicherweise fest, dass eine lokale Speicherung von Datenquellentabellen die Methode mit dem geringsten Aufwand oder ein erforderlicher Schritt für eine Sortieroperation ist. In diesem Fall kann eine Vergrößerung der verfügbaren Ressourcen die Leistung verbessern.

Die folgenden Konfigurationsparameter bzw. Faktoren wirken sich auf die Auswahl des Zugriffsplans durch den SQL- oder XQuery-Compiler aus:

- Die Größe der Pufferpools, die Sie angegeben haben, als Sie sie erstellt oder geändert haben

Bei der Auswahl des Zugriffsplans bezieht das Optimierungsprogramm den Ein-/Ausgabeaufwand für das Laden von Seiten von der Platte in den Pufferpool in die Kalkulation mit ein und schätzt die Anzahl der erforderlichen E/A-Operationen zur Erfüllung der Abfrage ab. Die Schätzung schließt eine Voraussage über die Nutzung des Pufferpools mit ein, da zum Lesen von Zeilen einer Seite, die sich bereits im Pufferpool befindet, keine weiteren physischen E/A-Operationen anfallen.

Das Optimierungsprogramm berücksichtigt den Wert der Spalte *npages* in den Systemkatalogtabellen SYSCAT.BUFFERPOOLS und, in Umgebungen mit partitionierten Datenbanken, in den Systemkatalogtabellen SYSCAT.BUFFERPOOLDBPARTITIONS.

Der Ein-/Ausgabeaufwand für das Lesen der Tabellen kann sich auf folgende Bereiche auswirken:

– Wie zwei Tabellen verknüpft werden.

– Ob ein Index ohne Clustering zum Lesen der Daten verwendet wird.

- Grad der Parallelität (dft_degree)

  Der Konfigurationsparameter `dft_degree` gibt die Parallelität durch Bereitstellen eines Standardwerts für das Sonderregister CURRENT DEGREE und die Bindeoption DEGREE an. Der Wert 1 bedeutet keine partitionsinterne Parallelität. Der Wert -1 bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität anhand der Anzahl von Prozessoren und der Art der Abfrage bestimmt.

  Anmerkung: Eine partitionsinterne Parallelverarbeitung findet nur statt, wenn Sie sie durch das Definieren des Konfigurationsparameters *intra_parallel* des Datenbankmanagers aktivieren.

- Standardabfrageoptimierungsklasse (dft_queryopt)

  Obwohl Sie beim Kompilieren von SQL- oder XQuery-Abfragen eine Abfrageoptimierungsklasse angeben können, können Sie außerdem eine Standardabfrageoptimierungsklasse definieren.

- Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)

  Mithilfe des Parameters *avg_appls* versucht das Optimierungsprogramm zu ermitteln, wie viel vom Pufferpool während der Ausführung für den ausgewählten Zugriffsplan verfügbar ist. Höhere Werte für diesen Parameter können das Optimierungsprogramm dahin gehend beeinflussen, dass es Zugriffspläne auswählt, die mit dem Pufferpool etwas sparsamer umgehen. Wenn Sie den Wert 1 angeben, geht das Optimierungsprogramm davon aus, dass der gesamte Pufferpool für die Anwendung verfügbar ist.

- Sortierspeichergröße (sortheap)

  Wenn die zu sortierenden Zeilen mehr als den im Sortierspeicher verfügbaren Speicherbereich in Anspruch nehmen, werden mehrere Sortierarbeitsgänge durchgeführt, wobei in jedem Arbeitsgang eine Untermenge der Gesamtmenge von Zeilen sortiert wird. Jeder Arbeitsgang des Sortiervorgangs wird in einer temporären Systemtabelle im Pufferpool gespeichert, die eventuell auf den Datenträger geschrieben wird. Wenn alle Arbeitsgänge des Sortiervorgangs abgeschlossen sind, werden die sortierten Untermengen zu einer einzigen sortierten Menge von Zeilen zusammengefügt. Eine Sortierung wird als „über eine Pipe geleitet (piped)" betrachtet, wenn sie keine temporäre Systemtabelle zur Speicherung der endgültigen, sortierten Liste von Daten erforderlich macht. Das heißt, dass die Ergebnisse der Sortierung in einem einzigen sequenziellen Zugriff gelesen werden können. Über eine Pipe geleitete Sortierungen führen zu einer besseren Leistung als nicht über eine Pipe geleitete und werden daher nach Möglichkeit verwendet.

Bei der Auswahl eines Zugriffsplans schätzt das Optimierungsprogramm den Aufwand der Sortieroperationen, einschließlich der Möglichkeiten, eine Sortierung über eine Pipe zu leiten, folgendermaßen ab:

  – Abschätzen der Menge der zu sortierenden Daten
  – Bestimmen mithilfe des Parameters *sortheap*, ob genügend Speicherbereich zur Verfügung steht, um die Sortierung über eine Pipe zu leiten.

- Maximaler Speicher für Sperrenliste (locklist) und maximale Anzahl Sperren pro Anwendung (maxlocks)

  Wenn die Isolationsstufe **RR** (Wiederholtes Lesen) verwendet wird, berücksichtigt das Optimierungsprogramm die Werte der Parameter *locklist* und *maxlocks*, um zu bestimmen, ob Sperren auf Zeilenebene möglicherweise durch Sperreneskalation in eine Sperre auf Tabellenebene umgewandelt werden. Wenn das Optimierungsprogramm eine Sperreneskalation für einen Tabellenzugriff voraussieht, wählt es für den Zugriffsplan eine Sperre auf Tabellenebene und vermeidet den Systemaufwand, der mit einer Sperreneskalation während der Ausführung der Abfrage verbunden wäre.

- CPU-Geschwindigkeit (cpuspeed)

  Der Parameter für die CPU-Geschwindigkeit wird vom Optimierungsprogramm zur Abschätzung des Aufwands für bestimmte Operationen verwendet. Die Schätzwerte zum CPU-Aufwand und zu verschiedenen E/A-Aufwänden helfen bei der Auswahl des besten Zugriffsplans für eine Abfrage.

  Die CPU-Geschwindigkeit eines Systems kann die Auswahl des Zugriffsplans wesentlich beeinflussen. Dieser Konfigurationsparameter wird bei der Installation oder Migration der Datenbank automatisch auf einen geeigneten Wert gesetzt. Sie sollten diesen Parameter nicht anpassen, es sei denn, Sie wollen eine Produktionsumgebung auf einem Testsystem modellieren oder die Auswirkungen einer Änderung der Hardware testen. Mithilfe dieses Parameters können Sie eine andere Hardwareumgebung modellieren, um die Zugriffspläne zu ermitteln, die für die andere Umgebung ausgewählt würden. Wenn der Datenbankmanager den Wert dieses automatischen Konfigurationsparameters neu berechnen soll, setzen Sie den Parameter auf den Wert -1.

- Anweisungszwischenspeicher (stmtheap)

  Die Größe des Anweisungszwischenspeichers hat zwar keinen Einfluss darauf, welchen Zugriffspfad das Optimierungsprogramm auswählt, kann sich jedoch auf den Grad der Optimierung auswirken, der für komplexe SQL- oder XQuery-Anweisungen ausgeführt wird.

  Wenn der Wert für den Parameter *stmtheap* nicht groß genug ist, empfangen Sie möglicherweise eine Warnung, die angibt, dass nicht genügend Speicher zur Verarbeitung der Anweisung zur Verfügung steht. Zum Beispiel kann der SQL-CODE +437 (SQLSTATE 01602) angeben, dass der Optimierungsgrad, der zur Kompilierung einer Anweisung verwendet wurde, geringer war als der Grad, den Sie angefordert haben.

- Kommunikationsbandbreite (comm_bandwidth)

  Die Kommunikationsbandbreite wird vom Optimierungsprogramm verwendet, um Zugriffspfade zu bestimmen. Das Optimierungsprogramm verwendet den Wert dieses Parameters, um den Aufwand zur Durchführung für bestimmte Operationen zwischen den Datenbankpartitionsservern in einer Umgebung mit partitionierten Datenbanken abzuschätzen.

- Zwischenspeichergröße für Anwendungen (applheapsz)

  Umfangreiche Schemata erfordern ausreichend Speicher im Anwendungszwischenspeicher.

Einschränkungen und Verhalten bei der Konfiguration von 'max_coordagents' und 'max_connections'

Der Standardwert für die Parameter *max_coordagents* und *max_connections* bei Version 9.5 ist AUTOMATIC, wobei *max_coordagents* auf 200 und *max_connections* auf -1 gesetzt ist (d. h., dieser Parameter wird auf den Wert von *max_coordagents* gesetzt). Durch diese Einstellungen wird der Konzentrator inaktiviert (OFF).

Bei der Onlinekonfiguration von *max_coordagents* oder *max_connections* sind einige Einschränkungen und Funktionsweisen zu beachten:

- Wenn der Wert von *max_coordagents* erhöht wird, wirkt sich die Einstellung unmittelbar aus und neue Anforderungen können neue koordinierende Agenten erstellen. Wenn der Wert vermindert wird, nimmt die Zahl der koordinierenden Agenten nicht sofort ab. Stattdessen nimmt die Zahl der koordinierenden Agenten nicht weiter zu und vorhandene koordinierende Agenten werden möglicherweise beendet, nachdem sie ihre aktuelle Arbeit erledigt haben, damit die Gesamtzahl der koordinierenden Agenten reduziert wird. Neue Arbeitsanforderungen, für die koordinierende Agenten erforderlich sind, werden erst berücksichtigt, wenn die Gesamtanzahl der koordinierenden Agenten den neuen Wert unterschreitet und ein neuer koordinierender Agenten freigegeben wird.
- Wenn der Wert von *max_connections* erhöht wird, wirkt sich die Einstellung unmittelbar aus und neue Verbindungen, die vorher aufgrund dieses Parameters blockiert wurden, werden zugelassen. Wenn der Wert vermindert wird, beendet der Datenbankmanager vorhandene Verbindungen nicht aktiv. Stattdessen sind neue Verbindungen erst möglich, wenn ausreichend vorhandene Verbindungen beendet wurden, damit der Wert unter das neue Maximum sinkt.
- Wenn *max_connections* auf -1 (Standardwert) gesetzt ist, entspricht die maximal zulässige Anzahl der Verbindungen dem Wert von *max_coordagents*. Wenn *max_coordagents* offline oder online aktualisiert wird, wird auch die maximal zulässige Anzahl der Verbindungen aktualisiert.

Wenn Sie den Wert von *max_coordagents* oder *max_connections* online ändern, können Sie ihn nicht dahingehend ändern, dass der Verbindungskonzentrator aktiviert wird, wenn er inaktiviert ist, bzw. dass der Verbindungskonzentrator inaktiviert wird, wenn er aktiviert ist. Beispiel: Wenn zur Zeit des DB2-Starts der Wert für *max_coordagents* kleiner als der für *max_connections* (Konzentrator ist aktiviert) ist, müssen alle Aktualisierungen, die online für diese beiden Parameter durchgeführt wurden, die Beziehung zwischen *max_coordagents* und *max_connections* aufrechterhalten. Ähnlich verhält es sich, wenn zur Zeit des DB2-Starts der Wert für *max_coordagents* größer-gleich dem Wert für *max_connections* (Konzentrator ist aktiviert) ist. In diesem Fall müssen alle online durchgeführten Aktualisierungen diese Beziehung aufrechterhalten.

Wenn Sie diese Art der Aktualisierung online durchführen, verzögert der Datenbankmanager die Aktualisierung anstatt sie zu unterlassen. Ähnlich wie in allen Fällen der Aktualisierung der Konfigurationsparameter für den Datenbankmanager, bei denen IMMEDIATE angegeben, aber nicht möglich ist, wird die Warnung SQL1362W zurückgegeben.

Wenn *max_coordagents* bzw. *max_connections* auf AUTOMATIC gesetzt ist, kann das folgende Verhalten erwartet werden:

- Beide Parameter können mit einem Anfangswert und der Einstellung AUTOMATIC konfiguriert werden. Beispiel: Mit dem folgenden Befehl wird dem Parameter *max_coordagents* der Wert 200 und die Einstellung AUTOMATIC zugeordnet:

  ```
  UPDATE DBM CONFIG USING max_coordagents 200 AUTOMATIC
  ```

  Diesen Parametern ist immer ein Wert zugeordnet. Dabei handelt es sich entweder um den Standardwert oder um einen von Ihnen angegebenen Wert. Ist nur AUTOMATIC bei der Aktualisierung eines der beiden Parameter angegeben, ist also kein Wert angegeben, und war dem Parameter zuvor ein Wert zugeordnet, bleibt dieser Wert bestehen. Dies betrifft nur die Einstellung AUTOMATIC.

  Anmerkung: Wenn der Konzentrator aktiviert ist, sind die diesen beiden Konfigurationsparametern zugeordneten Werte auch dann wichtig, wenn die Parameter auf AUTOMATIC gesetzt sind.
- Wenn beide Parameter auf AUTOMATIC gesetzt sind, ermöglicht der Datenbankmanager eine Zunahme der Anzahl der Verbindungen und koordinierenden Agenten je nach Auslastungsbedarf. Allerdings gilt dies nur unter Vorbehalt:
  1. Wenn der Konzentrator inaktiviert ist, behält der Datenbankmanager pro Verbindung nur *einen* einzigen koordinierenden Agenten.
  2. Wenn der Konzentrator aktiviert ist, versucht der Datenbankmanager, das durch die Werte für die Parameter festgelegte Verhältnis von koordinierenden Agenten und Verbindungen zu bewahren.

     Anmerkung:

     – Das Verfahren zur Wahrung des Verhältnisses wurde so konzipiert, dass es den Systembetrieb möglichst wenig beeinträchtigt. Es gewährleistet jedoch keine hundertprozentige Beibehaltung des Verhältnisses. In diesem Szenario sind immer neue Verbindungen möglich, auch wenn sie möglicherweise auf einen verfügbaren koordinierenden Agenten warten müssen. Neue koordinierende Agenten werden bei Bedarf zur Aufrechterhaltung des Verhältnisses erstellt. Werden Verbindungen beendet, kann der Datenbankmanager auch koordinierende Agenten zur Aufrechterhaltung des Verhältnisses beenden.

     – Der Datenbankmanager verkleinert das von Ihnen festgelegte Verhältnis nicht. Die von Ihnen festgelegten Anfangswerte für *max_coordagents* und *max_connections* werden als Untergrenze betrachtet.
- Die aktuellen und verzögerten Werte für beide Parameter können z. B. über den CLP oder mithilfe von APIs angezeigt werden. Bei den angezeigten Werten handelt es sich immer um die vom Benutzer festgelegten Werte. Beispiel: Wird der folgende Befehl abgesetzt und werden dann 30 gleichzeitige Verbindungen gestartet, die mit dieser Instanz arbeiten, lautet der angezeigte Wert für *max_connections* und *max_coordagents* weiterhin 20, AUTOMATIC:

  ```
  UPDATE DBM CFG USING max_connections 20 AUTOMATIC,
    max_coordagents 20 AUTOMATIC
  ```

  Wenn Sie die tatsächliche Anzahl der Verbindungen und koordinierenden Agenten ermitteln möchten, die momentan Monitorelemente ausführen, können Sie auch den Diagnosemonitor verwenden.
- Wenn *max_connections* auf AUTOMATIC mit einem größeren Wert als *max_coordagents* gesetzt ist (sodass der Konzentrator aktiviert ist) und *max_coordagents*

nicht auf AUTOMATIC gesetzt ist, ermöglicht der Datenbankmanager eine unbegrenzte Anzahl an Verbindungen, die nur eine begrenzte Anzahl an koordinierenden Agenten verwendet.

Anmerkung: Möglicherweise müssen Verbindungen auf verfügbare koordinierende Agenten warten.

Die Verwendung der Option AUTOMATIC für die Konfigurationsparameter *max_coordagents* und max_connections ist nur in den folgenden beiden Szenarios möglich:

1. Beide Parameter sind auf AUTOMATIC gesetzt.
2. Der Konzentrator wurde, im Gegensatz zu *max_coordagents*, mit *max_connections* auf AUTOMATIC gesetzt aktiviert.

Alle anderen Konfigurationen, bei denen AUTOMATIC für diese Parameter verwendet wird, werden blockiert und geben die Nachricht SQL6112N mit einem Ursachencode zurück, in dem die gültigen Einstellungen für AUTOMATIC für diese beiden Parameter erläutert werden.

Konfigurationsparameter des Datenbankmanagers

agent_stack_sz - Größe des Agentenstacks

Dieser Parameter steuert den virtuellen Speicherbereich, den DB2 jedem Agenten zuordnet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

Linux (32 Bit)
256 [16 – 1024]

Linux (64 Bit) und UNIX
1024 [256 – 32768]

Windows
16 [8 – 1000]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn ein Agent zur Arbeit für eine Anwendung initialisiert wird

Freigabe
Wenn ein Agent die Arbeit für eine Anwendung beendet

Sie können diesen Parameter zur Optimierung der Speicherauslastung des Servers für einen bestimmten Satz von Anwendungen verwenden. Komplexere Abfragen benötigen im Vergleich zu einfachen Abfragen mehr Stackspeicherbereich.

Mit diesem Parameter wird die ursprüngliche festgeschriebene Stackgröße für jeden Agenten in einer Windows-Umgebung festgelegt. Jeder Agentenstack kann standardmäßig bis auf die Größe des Standardreservestacks anwachsen, d. h. bis auf 256 KB (64 4-KB-Seiten). Dieser Grenzwert ist für die meisten Datenbankoperationen ausreichend. Unter UNIX und Linux wird der Wert des Konfigurationsparameters *agent_stack_sz* auf den nächsthöheren Potenzwert der Basis 2 aufgerundet. Die Standardeinstellung für UNIX sollte für die meisten Auslastungen ausreichend sein.

Wenn Sie jedoch eine umfangreiche SQL- oder XQuery-Anweisung vorbereiten, kann für den Agenten möglicherweise ein größerer Stackbereich erforderlich sein, und das System generiert eine Ausnahmebedingung aufgrund eines Stacküberlaufs (0xC00000FD). Wenn dies eintritt, wird der Server heruntergefahren, da der Fehler nicht behebbar ist.

Anmerkung: In Version 9.5 und späteren Versionen wird SQLCODE -973 anstelle einer Ausnahmebedingung aufgrund eines Stacküberlaufs zurückgegeben.

Der Agentenstack kann vergrößert werden, indem der Parameter *agent_stack_sz* auf einen höheren Wert als die Standardreservestackgröße von 64 Seiten gesetzt wird. Wenn der Wert des Parameters *agent_stack_sz* über der Standardreservestackgröße liegt, wird er vom Windows-Betriebssystem auf das nächste Vielfache von 1 MB gerundet. Wird die Agentenstackgröße beispielsweise auf 128 4-KB-Seiten erhöht, wird für jeden Agenten in Wirklichkeit ein Stack der Größe 1 MB reserviert. Wenn Sie für *agent_stack_sz* einen Wert festlegen, der unter der Standardreservestackgröße liegt, hat dies keine Auswirkungen auf die maximale Begrenzung, da der Agentenstack trotzdem bei Bedarf auf die Größe des Standardreservestacks anwächst. In diesem Fall ist der Wert für *agent_stack_sz* der ursprüngliche festgeschriebene Speicher für den Stack, wenn ein Agent erstellt wird.

Sie können die Standardreservestackgröße ändern, indem Sie mit dem Dienstprogramm 'db2hdr' die Headerdaten für die Datei db2syscs.exe ändern. Wenn Sie die Standardreservestackgröße ändern, wirkt sich dies auf alle Threads aus, während sich eine Änderung des Werts für *agent_stack_sz* nur auf die Stackgröße für Agenten auswirkt. Das Ändern der Standardstackgröße mithilfe des Dienstprogramms 'db2hdr' bietet den Vorteil einer besseren Granularität, wodurch die Stackgröße auf die minimal erforderliche Stackgröße festgelegt werden kann. Sie müssen jedoch DB2 stoppen und neu starten, damit die an db2syscs.exe vorgenommenen Änderungen wirksam werden.

Empfehlung: Wenn Sie beabsichtigen, mit umfangreichen oder komplexen XML-Daten in einer 32-Bit-Umgebung zu arbeiten, sollten Sie den Parameter *agent_stack_sz* auf mindestens 256 4-KB-Seiten aktualisieren. Bei sehr komplexen XML-Schemata kann es erforderlich sein, den Parameter *agent_stack_sz* wesentlich näher an den Grenzwert zu setzen, um Ausnahmebedingungen wegen Stacküberläufen beim Registrieren von Schemata oder Prüfen von XML-Dokumenten zu vermeiden.

Sie können die Stackgröße eventuell verringern, um anderen Clients mehr Adressraum zur Verfügung zu stellen. Dies ist möglich, wenn Ihre Umgebung folgende Kriterien erfüllt:

- Die Umgebung enthält nur einfache Anwendungen (z. B. einfache Online-Transaktionsprogramme, OLTP), in denen es keine komplexen Abfragen gibt.
- Für die Umgebung sind relativ viele gleichzeitig aktive Clients erforderlich (z. B. über 100).

Unter Windows stehen die Größe des Agentenstacks und die Anzahl gleichzeitig ausgeführter Clients in einem umgekehrten Verhältnis zueinander: Durch eine Vergrößerung des Stacks wird die mögliche Anzahl der Clients verringert, die gleichzeitig ausgeführt werden können. Dies liegt daran, dass der Adressraum auf Windows-Plattformen begrenzt ist.

agentpri - Agentenpriorität

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder Wert, der für diesen Konfigurationsparameter angegeben wird, funktioniert weiterhin genauso wie in früheren Versionen, und der Parameter wird weiterhin voll unterstützt. Wenn dieser Parameter für das Auslastungsmanagement (Workload-Management, WLM) verwendet wird, wird die Agentenpriorität der WLM-Serviceklasse ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Mit diesem Parameter wird die Priorität gesteuert, die sowohl allen Agenten als auch anderen Prozessen und Threads der Datenbankmanagerinstanz vom Scheduler des Betriebssystems zugewiesen wird. Durch diese Priorität wird festgelegt, wie den Datenbankmanagerprozessen, -agenten und -threads im Vergleich zu den anderen Prozessen und Threads, die auf dem System ausgeführt werden, CPU-Zeit zugewiesen wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

AIX -1 (System) [41 - 125]

Andere UNIX-Plattformen
-1 (System) [41 - 128]

Windows
-1 (System) [0 - 6]

Solaris
-1 (System) [0 - 59]

Wenn der Parameter auf den Wert -1 bzw. System gesetzt ist, wird keine besondere Aktion ausgeführt, und der Datenbankmanager erhält seine CPU-Zeit in der normalen Weise, in der das Betriebssystem allen Prozessen und Threads Prozessorzeit zuweist. Wenn der Parameter auf einen anderen Wert als -1 bzw. System gesetzt wird, erstellt der Datenbankmanager seine Prozesse und Threads mit einer statischen Priorität, die dem Wert des Parameters entspricht. Dadurch können Sie mit diesem Parameter die Priorität steuern, mit der die Prozesse und Threads des Datenbankmanagers (in einer Umgebung mit partitionierten Datenbanken gehören dazu auch koordinierende Agenten und Subagenten, parallele Systemsteuerprogramme und die FCM-Dämonen) auf Ihrem System ausgeführt werden.

Mit diesem Parameter kann der Durchsatz des Datenbankmanager erhöht werden. Die Werte für die Einstellung dieses Parameters sind von dem Betriebssystem abhängig, auf dem der Datenbankmanager ausgeführt wird. Beispielsweise ergeben in einer Linux- oder UNIX-Umgebung niedrige numerische Werte hohe Prioritäten. Wenn der Parameter auf einen Wert zwischen 41 und 125 gesetzt wird, erstellt der Datenbankmanager seine Agenten mit einer statischen UNIX-Priorität, die dem Wert dieses Parameters entspricht. Dies ist in Linux- oder UNIX-Umgebungen von Bedeutung, weil numerisch niedrige Werte hohe Prioritäten für den Datenbankmanager ergeben. Bei anderen Prozessen (einschließlich Anwendungen und Benutzern) können jedoch Verzögerungen auftreten, da sie nicht genügend CPU-Zeit erhalten. Sie sollten den Wert für diesen Parameter mit den anderen Aktivitäten, die Sie auf der Maschine erwarten, abstimmen.

Einschränkungen:

- Wenn Sie auf Linux- oder UNIX-Plattformen für diesen Parameter einen anderen als den Standardwert verwenden, können Sie den Governor nicht verwenden, um Agentenprioritäten zu ändern.
- Auf dem Solaris-Betriebssystem sollten Sie den Standardwert (-1) nicht ändern. Durch das Ändern des Standardwerts wird die Priorität des DB2-Prozesses auf Echtzeit gesetzt, wodurch alle verfügbaren Ressourcen in dem System monopolisiert werden können.

Empfehlung: Zu Anfang sollte der Standardwert verwendet werden. Dieser Wert stellt einen guten Kompromiss zwischen den Antwortzeiten für andere Benutzer bzw. Anwendungen und dem Durchsatz des Datenbankmanagers dar.

Wenn die Datenbankleistung von Bedeutung ist, können Sie durch Vergleichstests (Benchmark-Tests) die optimale Einstellung für diesen Parameter bestimmen. Eine Erhöhung der Priorität des Datenbankmanagers sollte nur mit großer Vorsicht vorgenommen werden, da die Leistung anderer Benutzerprozesse erheblich beeinträchtigt werden kann, besonders dann, wenn die CPU-Auslastung sehr hoch ist. Durch Erhöhen der Priorität der Datenbankmanagerprozesse und -threads können bedeutende Leistungssteigerungen erzielt werden.

aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene

Der Zwischenspeicher für die Anwendungsunterstützungsebene ist ein Kommunikationspuffer zwischen der lokalen Anwendung und dem zugeordneten Agenten. Dieser Puffer wird als gemeinsam benutzter Speicher von jedem Datenbankmanageragenten, der gestartet wird, zugeordnet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
15 [1 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn der Agentenprozess des Datenbankmanagers für die lokale Anwendung gestartet wird

Freigabe
Wenn der Agentenprozess des Datenbankmanagers beendet wird

Wenn die Anforderung an den Datenbankmanager oder die zugehörige Antwort nicht in den Puffer passt, wird sie in zwei oder mehr Sende-/Empfangspufferpaare aufgeteilt. Die Größe dieses Puffers sollte so festgelegt werden, dass die Mehrzahl der Anforderungen mit einem einzigen Sende-/Empfangspufferpaar verarbeitet werden kann. Die Größe der Anforderung hängt von der Speichermenge ab, die zur Speicherung folgender Daten erforderlich ist:

- Der Eingabe-SQL-Deskriptorbereich
- Alle zugeordneten Daten in den SQLVARs
- Der Ausgabe-SQL-Deskriptorbereich
- Andere Felder, die im Allgemeinen 250 Byte nicht überschreiten

Außer zur Steuerung dieses Kommunikationspuffers dient dieser Parameter auch den beiden folgenden Zwecken:

- Dieser Parameter wird verwendet, um die E/A-Blockgröße festzulegen, wenn ein Blockcursor geöffnet wird. Dieser Speicher für Blockcursor wird aus dem privaten Adressraum der Anwendung zugeordnet. Sie sollten daher die optimale Größe des privaten Speichers ermitteln, der jedem Anwendungsprogramm zugeordnet werden soll. Wenn der Data Server Runtime Client keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.
- Mit diesem Parameter wird die Kommunikationsgröße zwischen Agenten und db2fmp-Prozessen ermittelt. (Ein db2fmp-Prozess kann eine benutzerdefinierte Funktion oder eine abgeschirmte gespeicherte Prozedur sein.) Die Anzahl Byte wird jedem db2fmp-Prozess bzw. jedem im System aktiven Thread aus dem gemeinsam benutzten Speicher zugeordnet.

Die Daten, die von der lokalen Anwendung gesendet werden, werden vom Datenbankmanager in einem Bereich zusammenhängenden Speichers empfangen, der aus dem Abfragezwischenspeicher zugeordnet wird. Mit dem Parameter *aslheapsz* wird die Anfangsgröße des Abfragezwischenspeichers (für lokale und ferne Clients) festgelegt. Die maximale Größe des Abfragezwischenspeichers wird durch den Parameter *query_heap_sz* definiert.

Empfehlung: Wenn die Anforderungen Ihrer Anwendung im Allgemeinen klein sind und die Anwendung auf einem System mit eingeschränkter Speicherkapazität ausgeführt wird, ist es möglicherweise sinnvoll, den Wert dieses Parameters zu verringern. Wenn Ihre Abfragen im Allgemeinen sehr groß sind, mehr als eine Sende- und Empfangsanforderung erfordern und die Speicherkapazität Ihres Systems nicht eingeschränkt ist, kann es sinnvoll sein, den Wert dieses Parameters zu erhöhen.

Berechnen Sie anhand der folgenden Formel die Mindestanzahl der Seiten für *aslheapsz*:

```
aslheapsz >= ( sizeof(Eingabe-SQL-Deskriptorbereich)
                              + sizeof(jeder Eingabe-SQLVAR)
                              + sizeof(Ausgabe-SQL-Deskriptorbereich)
                              + 250 ) / 4096
```

Dabei ist sizeof(x) die Größe von x in Byte zur Berechnung der Seitenanzahl eines bestimmten Eingabe- oder Ausgabewerts.

Beachten Sie außerdem, welche Auswirkung dieser Parameter auf die Anzahl und die mögliche Größe von Blockcursorn hat. Große Zeilenblöcke können zu einer höheren Leistung führen, wenn die Anzahl oder die Größe der Zeilen, die übertragen werden, groß ist (z. B., wenn die Datenmenge größer als 4.096 Byte ist). Dies hat jedoch den Nachteil, dass größere Datensatzblöcke die Größe des für jede Verbindung benötigten Arbeitsspeichers erhöhen.

Größere Satzblöcke können außerdem mehr Abrufanforderungen verursachen, als tatsächlich für die Anwendung erforderlich wären. Sie können die Anzahl der Abrufanforderungen mit der Klausel OPTIMIZE FOR in der Anweisung SELECT in Ihrer Anwendung steuern.

audit_buf_sz - Prüfpuffergröße

Mit diesem Parameter wird die Größe des Puffers für die Prüfung der Datenbank angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0 - 65 000]

Maßeinheit
Seiten (4 KB)

Zuordnung
Beim Starten von DB2

Freigabe
Beim Stoppen von DB2

Der Standardwert für diesen Konfigurationsparameter ist Null (0). Wenn der Wert Null (0) ist, wird der Prüfpuffer nicht verwendet. Wenn der Wert größer als Null (0) ist, wird dem Prüfpuffer Speicherbereich zugeordnet, in den die von der Prüffunktion generierten Prüfsätze gestellt werden. Der Wert multipliziert mit 4-KB-Seiten ergibt die für den Prüfpuffer zugeordnete Speichermenge. Der Prüfpuffer kann nicht dynamisch zugeordnet werden; DB2 muss gestoppt und anschließend erneut gestartet werden, bevor der neue Wert für diesen Parameter in Kraft tritt.

Wenn Sie den Standardwert für diesen Parameter in einen Wert ändern, der größer als Null (0) ist, schreibt die Prüffunktion Datensätze asynchron im Vergleich zur Ausführung der Anweisungen, die die Prüfsätze generieren, auf Platte. Durch das Erhöhen des Standardparameterwerts Null (0) wird die Leistung von DB2 verbes-

sert. Der Wert Null (0) bedeutet, dass die Prüffunktion Datensätze synchron zur (d. h. zur gleichen Zeit wie die) Ausführung der Anweisungen, die die Prüfsätze generieren, auf Platte schreibt. Die synchrone Verarbeitung während der Prüfung verringert die Leistung von unter DB2 ausgeführten Anwendungen.

authentication - Authentifizierungstyp

Mit diesem Parameter wird festgelegt, wie und wo die Authentifizierung eines Benutzers stattfindet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DATA_ENCRYPT; DATA-_ENCRYPT_CMP; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

Wird für **authentication** der Wert SERVER angegeben, werden die Benutzer-ID und das Kennwort vom Client an den Server gesendet, sodass die Authentifizierung auf dem Server ausgeführt werden kann. Der Wert SERVER_ENCRYPT unterscheidet sich vom Wert SERVER nur darin, dass alle über das Netzwerk gesendeten Benutzer-IDs und Kennwörter verschlüsselt werden.

Der Wert DATA_ENCRYPT bedeutet, dass der Server verschlüsselte SERVER-Authentifizierungsschemata und die Verschlüsselung von Benutzerdaten akzeptiert. Die Authentifizierung funktioniert in exakt gleicher Weise wie bei SERVER_ENCRYPT.

Die folgenden Benutzerdaten werden bei Verwendung dieses Authentifizierungstyps verschlüsselt:

- SQL-Anweisungen.
- Daten von SQL-Programmvariablen.
- Ausgabedaten aus der Serververarbeitung einer SQL-Anweisung, einschließlich einer Beschreibung der Daten.
- Einige oder alle Antwortgruppendaten, die aus einer Abfrage resultieren.
- LOB-Streaming (LOB, Large Objects).
- SQLDA-Deskriptoren.

Der Wert DATA_ENCRYPT_CMP bedeutet, dass der Server verschlüsselte SERVER-Authentifizierungsschemata und die Verschlüsselung von Benutzerdaten akzeptiert. Darüber hinaus bietet dieser Authentifizierungstyp Kompatibilität mit Produkten früherer Versionen, die den Authentifizierungstyp DATA_ENCRYPT nicht unterstützen. Diese Produkte erhalten die Möglichkeit, die Verbindung mit dem Authentifizierungstyp SERVER_ENCRYPT und ohne Verschlüsselung von Benutzerdaten herzustellen. Produkte, die den neuen Authentifizierungstyp unterstützen, müssen

ihn verwenden. Dieser Authentifizierungstyp ist nur in der Konfigurationsdatei des Datenbankmanagers des Servers, jedoch nicht im Befehl CATALOG DATABASE gültig.

Anmerkung: Für eine Konfiguration zur Einhaltung von Standards (definiert im Abschnitt zur „Einhaltung von Standards") ist SERVER der einzige unterstützte Wert.

Der Wert CLIENT gibt an, dass alle Authentifizierungen auf dem Client stattfinden. Auf dem Server muss keine Authentifizierung mehr ausgeführt werden.

Der Wert KERBEROS bedeutet, dass die Authentifizierung auf einem Kerberos-Server mithilfe des Kerberos-Sicherheitsprotokolls für Authentifizierung ausgeführt wird. Bei Verwendung des Authentifizierungstyps KRB_SERVER_ENCRYPT auf dem Server und Unterstützung des Kerberos-Sicherheitssystems durch die Clients ist der tatsächliche Systemauthentifizierungstyp KERBEROS. Unterstützen die Clients das Kerberos-Sicherheitssystem nicht, ist der Systemauthentifizierungstyp praktisch äquivalent mit SERVER_ENCRYPT.

Der Wert GSSPLUGIN bedeutet, dass die Authentifizierung durch einen externen GSSAPI-basierten Sicherheitsmechanismus ausgeführt wird. Bei Verwendung des Authentifizierungstyps GSS_SERVER_ENCRYPT auf dem Server und Unterstützung des GSSPLUGIN-Sicherheitsmechanismus durch die Clients ist der tatsächliche Systemauthentifizierungstyp GSSPLUGIN (d. h., wenn die Clients eines der Plug-ins des Servers unterstützen). Unterstützen die Clients den GSSPLUGIN-Sicherheitsmechanismus nicht, ist der Systemauthentifizierungstyp praktisch äquivalent mit SERVER_ENCRYPT.

Empfehlung: In der Regel ist der Standardwert (SERVER) für lokale Clients geeignet. Wenn ferne Clients eine Verbindung zum Datenbankserver herstellen, ist SERVER_ENCRYPT der empfohlene Wert zum Schutz der Benutzer-ID und des Kennworts.

catalog_noauth - Katalogisieren ohne Berechtigung zulässig

Dieser Parameter gibt an, ob Benutzer Datenbanken und Knoten oder DCS- und ODBC-Verzeichnisse ohne SYSADM-Berechtigung katalogisieren und aus dem Katalog entfernen können.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

Datenbankserver mit lokalen und fernen Clients
NO [NO (0) — YES (1)]

Client; Datenbankserver mit lokalen Clients
YES [NO (0) — YES (1)]

Der Standardwert (0) für diesen Parameter gibt an, dass SYSADM-Berechtigung erforderlich ist. Wenn dieser Parameter auf 1 gesetzt ist, ist keine SYSADM-Berechtigung erforderlich.

clnt_krb_plugin - Client-Kerberos-Plug-in

Mit diesem Parameter wird der Name der Standard-Plug-in-Bibliothek für Kerberos angegeben, die für die clientseitige Authentifizierung sowie für die lokale Berechtigung zu verwenden ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null oder IBMkrb5 [beliebige gültige Zeichenfolge]

Standardmäßig lautet der Wert null auf Linux- und UNIX-Systemen sowie `IBMkrb5` auf Windows-Betriebssystemen. Das Plug-in wird verwendet, wenn der Client mit dem Authentifizierungstyp KERBEROS authentifiziert wird oder wenn die lokale Berechtigung durchgeführt und der Authentifizierungstyp in DBM CFG KERBEROS lautet.

clnt_pw_plugin - Client-Plug-in für Benutzer-ID und Kennwort

Mit diesem Parameter wird der Name der Benutzer-ID/Kennwort-Plug-in-Bibliothek angegeben, die für die clientseitige Authentifizierung sowie für die lokale Berechtigung zu verwenden ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Standardmäßig ist dieser Wert null und die von DB2 bereitgestellte Benutzer-ID/Kennwort-Plug-in-Bibliothek wird verwendet. Das Plug-in wird verwendet, wenn der Client mit dem Authentifizierungstyp CLIENT authentifiziert wird oder wenn die lokale Berechtigung durchgeführt und der Authentifizierungstyp in der

Datenbankmanagerkonfiguration (DBM CFG) CLIENT, SERVER, SERVER_ENCRYPT oder DATA_ENCRYPT lautet. Für Nichtrootinstallationen gilt, dass bei Verwendung der DB2-Bibliothek des Plug-ins für Benutzer-ID und Kennwort der Befehl db2rfe ausgeführt werden muss, bevor das DB2-Produkt verwendet wird.

cluster_mgr - Name des Cluster-Managers

Mithilfe dieses Parameters kann der Datenbankmanager inkrementelle Änderungen an der Clusterkonfiguration an den angegebenen Cluster-Manager übertragen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Informativ

Standardwert
Kein Standardwert

Gültige Werte

- TSA

Dieser Parameter wird bei der Konfiguration des Hochverfügbarkeitsclusters mit dem DB2-Dienstprogramm zur Instanzkonfiguration mit hoher Verfügbarkeit (db2haicu) festgelegt.

comm_bandwidth - Kommunikationsbandbreite

Dieser Parameter unterstützt das Abfrageoptimierungsprogramm bei der Ermittlung von Zugriffspfaden, indem er die Bandbreite zwischen Datenbankpartitionsservern angibt.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
-1 [.1 - 100 000]

Durch den Wert -1 wird der Parameter auf den Standardwert zurückgesetzt. Der Standardwert wird ausgehend von der Geschwindigkeit der zugrunde liegenden Kommunikation berechnet. Ein Wert von 100 kann für Systeme erwartet werden, die Gigabit Ethernet verwenden.

Maßeinheit
Megabyte pro Sekunde

Der Wert, der (in MB pro Sekunde) für die Übertragungsbandbreite berechnet wird, wird vom Abfrageoptimierungsprogramm zur Abschätzung des Aufwands

für bestimmte Operationen zwischen den Datenbankpartitionsservern eines partitionierten Datenbanksystems herangezogen. Das Optimierungsprogramm modelliert nicht die Kosten der Datenfernverarbeitung zwischen einem Client und einem Server. Dieser Parameter sollte daher nur die nominale Bandbreite zwischen den Datenbankpartitionsservern darstellen.

Sie können diesen Wert explizit festlegen, um ein Modell einer Produktionsumgebung auf Ihrem Testsystem zu erstellen oder die Auswirkungen einer Hardwareaufrüstung zu bewerten.

Empfehlung: Sie sollten diesen Parameter nur anpassen, wenn Sie ein Modell einer anderen Umgebung erstellen wollen.

Die Übertragungsbandbreite wird vom Optimierungsprogramm bei der Bestimmung der Zugriffspfade verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl REBIND PACKAGE).

conn_elapse - Antwortzeit für Verbindung

Dieser Parameter gibt an, innerhalb wie viel Sekunden eine TCP/IP-Verbindung zwischen zwei Datenbankpartitionsservern aufgebaut werden muss.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [0–100]

Maßeinheit
Sekunden

Wird der Verbindungsaufbau innerhalb der durch den Parameter angegebenen Zeit erfolgreich durchgeführt, kommt es zum Datenaustausch. Wird die Verbindung nicht rechtzeitig aufgebaut, wird ein weiterer Versuch zum Verbindungsaufbau durchgeführt. Kommt innerhalb der im Parameter *max_connretries* angegebenen Anzahl von Neuversuchen keine Verbindung zustande, wird eine Fehlernachricht ausgegeben.

cpuspeed - CPU-Geschwindigkeit

Dieser Parameter gibt die CPU-Geschwindigkeit der Maschine(n) wieder, auf der/denen die Datenbank installiert ist.

Wenn die DB2-Instanz erstellt wird, wird eine kleine UOW (Unit of Work) mehrere Male für das Betriebssystem ausgeführt. Die CPU-Geschwindigkeit ist die durchschnittliche Zeit, die zur Ausführung dieser UOW benötigt wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
-1 [$1x10^{-10}$ - 1] Der Wert -1 gibt an, dass der Wert dieses Parameters gemäß den Ergebnissen eines Messprogramms neu eingestellt wird.

Maßeinheit
Millisekunden

Dieses Programm wird ausgeführt, wenn keine Vergleichsergebnisse verfügbar sind, wenn die Daten für IBM RS/6000 Modell 530H in der Datei nicht gefunden werden oder wenn die Daten für Ihr System in der Datei nicht gefunden werden.

Sie können diesen Wert explizit festlegen, um ein Modell einer Produktionsumgebung auf Ihrem Testsystem zu erstellen oder die Auswirkungen einer Hardwareaufrüstung zu bewerten. Wenn der Wert auf -1 gesetzt wird, wird *cpuspeed* erneut berechnet.

Empfehlung: Sie sollten diesen Parameter nur anpassen, wenn Sie ein Modell einer anderen Umgebung erstellen wollen.

Der Wert für die CPU-Geschwindigkeit wird vom Optimierungsprogramm bei der Bestimmung von Zugriffspfaden verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl REBIND PACKAGE).

dft_account_str - Standardzeichenfolge für Abrechnung

Dieser Parameter fungiert als Standardsuffix für Abrechnungskennungen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Bei jeder Verbindungsanforderung durch eine Anwendung wird eine Abrechnungszeichenfolge vom Anwendungsrequester an einen DRDA-Anwendungsserver

gesendet, die aus einem von DB2 Connect erstellten Präfix und einem vom Benutzer eingegebenen Suffix besteht. Anhand dieser Abrechnungsdaten kann ein Systemadministrator die Ressourcennutzung für jeden Benutzerzugriff bestimmen.

Anmerkung: Dieser Parameter ist nur für DB2 Connect gültig.

Das Suffix wird vom Anwendungsprogramm, das die API `sqlesact()` aufruft, oder dem Benutzer, der die Umgebungsvariable DB2ACCOUNT definiert, übergeben. Wenn von der API oder der Umgebungsvariablen kein Suffix bereitgestellt wird, verwendet DB2 Connect den Wert dieses Parameters als Standardsuffix. Dieser Parameter ist insbesondere für Datenbankclients früherer Versionen (alle vor Version 2) nützlich, die nicht über Funktionen zum Senden einer Abrechnungszeichenfolge an DB2 Connect verfügen.

Empfehlung: Verwenden Sie in der Abrechnungszeichenfolge folgende Zeichen:
- Alphabetische Zeichen (A - Z)
- Numerische Zeichen (0 - 9)
- Unterstreichungszeichen (_)

dft_monswitches - Monitorschalter des Standarddatenbanksystems

Dieser Parameter ermöglicht Ihnen, eine Reihe von Schaltern zu definieren, die intern jeweils als ein Bit des Parameters dargestellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für
- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Anmerkung: Die Änderung tritt unmittelbar in Kraft, wenn Sie explizit eine Verbindung zur Instanz herstellen, bevor Sie die Schaltereinstellungen für 'dft_mon_xxxx' ändern. Ansonsten tritt die Einstellung beim nächsten Neustart der Instanz in Kraft.

Standardwert
Alle Schalter ausgeschaltet außer dem standardmäßig eingeschalteten *dft_mon_timestamp*

Der Parameter ist dahingehend eindeutig, dass Sie jeden dieser Schalter unabhängig aktualisieren können, indem Sie die folgenden Parameter festlegen:

dft_mon_uow
Standardwert des UOW-Schalters (Unit of Work - Arbeitseinheit) von Snapshot Monitor

dft_mon_stmt
Standardwert des Schalters für SQL-Anweisungen von Snapshot Monitor

dft_mon_table
Standardwert des Schalters für Tabellen von Snapshot Monitor

dft_mon_bufpool
Standardwert des Schalters für Pufferpool von Snapshot Monitor

dft_mon_lock
Standardwert des Schalters für Sperren von Snapshot Monitor

dft_mon_sort
Standardwert des Schalters für Sortiervorgänge von Snapshot Monitor

dft_mon_timestamp
Standardwert des Schalters für die Zeitmarke des Überwachungsprogramms für Momentaufnahme

Empfehlung: Jeder auf ON gesetzte Schalter (außer dft_mon_timestamp) weist den Datenbankmanager an, die zu diesem Schalter gehörenden Monitordaten zu sammeln. Das Erfassen zusätzlicher Monitordaten erhöht den Systemaufwand des Datenbankmanagers, wodurch die Systemleistung beeinträchtigt werden kann. Das Setzen des Schalters *dft_mon_timestamp* auf OFF wird wichtig, sobald sich die CPU-Auslastung 100% nähert. Wenn dies eintritt, wird die zur Ausgabe von Zeitmarken benötigte CPU-Zeit drastisch erhöht. Wenn der Zeitmarkenschalter inaktiviert ist, reduziert dies zudem erheblich den Gesamtaufwand für andere Daten in der Monitorschaltersteuerung.

Alle Überwachungsanwendungen erhalten diese Standardeinstellungen für die Schalter, wenn die Anwendung die erste Überwachungsanforderung absetzt (z. B. einen Schalter einstellt, den Ereignismonitor aktiviert, eine Momentaufnahme macht). In der Konfigurationsdatei sollten Sie einen Schalter nur dann aktivieren, wenn Sie Daten sammeln möchten, sobald der Datenbankmanager gestartet wird. (Andernfalls kann jede Überwachungsanwendung ihre eigenen Schalter einstellen, und die gesammelten Daten beziehen sich dann auf den Zeitpunkt, zu dem die Schalter eingestellt wurden.)

dftdbpath - Standarddatenbankpfad

Dieser Parameter enthält den Standarddateipfad, der zur Erstellung von Datenbanken unter dem Datenbankmanager verwendet wird. Wird beim Erstellen einer Datenbank kein Pfad angegeben, wird die Datenbank in dem Pfad erstellt, der durch den Parameter *dftdbpath* angegeben wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX Benutzerverzeichnis des Instanzeigners [beliebiger vorhandener Pfad]

Windows
Laufwerk, auf dem DB2 installiert ist [beliebiger vorhandener Pfad]

In einer Umgebung mit partitionierten Datenbanken müssen Sie sicherstellen, dass der Pfad, in dem die Datenbank erstellt wird, kein an das NFS angehängter Pfad (auf Linux- und UNIX-Plattformen) bzw. kein Netzwerklaufwerk ist (in Windows-Umgebungen). Der angegebene Pfad muss physisch auf jedem Datenbankpartitionsserver vorhanden sein. Um Verwirrung zu vermeiden, ist es am besten, einen Pfad anzugeben, der auf jedem Datenbankpartitionsserver lokal angehängt ist. Die Länge des Pfads darf maximal 205 Zeichen betragen. Das System hängt den Namen der Datenbankpartition am Ende des Pfads an.

Weil Datenbanken auf beträchtliche Größen anwachsen und möglicherweise viele Benutzer Datenbanken erstellen können (je nach Umgebung und Zielsetzung), ist es häufig sehr praktisch, alle Datenbanken an einer einzigen definierten Position erstellen und speichern zu lassen. Außerdem ist es von Vorteil, Datenbanken von anderen Anwendungen und Daten trennen zu können - sowohl aus Integritätsgründen als auch zur einfacheren Durchführung von Backup- und Recovery-Operationen.

In Linux- und UNIX-Umgebungen darf die Länge des im Parameter *dftdbpath* definierten Namens 215 Zeichen nicht überschreiten, und der Name muss ein gültiger, absoluter Pfadname sein. Unter Windows kann der im Parameter *dftdbpath* angegebene Name ein Laufwerkbuchstabe sein, auf den optional ein Doppelpunkt folgt.

Empfehlung: Wenn die Möglichkeit besteht, legen Sie umfangreiche Datenbanken auf einer anderen Platte an als andere Daten, auf die häufig zugegriffen wird, wie z. B. Dateien des Betriebssystems oder die Datenbankprotokolldateien.

diaglevel - Aufzeichnungsebene bei Fehlerdiagnose

Durch diesen Parameter wird der Typ der Diagnosefehler festgelegt, die in der Datei 'db2diag.log' aufgezeichnet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
3 [0 — 4]

Gültige Werte für diesen Parameter sind:

- **0** – Keine Aufzeichnung von Diagnosedaten
- **1** – Nur schwerwiegende Fehler
- **2** – Alle Fehler
- **3** – Alle Fehler und Warnungen
- **4** – Alle Fehler, Warnungen und Informationsnachrichten

Der Konfigurationsparameter *diagpath* wird zur Angabe des Verzeichnisses verwendet, in dem sich die Fehlerdatei, die Alertprotokolldatei und alle Speicherauszugsdateien befinden, die in Abhängigkeit vom Wert des Parameters *diaglevel* generiert werden.

Empfehlung: Sie können den Wert dieses Parameters erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können.

diagpath - Verzeichnispfad für Diagnosedaten

Mit diesem Parameter können Sie einen vollständig qualifizierten Pfad für DB2-Diagnosedaten angeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Null [beliebiger gültiger Pfadname]

Im angegebenen Verzeichnis können abhängig von Ihrer Plattform Speicherauszugsdateien, Trapdateien, eine Fehlerprotokolldatei, eine Benachrichtigungsdatei, eine Alertprotokolldatei und FODC-Pakete (FODC - First Occurrence Data Collection, Datenerfassung beim ersten Vorkommen) gespeichert werden.

Wenn dieser Parameter den Wert "null" hat, werden die Diagnoseinformationen in Dateien eines der folgenden Verzeichnisse (bzw. Ordner) geschrieben:

- In Windows-Umgebungen:
  - Benutzerdatendateien, zum Beispiel Dateien unter Instanzverzeichnissen, werden in folgende andere Verzeichnisse geschrieben, die sich von der Position unterscheiden, an der der Code installiert ist:
    - In Windows Vista-Umgebungen werden Datendateien in das Verzeichnis ProgramData\IBM\DB2\ geschrieben.
    - In Windows 2003- und XP-Umgebungen werden Benutzerdatendateien in das Verzeichnis Dokumente und Einstellungen\All Users\Application Data\IBM\DB2*Name_der_Kopie* geschrieben. Dabei ist *Name_der_Kopie* der Name Ihrer DB2-Kopie.
- In Linux- und UNIX-Umgebungen: Informationen werden in das Verzeichnis *INSTHOME*/sqllib/db2dump geschrieben. Dabei ist *INSTHOME* das Ausgangsverzeichnis der Instanz.

In Version 9.5 wird der Standardwert von **DB2INSTPROF** auf der globalen Ebene an der oben gezeigten neuen Position gespeichert. Andere Profilregistrierdaten-

bankvariablen, die die Position der Laufzeitdatendateien angeben, müssen den Wert von **DB2INSTPROF** abfragen. Bei den anderen Variablen handelt es sich um die folgenden:

- **DB2CLIINIPATH**
- **DIAGPATH**
- **SPM_LOG_PATH**

Empfehlung: Verwenden Sie die Standardeinstellung für den Konfigurationsparameter *diagpath*, oder verwenden Sie eine zentrale Speicherposition für den Wert von *diagpath* mehrerer Instanzen an.

In einer Umgebung mit partitionierten Datenbanken muss der Parameter *diagpath* lokalen Speicher auf dem Host verwenden, um die beste Leistung der Protokollierung zu erhalten. Dadurch wird ein separates Protokollierungs- und Diagnoseverzeichnis für die einzelnen physischen Partitionen erstellt. Sie können die Tabellenfunktion PD_GET_DIAG_HIST verwenden, um die Protokollsätze aus den verschiedenen Partitionen abzurufen; mit der Tabellenfunktion PD_GET-_LOG_MSGS können Sie das Benachrichtigungsprotokoll aus allen Partitionen abrufen.

dir_cache - Verzeichniscacheunterstützung

Dieser Parameter legt fest, ob die Datenbank-, die Knoten- und die DCS-Verzeichnisdateien in den Cache gestellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Yes [Yes; No]

Zuordnung

- Wenn eine Anwendung die erste Verbindungsanforderung absetzt, wird der Anwendungsverzeichniscache zugeordnet.
- Wenn eine Datenbankmanagerinstanz gestartet (`db2start`) wird, wird der Serververzeichniscache zugeordnet.

Freigabe

- Wenn ein Anwendungsprozess beendet wird, wird der Anwendungsverzeichniscache freigegeben.
- Wenn eine Datenbankmanagerinstanz gestoppt (`db2stop`) wird, wird der Serververzeichniscache freigegeben.

Durch die Verwendung des Verzeichniscache wird der Aufwand für die Verbindung verringert, indem die E/A-Operationen für Verzeichnisdateien vermieden und die Suchoperationen in Verzeichnissen zum Abruf von Verzeichnisdaten minimiert werden. Es gibt zwei Arten von Verzeichniscaches:

- Ein Anwendungsverzeichniscache, der für jeden Anwendungsprozess auf dem System zugeordnet und verwendet wird, auf dem die Anwendung ausgeführt wird.
- Ein Serververzeichniscache, der für einige der internen Datenbankmanagerprozesse zugeordnet und verwendet wird.

Wenn eine Anwendung die erste Verbindungsanforderung absetzt, werden in einem Anwendungsverzeichniscache alle Verzeichnisdateien gelesen, und die Informationen werden im privaten Speicher dieser Anwendung zwischengespeichert. Der Cache wird vom Anwendungsprozess auch für nachfolgende Verbindungsanforderungen verwendet und während der Dauer des Anwendungsprozesses beibehalten. Wenn eine Datenbank nicht im Anwendungsverzeichniscache gefunden wird, werden die Verzeichnisdateien nach den Informationen durchsucht, aber der Cache wird nicht aktualisiert. Wenn die Anwendung einen Verzeichniseintrag ändert, wird durch die nächste Verbindungsanforderung innerhalb dieser Anwendung eine Aktualisierung des Cache für diese Anwendung bewirkt. Der Anwendungsverzeichniscache für andere Anwendungen wird nicht aktualisiert. Wenn der Anwendungsprozess beendet ist, wird der Cache freigegeben. (Zur Aktualisierung des Verzeichniscache, der von einer Sitzung des Befehlszeilenprozessors verwendet wird, geben Sie den Befehl `db2 terminate` ein.)

Wenn eine Datenbankmanagerinstanz gestartet wird (`db2start`), werden in einem Serververzeichniscache alle Verzeichnisdateien gelesen, und die Informationen werden im Servercache zwischengespeichert. Dieser Cache wird beibehalten, bis die Instanz gestoppt (`db2stop`) wird. Wenn ein Verzeichniseintrag in diesem Cache nicht gefunden wird, werden die Verzeichnisdateien nach den Daten durchsucht. Dieser Serververzeichniscache wird während der Ausführung der Instanz zu keinem Zeitpunkt aktualisiert.

Empfehlung: Verwenden Sie einen Verzeichniscache, wenn Ihre Verzeichnisdateien nicht häufig geändert werden und die Leistung von entscheidender Bedeutung ist.

Auf fernen Clients kann darüber hinaus der Verzeichniscache von Vorteil sein, wenn Ihre Anwendungen mehrere verschiedene Verbindungsanforderungen absetzen. In diesem Fall verringert das Zwischenspeichern die Häufigkeit, mit der eine einzige Anwendung die Verzeichnisdateien lesen muss.

Der Verzeichniscache kann auch die Leistung bei der Erstellung von Momentaufnahmen des Datenbanksystemmonitors erhöhen. Außerdem sollten Sie beim Aufruf der Momentaufnahme den Datenbanknamen explizit angeben und nicht den Aliasnamen für die Datenbank verwenden.

Anmerkung: Bei der Erstellung von Momentaufnahmen können Fehler auftreten, wenn der Verzeichniscache aktiviert wurde und Datenbanken katalogisiert, entkatalogisiert, erstellt oder gelöscht werden, nachdem der Datenbankmanager gestartet wurde.

discover - Discovery-Modus

Mit diesem Parameter wird ermittelt, welche Art von Erkennungsanforderungen der Client (falls überhaupt) vornehmen kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients

- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
SEARCH [DISABLE, KNOWN, SEARCH]

Aus der Perspektive eines Clients gilt einer der folgenden Fälle:

- Wenn SEARCH für *discover* angegeben ist, kann der Client Discovery-Anforderungen mit dem Parameter SEARCH zum Suchen nach DB2-Serversystemen im Netzwerk absetzen. Discovery mit dem Parameter SEARCH stellt eine Implikation der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität zur Verfügung. Wenn für *discover* SEARCH angegeben ist, können vom Client sowohl Discovery-Anforderungen mit dem Parameter SEARCH als auch Discovery-Anforderungen mit dem Parameter KNOWN abgesetzt werden.
- Wenn für *discover* KNOWN angegeben ist, können vom Client nur Discovery-Anforderungen mit dem Parameter KNOWN abgesetzt werden. Durch Angabe einiger Verbindungsinformationen für den Verwaltungsserver auf einem bestimmten System werden alle Instanz- und Datenbankinformationen auf dem DB2-System an den Client zurückgegeben.
- Wenn für *discover* DISABLE angegeben ist, ist Discovery auf dem Client inaktiviert.

Der Standard-Discovery-Modus ist SEARCH.

discover_inst - Discovery-Serverinstanz

Dieser Parameter gibt an, ob diese Instanz von DB2 Discovery aufgespürt werden kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
ENABLE [ENABLE, DISABLE]

Der Standardwert des Parameters, "enable", gibt an, dass die Instanz aufgespürt werden kann, der Wert "disable" dagegen verhindert, dass die Instanz aufgespürt wird.

fcm_num_buffers - Anzahl FCM-Puffer

Dieser Parameter gibt die Anzahl von 4-KB-Puffern an, die sowohl zwischen den als auch innerhalb der Datenbankserver für interne Kommunikation (Nachrichten) verwendet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
Automatic [128 - 65 300]

64-Bit-Plattformen
Automatic [128 - 524 288]

- Datenbankserver mit lokalen und fernen Clients: Standardwert ist 1024.
- Datenbankserver mit lokalen Clients: Standardwert ist 512.
- Partitionierter Datenbankserver mit lokalen und fernen Clients: Standardwert ist 4096.

In Datenbanksystemen mit einer Partition wird dieser Parameter nur dann verwendet, wenn die partitionsinterne Parallelität durch Ändern des Parameters *intra_parallel* vom Standardwert NO auf den Wert YES aktiviert wurde.

Es ist möglich, sowohl einen Anfangswert als auch das Attribut AUTOMATIC anzugeben.

Wenn dieser Parameter auf den Wert AUTOMATIC gesetzt ist, überwacht FCM die Ressourcennutzung und kann Ressourcen entweder erhöhen oder verringern, wenn sie innerhalb von 30 Minuten nicht genutzt werden. Der Betrag, um den Ressourcen erhöht oder verringert werden können, hängt von der Plattform ab. Insbesondere können Ressourcen unter Linux nur um 25 % über den Anfangswert erhöht werden. Wenn der Datenbankmanager beim Starten einer Instanz die angegebene Anzahl von Ressourcen nicht zuordnen kann, verringert er die Konfigurationswerte inkrementell, bis er die Instanz starten kann.

Wenn Sie über mehrere logische Knoten auf derselben Maschine verfügen, kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen. Außerdem kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen, wenn wegen der Anzahl von Benutzern im System, wegen der Anzahl von Datenbankpartitionsservern im System oder wegen der Komplexität der Anwendungen die verfügbaren Nachrichtenpuffer nicht ausreichen.

Wenn Sie mehrere logische Knoten verwenden, wird ein Pool mit der im Parameter *fcm_num_buffers* angegebenen Anzahl von Puffern von allen logischen Knoten auf derselben Maschine gemeinsam benutzt. Die Größe des Pools wird durch die Mul-

tiplikation des Werts von *fcm_num_buffers* mit der Anzahl der logischen Knoten auf dieser physischen Maschine ermittelt. Überprüfen Sie daher den verwendeten Wert erneut. Berücksichtigen Sie, wie viele FCM-Puffer auf der Maschine (oder den Maschinen) mit mehreren logischen Knoten insgesamt zugeordnet werden.

fcm_num_channels - Anzahl FCM-Kanäle

Mit diesem Parameter wird die Anzahl der FCM-Kanäle für jede Datenbankpartition angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients
- Satellitendatenbankserver mit lokalen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX-Plattformen (32-Bit)
Automatic - mit den Anfangswerten 256, 512, 2 048 [128 - 120 000]

UNIX-Plattformen (64-Bit)
Automatic - mit den Anfangswerten 256, 512, 2 048 [128 - 524 288]

Windows (32-Bit)
Automatic - mit dem Anfangswert 10 000 [128 - 120 000]

Windows (64-Bit)
Automatic - mit den Anfangswerten 256, 512, 2 048 [128 - 524 288]

- Für einen Datenbankserver mit lokalen und fernen Clients ist der Anfangswert 512.
- Für einen Datenbankserver mit lokalen Clients ist der Anfangswert 256.
- Für Server in Umgebungen mit partitionierten Datenbanken mit lokalen und fernen Clients ist der Anfangswert 2 048.

In nicht partitionierten Datenbankumgebungen muss der Parameter *intra_parallel* aktiviert werden, bevor der Parameter *fcm_num_channels* verwendet werden kann.

Ein FCM-Kanal stellt einen logischen Kommunikationsendpunkt zwischen EDUs (Engine Dispatchable Units) dar, die in der DB2-Steuerkomponente ausgeführt werden. Sowohl Steuerungsflüsse (Anforderung und Antwort) als auch Datenflüsse (Daten aus Tabellenwarteschlangen) nutzen Kanäle, um Daten zwischen Partitionen zu übertragen.

Wenn dieser Parameter auf den Wert AUTOMATIC gesetzt ist, überwacht FCM die Kanalnutzung und ordnet Ressourcen je nach Bedarfsänderung inkrementell zu bzw. gibt sie inkrementell frei.

fed_noauth - Authentifizierung bei Servern mit föderierten Datenbanken umgehen

Dieser Parameter legt fest, ob die Authentifizierung für einen Server mit föderierten Datenbanken in der Instanz umgangen wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
No [Yes; No]

Wenn der Parameter *fed_noauth* auf *yes*, der Parameter *authentication* auf *server* oder auf *server_encrypt* und der Parameter *federated* auf *yes* gesetzt ist, wird die Authentifizierung in der Instanz umgangen. In diesem Fall wird angenommen, dass die Authentifizierung an der Datenquelle erfolgt. Gehen Sie vorsichtig vor, wenn *fed_noauth* auf *yes* gesetzt ist. Es wird weder auf dem Client noch in DB2 eine Authentifizierung durchgeführt. Jeder Benutzer, der den SYSADM-Authentifizierungsnamen kennt, kann SYSADM-Berechtigung für den Server mit föderierten Datenbanken erlangen.

federated - Unterstützung für Systeme föderierter Datenbanken

Dieser Parameter aktiviert bzw. inaktiviert die Unterstützung für Anwendungen, die verteilte Anforderungen für von Datenquellen (wie z. B. der DB2-Produktfamilie und Oracle) verwaltete Daten übergeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes; No]

federated_async - Maximale ATQs pro Abfrage (Konfigurationsparameter)

Dieser Parameter legt die maximale Anzahl von ATQs (Asynchrony Table Queues - Asynchrony-Tabellenwarteschlangen) im Zugriffsplan fest, die vom Server mit föderierten Datenbanken unterstützt wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients, wenn die Föderation aktiviert ist.

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
0 [0 - 32 767 einschließlich, -1, ANY]

Wenn ANY oder -1 angegeben ist, ermittelt das Optimierungsprogramm die Anzahl der ATQs für den Zugriffsplan. Das Optimierungsprogramm ordnet allen auswählbaren SHIP-Operatoren oder fernen Pushdown-Operatoren im Plan eine ATQ zu. Der für die Serveroption DB2_MAX_ASYNC_REQUESTS_PER_QUERY angegebene Wert schränkt die Anzahl der asynchronen Anforderungen ein.

Empfehlung
Der Konfigurationsparameter *federated_async* stellt den Standard- oder Anfangswert für das Sonderregister und die Bindeoption bereit. Sie können den Wert für diesen Parameter überschreiben, indem Sie den Wert des Sonderregisters CURRENT FEDERATED ASYNCHRONY, der Bindeoption FEDERATED_ASYNCHRONY oder der Vorkompilierungsoption FEDERATED_ASYNCHRONY erhöhen oder verringern.

Wenn das Sonderregister oder die Bindeoption den Konfigurationsparameter *federated_async* nicht überschreibt, legt der Wert des Parameters die maximale Anzahl von ATQs im Zugriffsplan fest, die der Server mit föderierten Datenbanken zulässt. Wenn das Sonderregister oder die Bindeoption diesen Parameter überschreibt, legt der Wert des Sonderregisters oder der Bindeoption die maximale Anzahl der ATQs im Plan fest.

Änderungen am Konfigurationsparameter *federated_async* wirken sich auf dynamische Anweisungen aus, sobald die aktuelle Arbeitseinheit festgeschrieben wird. Nachfolgende dynamische Anweisungen können den neuen Wert automatisch identifizieren. Ein Neustart der föderierten Datenbank ist nicht erforderlich. Pakete mit eingebettetem SQL werden weder inaktiviert noch implizit erneut gebunden, wenn der Wert des Konfigurationsparameters *federated_async* geändert wird.

Wenn Sie möchten, dass sich der neue Wert des Konfigurationsparameters *federated_async* auch auf statische SQL-Anweisungen auswirkt, müssen Sie das Paket erneut binden.

fenced_pool - Maximale Anzahl abgeschirmter Prozesse

Für db2fmp-Threadprozesse (Prozesse, die threadsichere gespeicherte Prozeduren und benutzerdefinierte Funktionen bedienen) stellt dieser Parameter die Anzahl Threads dar, die in jedem db2fmp-Prozess in den Cache gestellt werden. Für db2fmp-Prozesse, die keine Threadprozesse sind, stellt dieser Parameter die Anzahl der zwischengespeicherten Prozesse dar.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
-1 (*max_coordagents*), Automatic [-1; 0–64 000]

Maßeinheit
Zähler

Einschränkungen:

- Wenn dieser Parameter dynamisch aktualisiert und der Wert verringert wird, beendet der Datenbankmanager db2fmp-Threads oder -Prozesse nicht proaktiv, sondern stellt sie nicht mehr in den Cache, da sie zur Reduzierung der Anzahl von in den Cache gestellten db2fmp-Prozessen auf den neuen Wert verwendet werden.
- Wenn dieser Parameter dynamisch aktualisiert und der Wert erhöht wird, stellt der Datenbankmanager mehr db2fmp-Threads und -Prozesse in den Cache, wenn diese erstellt werden.
- Wenn dieser Parameter auf -1 gesetzt ist (Standardwert), wird der Wert des Konfigurationsparameters *max_coordagents* angenommen. Beachten Sie, dass nur der Wert von *max_coordagents* angenommen wird, nicht die automatische Einstellung oder das automatische Verhalten.
- Wenn dieser Parameter auf AUTOMATIC gesetzt ist, gilt Folgendes (dies ist auch der Standardwert):
  - Der Datenbankmanager erlaubt eine Zunahme der Anzahl der in den Cache gestellten db2fmp-Threads und -Prozesse auf der Basis der oberen Grenze der koordinierenden Agenten. Besonders das automatische Verhalten dieses Parameters erlaubt ihm, in Abhängigkeit von der maximalen Anzahl von koordinierenden Agenten, die der Datenbankmanager je seit dem Start (gleichzeitig) registriert hat, an Größe zuzunehmen.
  - Der diesem Parameter zugeordnete Wert stellt einen unteren Grenzwert für die Anzahl der in den Cache zu stellenden db2fmp-Threads und -Prozesse dar.

Empfehlung: Wenn Ihre Umgebung abgeschirmte gespeicherte Prozeduren oder benutzerdefinierte Funktionen verwendet, kann über diesen Parameter sichergestellt werden, dass zur Verarbeitung der maximalen Anzahl gleichzeitig auf der Instanz ablaufender gespeicherter Prozeduren und benutzerdefinierter Funktionen eine ausreichende Anzahl db2fmp-Prozesse verfügbar ist. Dadurch wird sichergestellt, dass beim Ausführen von gespeicherten Prozeduren und benutzerdefinierten Funktionen keine neuen Prozesse im abgeschirmten Modus erstellt werden müssen.

Wenn sich herausstellt, dass der Standardwert für Ihre Umgebung nicht geeignet ist, weil eine unangemessen große Menge Systemressourcen für db2fmp-Prozesse verwendet wird und es deshalb zu Leistungseinbußen des Datenbankmanager kommt, kann anhand der folgenden Informationen eine erste Optimierung für diesen Parameters vorgenommen werden:

```
fenced_pool = Anzahl Anwendungen, die gleichzeitig Aufrufe gespeicherter
Prozeduren und UDF-Aufrufe absetzen dürfen
```

Wenn der Parameter *keepfenced* auf den Wert YES gesetzt ist, bleibt jeder im Cache-Pool erstellte db2fmp-Prozess bestehen und verbraucht auch dann noch Systemressourcen, wenn der Aufruf der Routine im abgeschirmten Modus verarbeitet und an den Agenten zurückgegeben wurde.

Wenn *keepfenced* auf NO gesetzt ist, werden db2fmp-Prozesse, die keine Threadprozesse sind, nach Beendigung der Ausführung beendet, und es gibt keinen Cache-Pool. Multithread-db2fmp-Prozesse sind weiter vorhanden, es werden in diesen Prozessen jedoch keine Threads in den Pool gestellt. Dies bedeutet, dass Sie auf Ihrem System einen db2fmp-C-Threadprozess und einen db2fmp-Java-Threadprozess haben können, auch wenn *keepfenced* auf NO gesetzt ist.

In früheren Versionen hatte dieser Parameter den Namen *maxdari*.

group_plugin - Gruppen-Plug-in

Mit diesem Parameter wird der Name der Gruppen-Plug-in-Bibliothek angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Standardmäßig ist dieser Wert null, und DB2 verwendet die Gruppensuchfunktion (Lookup) des Betriebssystems. Das Plug-in wird für alle Gruppensuchen verwendet. Für Nichtrootinstallationen gilt, dass bei Verwendung der DB2-Bibliothek des Plug-ins für Benutzer-ID und Kennwort der Befehl db2rfe ausgeführt werden muss, bevor das DB2-Produkt verwendet wird.

health_mon - Überwachung mit dem Diagnosemonitor

Mit diesem Parameter können Sie angeben, ob Sie anhand verschiedener Diagnoseanzeiger eine Instanz, seine zugeordneten Datenbanken und Datenbankobjekte überwachen wollen.

Konfigurationstyp
Datenbankmanager

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
On [On; Off]

Zugehörige Parameter

Wenn *health_mon* aktiviert ist (Standardeinstellung), erfasst ein Agent Informationen zum ordnungsgemäßen Betrieb der ausgewählten Objekte. Wenn auf der Basis der von Ihnen festgelegten Schwellenwerte der Betrieb eines Objekts als nicht ordnungsgemäß eingestuft wird, können automatisch Benachrichtigungen versendet und Aktionen unternommen werden. Wenn *health_mon* inaktiviert ist, wird der ordnungsgemäße Betrieb von Objekten nicht überwacht.

Sie können mit der Diagnosezentrale oder dem Befehlszeilenprozessor die Instanz und die Datenbankobjekte auswählen, die Sie überwachen wollen. Basierend auf den vom Diagnosemonitor erfassten Daten können Sie auch angeben, wann Benachrichtigungen gesendet und welche Aktionen unternommen werden sollen.

indexrec - Zeitpunkt für Indexneuerstellung

Dieser Parameter gibt an, wann der Datenbankmanager versucht, ungültige Indizes neu zu erstellen und ob eine Indexerstellung während einer aktualisierenden DB2-Recovery oder einer Wiedergabe des HADR-Protokolls in der Bereitschaftsdatenbank wiederholt wird.

Konfigurationstyp
Datenbank und Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX Datenbankmanager
restart [restart; restart_no_redo; access; access_no_redo]

Windows Datenbankmanager
restart [restart; restart_no_redo; access; access_no_redo]

Datenbank
Systemeinstellung verwenden [system; restart; restart_no_redo; access; access_no_redo]

Für diesen Parameter sind fünf Einstellungen möglich:

SYSTEM
Verwenden Sie die Systemeinstellung, die in der Konfigurationsdatei des Datenbankmanagers angegeben wurde, um festzulegen, wann ungültige Indizes neu erstellt werden und ob Protokollsätze zur Indexerstellung während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt werden müssen. (Anmerkung: Diese Einstellung ist nur für Datenbankkonfigurationen gültig.)

ACCESS
Ungültige Indizes werden neu erstellt, wenn zum ersten Mal auf den Index zugegriffen wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird

und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt.

ACCESS_NO_REDO
Ungültige Indizes werden erneut erstellt, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird. Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt; diese Indizes bleiben ungültig. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt.

RESTART
Die Standardeinstellung für *indexrec*. Ungültige Indizes werden neu erstellt, wenn der Befehl RESTART DATABASE explizit oder implizit abgesetzt wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme neu erstellt.

Beachten Sie, dass der Befehl RESTART DATABASE implizit abgesetzt wird, wenn der Parameter *autorestart* aktiviert ist.

RESTART_NO_REDO
Ungültige Indizes werden neu erstellt, wenn der Befehl RESTART DATABASE explizit oder implizit abgesetzt wird. (Der Befehl RESTART DATABASE wird implizit abgesetzt, wenn der Parameter *autorestart* aktiviert ist-.)Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt; stattdessen werden diese Indizes neu erstellt, wenn die aktualisierende Recovery beendet wird oder wenn die HADR-Übernahme stattfindet.

Indizes können ungültig werden, wenn nicht behebbare Plattenfehler auftreten. Wenn dabei die Daten selbst beschädigt werden, können die Daten verloren gehen. Wird jedoch ein Index beschädigt, kann der Index durch Neuerstellung wiederhergestellt werden. Wird ein Index neu erstellt, während Benutzer mit der Datenbank verbunden sind, können zwei Probleme auftreten:

- Während der Erstellung der Indexdatei könnte es zu einer unerwarteten Verschlechterung der Antwortzeit kommen. Benutzer, die auf die Tabelle zugreifen und diesen bestimmten Index verwenden, müssen auf die Neuerstellung des Index warten.
- Nach der Indexneuerstellung könnten unerwartete, aktive Sperren beibehalten werden, besonders dann, wenn die Benutzertransaktion, die die Indexneuerstellung ausgelöst hat, keine Commit- oder Rollbackoperation ausgeführt hat.

Empfehlung: Die beste Option für diesen Parameter auf einem Server mit vielen Benutzern, wenn die Zeit für den Neustart keine kritische Rolle spielt, ist die Indexneuerstellung beim Neustart der Datenbank (DATABASE RESTART) als Teil des Vorgangs, mit dem die Datenbank nach einem Systemabsturz wiederhergestellt und online verfügbar gemacht wird.

Wenn dieser Parameter auf „ACCESS" oder auf „ACCESS_NO_REDO" gesetzt wird, verschlechtert sich während der Indexneuerstellung die Leistung des

Datenbankmanagers. Jeder Benutzer, der auf den Index oder die Tabelle zugreift, der bzw. die gerade neu erstellt wird, muss zunächst auf die Beendigung der Indexneuerstellung warten.

Wenn dieser Parameter auf „RESTART" gesetzt wird, dauert der Neustart der Datenbank wegen der Indexneuerstellung länger, jedoch wird die normale Verarbeitung nicht mehr beeinträchtigt, sobald die Datenbank wieder online verfügbar ist.

Anmerkung: Bei der Datenbankrecovery werden alle ausführbaren Dateien von SQL-Prozeduren im Dateisystem entfernt, die zu der gerade wiederhergestellten Datenbank gehören. Wenn der Parameter *indexrec* auf RESTART gesetzt wird, werden alle ausführbaren Dateien von SQL-Prozeduren aus dem Datenbankkatalog extrahiert und in das Dateisystem zurückgestellt, wenn die nächste Verbindung zur Datenbank hergestellt wird. Wenn der Parameter *indexrec* nicht auf RESTART gesetzt ist, wird die ausführbare Datei einer SQL-Prozedur erst in das Dateisystem extrahiert, wenn diese SQL-Prozedur zum ersten Mal ausgeführt wird.

Der Unterschied zwischen den Werten RESTART und RESTART_NO_REDO bzw. zwischen den Werten ACCESS und ACCESS_NO_REDO ist nur von Bedeutung, wenn die vollständige Protokollierung für Indexerstellungsoperationen, wie z. B. die Operationen CREATE INDEX und REORG INDEX, oder für eine Indexneuerstellung aktiviert ist. Sie können die Protokollierung aktivieren, indem Sie den Datenbankkonfigurationsparameter *logindexbuild* oder LOG INDEX BUILD beim Ändern einer Tabelle aktivieren. Wenn Sie *indexrec* entweder auf RESTART oder auf ACCESS setzen, kann für Operationen, die eine protokollierte Indexerstellung einbeziehen, eine aktualisierende Recovery durchgeführt werden, ohne dass das Indexobjekt einen ungültigen Status aufweist; dies hätte zur Folge, dass der Index später neu erstellt werden müsste.

instance_memory - Instanzspeicher

Mit diesem Parameter wird die maximale Speichermenge angegeben, die der Datenbankpartition zugeordnet werden kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]

32-Bit-Plattformen
Automatic [0 - 1 000 000]

64-Bit-Plattformen
Automatic [0 - 68 719 476 736]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Instanz gestartet wird

Freigabe
Wenn die Instanz gestoppt wird

Der Standardwert für **instance_memory** ist AUTOMATIC, d. h. der tatsächliche Wert wird zum Zeitpunkt der Aktivierung der Datenbankpartition (db2start) berechnet. Der tatsächlich verwendete Wert liegt zwischen 75 und 95 Prozent des physischen Arbeitsspeichers des Systems, geteilt durch die Anzahl konfigurierter lokaler Datenbankpartitionen der Instanz. Der Wert sollte für dedizierte Datenbankserversysteme geeignet sein.

Anmerkung:

- Wenn der für **instance_memory** angegebene Wert größer ist als der physische Hauptspeicher im System, schlägt db2start mit folgendem SQL-Fehler fehl: SQL1220N Die gemeinsam genutzte Speichergruppe des Datenbankmanagers kann nicht zugeordnet werden.
- Wenn **instance_memory** dynamisch auf einen Wert aktualisiert wird, der geringer als der verfügbare physische Arbeitsspeicher ist, wird die Anforderung verarbeitet und eine neue Obergrenze festgelegt. Dynamische Verringerungen auf den Wert von **instance_memory** sind nur zulässig, wenn die neue Einstellung größer als der derzeit verwendete Wert von **instance_memory** ist, andernfalls wird die Anforderung bis zum nächsten Aufruf von db2start verzögert.
- Wenn **instance_memory** dynamisch auf einen Wert aktualisiert wird, der den physischen Arbeitsspeicher übersteigt, während die Instanz aktiv ist, wird die Anforderung verzögert, und der nächste Aufruf von db2start schlägt mit dem folgenden SQL-Fehler fehl: SQL1220N Die gemeinsam genutzte Speichergruppe des Datenbankmanagers kann nicht zugeordnet werden.

Wenn der gemeinsam genutzte FCM-Speicher (FCM - Fast Communication Manager) zugeordnet wird, wird der Anteil der einzelnen lokalen Datenbankpartitionen an der gesamten Größe des gemeinsam genutzten FCM-Speichers für das System im Grenzwert **instance_memory** der Datenbankpartition berücksichtigt.

Wenn für einen bestimmten Zwischenspeicher mehr Speicherkapazität angefordert wird und der Speichergrenzwert der Datenbankpartition (**instance_memory**) bereits erreicht wurde, versucht DB2 zunächst, in allen gemeinsam genutzten und privaten Zwischenspeichern die Speicherbelegung um die angeforderte Speicherkapazität zu reduzieren. Falls immer noch nicht ausreichend freier Instanzspeicher (Parameter **instance_memory**) vorhanden ist, schlägt die Anforderung fehl, und die Anwendung, die die Anforderung initialisiert hat, empfängt den betreffenden SQLCODE-Wert, der beschreibt, bei welchem Zwischenspeicher ein Fehler wegen zu geringen Speichers aufgetreten ist.

Eine Ausnahme zu diesem Verhalten wird gemacht, wenn Speicheranforderungen bestehen, die für den Betrieb von DB2 kritisch sind (d. h. wenn der Anforderung nicht entsprochen wird, würde die Datenbank als ungültig markiert oder die Instanz beendet werden). Beachten Sie, dass im Falle kritischer Anforderungen zunächst versucht wird, die aktuelle Speicherbelegung durch die Datenbankpartition zu verringern. Wenn der freie Instanzspeicher (**instance_memory**) immer noch nicht ausreicht, fordert DB2 die Speicherkapazität vom Betriebssystem an. Lässt das Betriebssystem die Speicheranforderung zu, überschreitet der aktuelle Wert von **instance_memory** den konfigurierten Grenzwert, jedoch schlagen alle anderen nicht kritischen Speicheranforderungen so lange fehl, bis wieder ausreichender freier Speicher zur Verfügung steht.

Anmerkung: Einschränkungen bei DPF-Instanzen: Obwohl der Parameter **instance_memory** die Speicherkapazität festlegt, die eine einzelne DB2-Datenbankpartition zuordnen kann, handelt es sich um einen Konfigurationsparameter auf Instanzebene, sodass alle Datenbankpartitionen in der Instanz dieselbe Einstellung für den Parameter **instance_memory** besitzen. Wenn der Parameter **instance_memory** jedoch auf AUTOMATIC gesetzt wird, wird die tatsächliche Obergrenze auf jeder separaten Maschine einzeln auf der Basis der Kapazität des Arbeitsspeichers (RAM) und der Anzahl der definierten lokalen Partitionen berechnet. Daher ist es möglich, dass für verschiedene Partitionen verschiedene Speichergrenzwerte gelten.

DB2-Speicherbelegung steuern:

Wenn der Parameter **instance_memory** auf AUTOMATIC gesetzt wurde, wird beim Starten der Instanz (db2start) eine feste Obergrenze für die gesamte Speicherbelegung durch die Instanz festgelegt. Die tatsächliche Speicherbelegung durch DB2 variiert je nach Auslastung. Wenn STMM für die Optimierung des Datenbankspeichers (Parameter **database_memory**) aktiviert wurde, was für neue Datenbanken der Standardwert ist, aktualisiert STMM während der Laufzeit dynamisch die Größe der für die Leistung kritischen Zwischenspeicher innerhalb der gemeinsam genutzten Speicher der Datenbank. Dies geschieht in Abhängigkeit von dem freien physischen Hauptspeicher im System, während gleichzeitig sichergestellt wird, dass ausreichend freier Instanzspeicher (Parameter **instance_memory**) für funktionalen Speicherbedarf zur Verfügung steht.

Je nach Workload passt sich die Standardspeicherkonfiguration von DB2 an den Speicherbedarf der Instanz an, ohne dass explizite automatische Leistungsoptimierung des gesamten Instanzspeichers erforderlich wäre. Beispiele:

- Bei stark ausgelasteten Instanzen erhöht STMM die Größe leistungskritischer Zwischenspeicher nach Bedarf. Es wird mehr funktionaler Speicher belegt, da mehr Datenbankagenten für Anwendungen aktiv sind und dafür mehr funktionalen Speicher beanspruchen. Wenn ausreichender freier Instanzspeicher (**instance_memory**), aber nur sehr wenig freier physischer Hauptspeicher im System vorhanden ist, beginnt STMM, die Größe leistungskritischer Zwischenspeicher zu verringern, wobei sichergestellt wird, dass das System nicht das Paging startet. Wenn der funktionale Speicherbedarf sinkt, steht wieder mehr freier physischer Hauptspeicher im System zur Verfügung, und STMM beginnt, die Größe der leistungskritischen Zwischenspeicher wieder zu erhöhen.
- Bei weniger stark ausgelasteten Instanzen wird von der Instanz weniger funktionaler Speicher belegt, und falls nicht ausreichender freier physischer Hauptspeicher im System vorhanden ist, verkleinert STMM die leistungskritischen Zwischenspeicher.

Wenn der Parameter **instance_memory** auf einen bestimmten Wert gesetzt wurde und bei mindestens einer aktiven Datenbank, für die STMM aktiviert ist, der Wert AUTOMATIC für den Parameter **database_memory** festgelegt wurde, erhöht STMM den Wert für **database_memory**, sodass DB2 fast die gesamte im Parameter **instance_memory** festgelegte Speicherkapazität verwenden kann, wobei sichergestellt wird, dass ausreichend freier Instanzspeicher (**instance_memory**) für funktionale Speicheranforderungen verfügbar ist. In diesem Szenario überwacht STMM nicht den freien physischen Hauptspeicher im System, daher muss der Parameter **instance_memory** angemessen konfiguriert werden, um sicherzustellen, dass kein Paging stattfindet.

Verwenden Sie die neue benutzerdefinierte Funktion (User-Defined Function) admin_get_dbp_mem_usage, um die gesamte Speicherbelegung durch eine DB2-Instanz für eine bestimmte Datenbankpartition oder für alle Datenbankpartitionen abzurufen. Diese UDF gibt auch den aktuellen oberen Grenzwert zurück.

Einschränkungen in einigen Linux-Kernels:
Aufgrund von Betriebssystemeinschränkungen in einigen Linux-Kernels lässt STMM die Einstellung AUTOMATIC für den Parameter **database_memory** nicht zu, es sei denn, für **instance_memory** wird ein bestimmter Wert definiert (nicht AUTOMATIC). Wenn **database_memory** auf AUTOMATIC gesetzt wurde und **instance_memory** später zurück auf AUTOMATIC gesetzt wird, wird der Konfigurationsparameter **database_memory** bei der nächsten Aktivierung der Datenbank mit der Einstellung COMPUTED aktualisiert. Wenn einige Datenbanken bereits aktiv sind, stoppt STMM die Optimierung von deren gesamten Speicherkapazitäten (Parameter **database_memory**). Diese Einschränkung besteht für Red Hat Enterprise Linux (RHEL) 5-Plattformen und SUSE Linux Enterprise Server 10 SP1-Plattformen (oder höhere Versionen) nicht mehr.

intra_parallel - Partitionsinterne Parallelität aktivieren

Dieser Parameter gibt an, ob der Datenbankmanager partitionsinterne Parallelität verwenden kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NO (0) [SYSTEM (-1), NO (0), YES (1)]

Der Wert -1 bewirkt, dass der Parameter auf „YES" oder „NO" gesetzt wird, abhängig von der Hardware, auf welcher der Datenbankmanager ausgeführt wird.

Ist dieser Parameter auf "YES" gesetzt, können die durch die Parallelverarbeitung erreichten Leistungssteigerungen unter anderem bei Datenbankabfragen und der Indexerstellung genutzt werden.

Anmerkung:

- Bei der parallelen Indexerstellung wird dieser Konfigurationsparameter nicht verwendet.
- Wenn Sie diesen Parameterwert ändern, werden Pakete möglicherweise erneut an die Datenbank gebunden und es kann zu Leistungseinbußen kommen.

java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter

Dieser Parameter legt die maximale Größe des Zwischenspeichers fest, der von dem zur Verarbeitung von gespeicherten Java-DB2-Prozeduren und benutzerdefinierten Funktionen gestarteten Java-Interpreter verwendet wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

HP-UX
4096 [0 - 524 288]

Alle anderen Betriebssysteme
2048 [0 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn eine gespeicherte Java-Prozedur oder eine benutzerdefinierte Funktion gestartet wird

Freigabe
Wenn der db2fmp-Prozess (abgeschirmt) oder der db2agent-Prozess (gesichert) beendet wird

Es gibt einen Zwischenspeicher für jeden DB2-Prozess (einen für jeden Agenten oder Subagenten auf Linux- und UNIX-Plattformen und einen für jede Instanz auf anderen Plattformen). Es gibt einen Zwischenspeicher für jede abgeschirmte benutzerdefinierte Funktion und für jede abgeschirmte gespeicherte Prozedur. Es gibt einen Zwischenspeicher für jeden Agenten (ausschließlich der Subagenten) für gesicherte Routinen. Es gibt einen Zwischenspeicher für jeden db2fmp-Prozess, der eine gespeicherte Java-Prozedur ausführt. Für Multithread-db2fmp-Prozesse werden mehrere Anwendungen, die threadsichere abgeschirmte Routinen verwenden, von einem einzigen Zwischenspeicher bedient. Nur die Agenten oder Prozesse, die in Java geschriebene benutzerdefinierte Funktionen oder gespeicherte Prozeduren ausführen, ordnen diesen Speicher zu. Bei partitionierten Datenbanksystemen wird in jeder Datenbankpartition derselbe Wert verwendet.

XML-Daten werden umgesetzt, wenn sie als Parameter IN, OUT oder INOUT an gespeicherte Prozeduren übergeben werden. Wenn Sie mit gespeicherten Java-Prozeduren arbeiten, müssen eventuell die Zwischenspeichergröße je nach Anzahl und Größe der XML-Argumente sowie die Anzahl externer gespeicherter Prozeduren, die gleichzeitig ausgeführt werden, erhöht werden.

jdk_path - Installationspfad für Software Developer's Kit für Java

Dieser Parameter gibt das Verzeichnis an, in dem das Software Developer's Kit (SDK) für Java installiert ist, das verwendet wird, um gespeicherte Java-Prozeduren und benutzerdefinierte Funktionen auszuführen. CLASSPATH und andere vom Java-Interpreter verwendete Umgebungsvariablen werden aus dem Wert dieses Parameters berechnet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [Gültiger Pfad]

Wenn das SDK für Java zusammen mit Ihrem DB2-Produkt installiert wurde, ist dieser Parameter richtig definiert. Wenn Sie allerdings den Datenbankmanagerparameter (dbm cfg) zurücksetzen, müssen Sie angeben, wo das SDK für Java installiert ist.

keepfenced - Abgeschirmten Prozess beibehalten

Dieser Parameter gibt an, ob ein Prozess im abgeschirmten Modus beibehalten wird, nachdem der Aufruf einer Routine im abgeschirmten Modus beendet ist. Prozesse im abgeschirmten Modus werden als getrennte Systementitäten erstellt, um vom Benutzer geschriebenen Code im abgeschirmten Modus vom Agentenprozess des Datenbankmanagers zu isolieren. Dieser Parameter gilt nur für Datenbankserver.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Yes [Yes; No]

Wenn *keepfenced* auf *no* gesetzt ist und die ausgeführte Routine nicht threadsicher ist, wird für jeden Aufruf im abgeschirmten Modus ein neuer Prozess im abgeschirmten Modus erstellt und zerstört. Wenn *keepfenced* auf *no* gesetzt ist und die ausgeführte Routine threadsicher ist, wird der Prozess im abgeschirmten Modus beibehalten, der für den Aufruf erstellte Thread wird jedoch beendet. Wenn *keepfenced* auf *yes* gesetzt ist, wird ein Thread oder Prozess im abgeschirmten Modus

für nachfolgende Aufrufe im abgeschirmten Modus wiederverwendet. Wird der Datenbankmanager gestoppt, werden alle noch anstehenden Threads und Prozesse im abgeschirmten Modus beendet.

Wenn dieser Parameter auf *yes* gesetzt wird, werden vom Datenbankmanager zusätzliche Systemressourcen für jeden Prozess im abgeschirmten Modus in Anspruch genommen, der aktiviert wird, solange die durch den Parameter *fenced_pool* definierte Anzahl noch nicht erreicht ist. Ein neuer Prozess wird nur dann erstellt, wenn zur Verarbeitung eines nachfolgenden Aufrufs einer Routine im abgeschirmten Modus kein Prozess im abgeschirmten Modus verfügbar ist. Dieser Parameter wird ignoriert, wenn *fenced_pool* auf 0 gesetzt ist.

Empfehlung: In einer Umgebung, in der die Anzahl der Anforderungen im abgeschirmten Modus im Vergleich zu den übrigen Anforderungen groß ist und Systemressourcen nicht begrenzt sind, kann dieser Parameter auf *yes* gesetzt werden. Dadurch wird die Leistung des Prozesses im abgeschirmten Modus erhöht, weil der zusätzliche Systemaufwand zur Ersterstellung eines Prozesses im abgeschirmten Modus vermieden wird, da ein bereits vorhandener Prozess im abgeschirmten Modus zur Verarbeitung des Aufrufs verwendet wird. Dies erspart vor allem bei Java-Routinen den Aufwand, die JVM (Java Virtual Machine) zu starten, was eine erhebliche Leistungssteigerung bedeutet.

Zum Beispiel könnte bei einer OLTP-Bankanwendung (OLTP - Online-Transaktionsprogramm) für Soll- und Haben-Transaktionen der Code, mit dem jede Transaktion ausgeführt wird, in einer gespeicherten Prozedur ausgeführt werden, die in einem Prozess im abgeschirmten Modus ausgeführt wird. In dieser Anwendung wird die Hauptauslastung aus Prozessen im abgeschirmten Modus heraus ausgeführt. Wenn dieser Parameter auf *no* gesetzt wird, entsteht für jede Transaktion der Systemaufwand zum Erstellen eines neuen Prozesses im abgeschirmten Modus, wodurch die Leistung erheblich beeinträchtigt wird. Wenn dieser Parameter jedoch auf *yes* gesetzt wird, versucht jede Transaktion, einen vorhandenen Prozess im abgeschirmten Modus zu verwenden, wodurch der zusätzliche Systemaufwand vermieden wird.

In früheren Versionen von DB2 hatte dieser Parameter den Namen *keepdari*.

local_gssplugin - GSS-API-Plug-in für lokale Berechtigung auf Instanzebene

Mit diesem Parameter wird der Name der GSS-API-Plug-in-Standardbibliothek angegeben, die zur lokalen Berechtigung auf Instanzebene zu verwenden ist, wenn der Konfigurationsparameter *authentication* des Datenbankmanagers auf den Wert GSSPLUGIN oder GSS_SERVER_ENCRYPT gesetzt ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

max_connections - Maximale Anzahl von Clientverbindungen

Dieser Parameter gibt die maximal zulässige Anzahl Clientverbindungen pro Datenbankpartition an.

Konfigurationstyp
Datenbankmanager

Parametertyp
Online konfigurierbar

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Datenbankserver oder Connect-Server mit lokalen und fernen Clients (für *max_connections*, *max_coordagents*, *num_initagents*, *num_poolagents* und auch für *federated_async*, wenn Sie eine Umgebung mit föderierten Datenbanken verwenden)

Standardwert [Bereich]
-1 und AUTOMATIC (*max_coordagents*) [-1 und AUTOMATIC; 1 – 64 000]

Die Einstellung -1 bedeutet, dass der dem Parameter *max_coordagents* zugeordnete Wert und nicht die automatische Einstellung bzw. das automatische Verhalten verwendet wird. Die Einstellung AUTOMATIC bedeutet, dass der Datenbankmanager den Wert für diesen Parameter mit der am besten geeigneten Technik auswählt. Die Einstellung AUTOMATIC ist ein Aktivierungs- bzw. Inaktivierungsschalter in der Konfigurationsdatei und vom entsprechenden Wert unabhängig. Daher können beide Einstellungen, -1 und AUTOMATIC, die Standardeinstellung sein.

Detaillierte Informationen finden Sie in „Einschränkungen und Verhalten bei der Konfiguration von 'max_coordagents' und 'max_connections'" auf Seite 527.

Der Konzentrator

Der Konzentrator ist inaktiviert, wenn *max_connections* kleiner oder gleich *max_coordagents* ist. Der Konzentrator ist aktiviert, wenn *max_connections* größer als *max_coordagents* ist.

Dieser Parameter steuert die maximale Anzahl der Clientanwendungen, die mit einer Datenbankpartition der Instanz verbunden sein können. In der Regel wird jede Anwendung einem Koordinatoragenten zugeordnet. Der Agent vereinfacht die Operationen zwischen der Anwendung und der Datenbank. Die Konzentratorfunktion wird nicht aktiviert, wenn der Standardwert für diesen Parameter verwendet wird. Daher arbeitet jeder Agent in seinem eigenen privaten Speicher und verwendet Ressourcen des Datenbankmanagers und globale Datenbankressourcen wie z. B. den Pufferpool gemeinsam mit anderen Agenten. Die Konzentratorfunktion wird hingegen aktiviert, wenn der Parameter auf einen Wert gesetzt wird, der den Standardwert überschreitet.

max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten

Dieser Parameter gibt an, wie häufig höchstens versucht wird, eine TCP/IP-Verbindung zwischen zwei Datenbankpartitionsservern aufzubauen.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
5 [0–100]

Wenn der Verbindungsaufbau zwischen zwei Datenbankpartitionsservern fehlschlägt (z. B. bei Erreichen des im Parameter *conn_elapse* angegebenen Werts), gibt *max_connretries* an, wie viele Wiederholungen durchgeführt werden dürfen, um die Verbindung zu einem Datenbankpartitionsserver herzustellen. Bei Überschreitung des für diesen Parameter angegebenen Werts wird eine Fehlernachricht zurückgegeben.

max_coordagents - Maximale Anzahl koordinierender Agenten

Dieser Parameter wird verwendet, um die Anzahl koordinierender Agenten zu begrenzen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
200, Automatic [-1; 0–64 000]

Die Einstellung -1 wird in den Wert 200 umgesetzt.

Detaillierte Informationen finden Sie in „Einschränkungen und Verhalten bei der Konfiguration von 'max_coordagents' und 'max_connections'" auf Seite 527.

Der Konzentrator

Wenn der Konzentrator nicht aktiviert ist, d. h., wenn *max_connections* kleinergleich *max_coordagents* ist, legt dieser Parameter die maximale Anzahl koordinierender Agenten fest, die gleichzeitig auf einem Serverknoten vorhanden sein können.

Für jede lokale oder ferne Anwendung, die eine Verbindung zu einer Datenbank oder einer Instanz herstellt, wird ein koordinierender Agent aufgerufen. Anforderungen, für die eine Instanzverbindung erforderlich ist, sind z. B. CREATE DATABASE, DROP DATABASE und Befehle des Datenbanksystemmonitors.

Wenn der Konzentrator aktiviert ist, d. h., wenn *max_connections* größer als *max_coordagents* ist, kann es mehr Verbindungen als Koordinatoragenten für die Verbindungen geben. Eine Anwendung ist nur dann aktiv, wenn sie von einem Koordinatoragenten bedient wird. Andernfalls ist die Anwendung inaktiv. Anforderungen von einer aktiven Anwendung werden vom Koordinatoragenten der Datenbank (und in SMP- oder MPP-Konfigurationen von Subagenten) bedient. Anforderungen von einer inaktiven Anwendung werden in eine Warteschlange gestellt, bis ein Datenbankkoordinatoragent zur Bedienung der Anwendung zugeordnet wird, wenn diese aktiv wird. Daher kann dieser Parameter zur Steuerung der Systembelastung verwendet werden.

max_querydegree - max_querydegree - Maximaler Grad der Parallelität bei Abfragen

Dieser Parameter gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf dieser Instanz des Datenbankmanagers ausgeführt werden. Eine SQL-Anweisung verwendet bei ihrer Ausführung innerhalb einer Datenbankpartition nicht mehr als diese Anzahl paralleler Operationen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
-1 (ANY) [ANY, 1 - 32 767] (ANY bedeutet 'vom System festgelegt')

Der Konfigurationsparameter *intra_parallel* muss aktiviert („YES") sein, damit die Datenbankpartition die partitionsinterne Parallelität für SQL-Anweisungen verwenden kann. Der Parameter *intra_parallel* ist für die parallele Indexerstellung nicht mehr erforderlich.

Der Standardwert für diesen Konfigurationsparameter lautet -1. Dieser Wert bedeutet, dass das System den vom Optimierungsprogramm festgelegten Parallelitätsgrad verwendet. Andernfalls wird der vom Benutzer angegebene Wert verwendet.

Anmerkung: Der Grad der Parallelität für eine SQL-Anweisung kann bei der Kompilierung der Anweisung mithilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption DEGREE angegeben werden.

Der maximale Grad der Parallelität für eine aktive Anwendung bei Abfragen kann mit dem Befehl SET RUNTIME DEGREE geändert werden. Der zur Laufzeit tatsächlich verwendete Parallelitätsgrad ist der niedrigste der folgenden Werte:

- Konfigurationsparameter *max_querydegree*
- Parallelitätsgrad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

Dieser Konfigurationsparameter gilt nur für Abfragen.

max_time_diff - Maximale Zeitdifferenz zwischen Knoten

Dieser Parameter gibt die maximale Zeitdifferenz (in Minuten) an, die zwischen den in der Knotenkonfigurationsdatei aufgelisteten Datenbankpartitionsservern zulässig ist.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
60 [1 - 1 440]

Maßeinheit
Minuten

Jeder Datenbankpartitionsserver verfügt über eine eigene Systemuhr. Sind einer Transaktion zwei oder mehr Datenbankpartitionsserver zugeordnet, deren Systemzeiten sich um mehr als diese Zeitdifferenz unterscheiden, wird die Transaktion zurückgewiesen und ein SQLCODE-Wert zurückgegeben. (Die Transaktion wird nur zurückgewiesen, wenn eine Datenänderung mit ihr verbunden ist.)

DB2 verwendet die *Weltzeit* (UTC - Coordinated Universal Time), damit unterschiedliche Zeitzonen beim Festlegen dieses Parameters nicht ins Gewicht fallen. Die Weltzeit ist identisch mit der Westeuropäischen Zeit (Greenwich Mean Time).

maxagents - Maximale Anzahl von Agenten

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Mit diesem Parameter wird die maximale Anzahl von Datenbankmanageragenten (Koordinatoragenten und Subagenten) angegeben, die zu einem gegebenen Zeitpunkt zur Verfügung stehen, um Anwendungsanforderungen zu empfangen.

Konfigurationstyp
Datenbankmanager

Gilt für
- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
200 [1 - 64 000]

400 [1 - 64 000] Auf partitioniertem Datenbankserver mit lokalen und fernen Clients

Maßeinheit
Zähler

Wenn Sie die Anzahl koordinierender Agenten begrenzen möchten, verwenden Sie den Parameter *max_coordagents*.

Dieser Parameter kann in Umgebungen mit eingeschränkten Speicherkapazitäten nützlich sein, um den Gesamtspeicherbedarf des Datenbankmanagers zu begrenzen, da jeder zusätzliche Agent zusätzlichen Speicher benötigt.

Empfehlung: Der Wert des Parameters *maxagents* sollte mindestens der Summe der Werte für *maxappls* aller Datenbanken, auf die gleichzeitig zugegriffen werden kann, entsprechen. Wenn die Anzahl der Datenbanken größer als der Wert des Parameters *numdb* ist, dann besteht die sicherste Methode zur Angabe dieses Werts darin, das Produkt von *numdb* und dem größten Wert für *maxappls* zu bilden.

Für die Initialisierung und Verwaltung jedes weiteren Agenten ist zusätzlicher Speicher erforderlich, der beim Starten des Datenbankmanagers zugeordnet wird.

Falls bei dem Versuch, eine Verbindung zu einer Datenbank herzustellen, Speicherfehler vorkommen, führen Sie die folgenden Konfigurationsanpassungen aus:

- Erhöhen Sie in einer Umgebung mit nicht partitionierten Datenbanken, in der abfrageinterne Parallelitäten nicht aktiviert sind, den Wert des Datenbankkonfigurationsparameters *maxagents*.
- Erhöhen Sie in einer Umgebung mit partitionierten Datenbanken oder einer Umgebung, in der abfrageinterne Parallelitäten aktiviert sind, den höheren der beiden Parameter *maxagents* oder *max_coordagents*.

maxcagents - Maximale Anzahl gleichzeitig aktiver Agenten

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Dieser Parameter wird zur Steuerung der Systembelastung in Phasen hoher gleichzeitiger Anwendungsaktivität verwendet, indem die maximale Anzahl von Datenbankmanageragenten festgelegt wird, die gleichzeitig eine Datenbankmanagertransaktion ausführen können.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
-1 (*max_coordagents*) [-1; 1 – *max_coordagents*]

Maßeinheit
Zähler

Dieser Parameter schränkt nicht die Anzahl der Anwendungen ein, die mit einer Datenbank verbunden sein können. Er begrenzt nur die Anzahl der Datenbankmanageragenten, die gleichzeitig vom Datenbankmanager verarbeitet werden können, wodurch der Bedarf an Systemressourcen in Phasen sehr starker Belastung eingeschränkt wird. Sie könnten beispielsweise ein System haben, das eine große Anzahl von Verbindungen anfordert, jedoch nur über eine begrenzte Speicherkapazität zur Verarbeitung dieser Verbindungen verfügt. In diesem Fall kann die Anpassung dieses Parameters sehr nützlich sein, da es hier aufgrund hoher gleichzeitiger Aktivität zu übermäßigem Seitenwechseln durch das Betriebssystem kommen kann.

Der Wert -1 gibt an, dass der Grenzwert gleich dem Wert des Parameters *max_coordagents* ist.

Empfehlung: In den meisten Fällen kann der Standardwert für diesen Parameter übernommen werden. In Fällen, in denen es durch den gleichzeitigen Zugriff zahlreicher Anwendungen zu Problemen kommt, können Sie durch Vergleichstests (Benchmark-Tests) die beste Einstellung für diesen Parameter ermitteln, um die Leistung der Datenbank zu optimieren.

mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor

Mit diesem Parameter wird der Speicherbereich in Seiten festgelegt, der für Daten des Datenbanksystemmonitors zugeordnet werden soll. Der Speicher wird aus dem Monitorzwischenspeicher zugeordnet, wenn Sie eine Datenbankmonitorfunktion ausführen, wie z. B. Erstellen einer Momentaufnahme (Snapshot), Aktivieren eines Monitorschalters oder eines Ereignismonitors, Zurücksetzen eines Monitors o. ä.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass der Monitorzwischenspeicher nach Bedarf erhöht wird, bis der Grenzwert des Parameters *instance_memory* erreicht ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Automatic [0 - 60 000]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn der Datenbankmanager mit dem Befehl db2start gestartet wird

Freigabe
Wenn der Datenbankmanager mit dem Befehl db2stop gestoppt wird

Erhält der Parameter den Wert 0, kann der Datenbankmanager keine Daten des Datenbanksystemmonitors sammeln.

Empfehlung: Die für die Monitorfunktionen erforderliche Speicherkapazität ist von der Anzahl der Überwachungsanwendungen (Anwendungen, die Momentaufnahmen machen, oder Ereignismonitoren), den aktivierten Schaltern und der Datenbankaktivität abhängig.

Wenn die konfigurierte Speicherkapazität in diesem Zwischenspeicher erschöpft und kein weiterer nicht reservierter Speicher mehr verfügbar ist, wird eine der folgenden Aktionen ausgeführt:

- Wenn die erste Anwendung eine Verbindung zu der Datenbank herstellt, für die dieser Ereignismonitor definiert ist, wird eine Fehlernachricht in das Benachrichtigungsprotokoll für die Verwaltung geschrieben.
- Wenn ein Ereignismonitor fehlschlägt, der mithilfe der Anweisung SET EVENT MONITOR dynamisch gestartet wird, wird ein Fehlercode an die Anwendung zurückgegeben.
- Wenn ein Monitorbefehl oder eine API-Unterroutine fehlschlägt, wird ein Fehlercode an die Anwendung zurückgegeben.

nodetype - Knotenart des Systems

Dieser Parameter liefert Informationen zu den DB2-Produkten, die auf Ihrem System installiert sind, und daher auch zur Art Ihrer Datenbankmanagerkonfiguration.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Informativ

Die folgenden Werte können von diesem Parameter zurückgegeben werden, und dieser Knotenart können die folgenden Produkte zugeordnet sein:

- **Datenbankserver mit lokalen und fernen Clients:** Ein DB2-Serverprodukt, das lokale und ferne Data Server Runtime Client-Systeme unterstützt und auf andere ferne Datenbankserver zugreifen kann.
- **Client:** Ein Data Server Runtime Client, der auf ferne Datenbankserver zugreifen kann.
- **Datenbankserver mit lokalen Clients:** Ein Verwaltungssystem für relationale DB2-Datenbanken, das lokale Data Server Runtime Client-Systeme unterstützt und auf andere ferne Datenbankserver zugreifen kann.

- **Partitionierter Datenbankserver mit lokalen und fernen Clients:** Ein DB2-Serverprodukt, das lokale und ferne Data Server Runtime Client-Systeme unterstützt, auf andere ferne Datenbankserver zugreifen kann und parallelitätsfähig ist.

notifylevel - Benachrichtigungsstufe

Dieser Parameter gibt den Typ der Hinweisnachrichten zur Systemverwaltung an, die in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
3 [0 — 4]

Auf Linux- und UNIX-Plattformen ist das Protokoll mit Benachrichtigungen für die Systemverwaltung eine Textdatei mit dem Namen *instanz*.nfy. Unter Windows werden alle Hinweisnachricht für die Systemverwaltung in das Ereignisprotokoll geschrieben. Die Fehler können von DB2, dem Diagnosemonitor, den Capture- und Apply-Programmen sowie von Benutzeranwendungen geschrieben werden.

Gültige Werte für diesen Parameter sind:

- **0** — Keine Aufzeichnung von Hinweisnachrichten zur Systemverwaltung. (Diese Einstellung wird nicht empfohlen.)
- **1** — Schwer wiegende oder nicht behebbare Fehler. Es werden nur schwer wiegende und nicht behebbare Fehler protokolliert. Zur Behebung einiger dieser Fehler benötigen Sie möglicherweise Unterstützung vom DB2-Service.
- **2** — Sofortmaßnahmen erforderlich. Es werden Bedingungen protokolliert, die Sofortmaßnahmen durch den Systemadministrator oder den Datenbankadministrator erfordern. Wenn die Bedingung nicht behoben wird, könnte dies zu einem schwer wiegenden Fehler führen. Auf dieser Ebene können ebenfalls äußerst wichtige, nicht fehlerbezogene Aktivitäten (wie zum Beispiel eine Recovery) protokolliert werden. Auf dieser Ebene werden Alarmnachrichten des Diagnosemonitors erfasst.
- **3** — Wichtige Informationen, keine Sofortmaßnahmen erforderlich. Es werden Bedingungen protokolliert, die nicht bedrohlich sind und keine Sofortmaßnahmen erfordern, die jedoch auf ein nicht optimales System hinweisen können. Auf dieser Ebene werden Alarmnachrichten, Warnungen und Informationsnachrichten des Diagnosemonitors erfasst.
- **4** — Informationsnachrichten.

Das Protokoll mit Benachrichtigungen für die Systemverwaltung enthält Nachrichten mit Werten bis einschließlich dem Wert von *notifylevel*. Wenn der Parameter *notifylevel* beispielsweise auf 3 gesetzt ist, enthält dieses Protokoll Nachrichten der Ebenen 1, 2 und 3.

Eine Benutzeranwendung muss die API 'db2AdminMsgWrite' aufrufen, um in die Benachrichtigungsdatei oder das Windows-Ereignisprotokoll schreiben zu können.

Empfehlung: Sie können den Wert dieses Parameters erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können. Wenn der Diagnosemonitor Benachrichtigung an die in seiner Konfiguration angegebenen Ansprechpartner senden soll, müssen Sie für den Parameter *notifylevel* einen Wert von 2 oder höher angeben.

num_initagents - Anfangswert für die Anzahl Agenten im Pool

Dieser Parameter gibt an, wie viele inaktive Agenten beim Ausführen von DB2START im Agentenpool erstellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
0 [0–64 000]

Der Datenbankmanager startet immer die inaktiven *num_initagents*-Agenten im Rahmen des Befehls 'db2start', außer, der Wert dieses Parameters ist beim Start größer als *num_poolagents* und *num_poolagents* ist nicht auf AUTOMATIC gesetzt. In diesem Fall startet der Datenbankmanager nur die inaktiven num_poolagents-Agenten, da es keinen Grund gibt, mehr inaktive Agenten zu starten als in einen Pool gestellt werden können.

num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse

Dieser Parameter gibt die Anfangszahl der inaktiven db2fmp-Prozesse an, die keine Threadprozesse sind und die zum Startzeitpunkt der Datenbank (DB2START) im db2fmp-Pool erstellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
0 [0–64 000]

Wenn Sie diesen Parameter festlegen, wird die einleitende Startzeit für das Ausführen von nicht threadsicheren C- und COBOL-Routinen verringert. Dieser Parameter wird ignoriert, wenn *keepfenced* nicht angegeben wird.

Es ist viel wichtiger, den Parameter *fenced_pool* auf eine für Ihr System geeignete Größe festzulegen, als zum Startzeitpunkt der Datenbank (DB2START) eine Reihe von db2fmp-Prozessen zu starten.

In früheren Versionen hatte dieser Parameter den Namen *num_initdaris*.

num_poolagents - Agentenpoolgröße

Dieser Parameter definiert die maximale Größe des Pools inaktiver Agenten.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert
100, Automatic [-1, 0–64 000]

Dieser Konfigurationsparameter ist auf AUTOMATIC mit dem Standardwert 100 gesetzt. Die Einstellung -1 wird weiterhin unterstützt und in den Wert 100 umgesetzt. Wenn dieser Parameter auf AUTOMATIC gesetzt ist, verwaltet der Datenbankmanager automatisch die Anzahl inaktiver Agenten, die in einem Pool zusammengefasst werden sollen. In der Regel bedeutet dies Folgendes: Wenn ein Agent seine Arbeit beendet, wird nicht der Agent selbst beendet, sondern er ist nur für einen bestimmten Zeitraum inaktiv. In Abhängigkeit von der Auslastung und dem Typ des Agenten, wird er möglicherweise nach einer bestimmten Zeit beendet.

Wenn Sie AUTOMATIC verwenden, können Sie weiterhin einen Wert für den Konfigurationsparameter num_poolagents angeben. Weitere inaktive Agenten werden immer dann in einem Pool zusammengefasst, wenn die aktuelle Anzahl von inaktiven Agenten, die in einem Pool zusammengefasst sind, kleiner-gleich dem von Ihnen angegebenen Wert ist.

Beispiele:

num_poolagents **ist auf 100 und AUTOMATIC gesetzt**
Wenn ein Agent frei wird, wird er dem Pool inaktiver Agenten hinzugefügt; dort prüft der Datenbankmanager zu einem bestimmten Zeitpunkt, ob er beendet werden sollte. Wenn zu dem Zeitpunkt, zu dem der Datenbankmanager die Beendigung des Agenten in Betracht zieht, die Gesamtzahl der inaktiven Agenten, die in einem Pool zusammengefasst sind, größer als 100 ist, wird dieser Agent beendet. Wenn weniger als 100 inaktive Agenten vorhanden sind, wartet der inaktive Agent weiterhin auf Arbeit. Durch die Verwendung der Einstellung AUTOMATIC können wei-

tere inaktive Agenten (über 100) in einem Pool zusammengefasst werden; dies ist möglicherweise in Zeiten größerer Systemaktivität nützlich, wenn die Häufigkeit von Arbeit in größerem Umfang schwanken kann. In den Fällen, in denen höchstwahrscheinlich weniger als 100 inaktive Agenten zu einer beliebigen Zeit auftreten, werden Agenten garantiert in einem Pool zusammengefasst. Zu Zeiten von weniger Systemaktivität kann dies positiv genutzt werden; der Initialisierungsaufwand für neue Arbeit ist dann geringer.

num_poolagents **wird dynamisch konfiguriert**
Wenn der Wert des Parameters auf einen Wert erhöht wird, der die Anzahl der in einem Pool zusammengefassten Agenten überschreitet, wirkt sich diese Änderung unmittelbar aus. Wenn neue Agenten inaktiviert werden, werden sie in einem Pool zusammengefasst. Wenn der Wert des Parameters verringert wird, verkleinert der Datenbankmanager die Anzahl der Agenten in dem Pool nicht unmittelbar. Stattdessen bleibt die Größe des Pools bestehen und Agenten werden nach der Verwendung beendet und werden wieder inaktiv; dadurch wird die Anzahl der Agenten in dem Pool schrittweise auf den neuen Grenzwert reduziert.

Empfehlung: Für die meisten Umgebungen sind die Standardwerte 0 und AUTOMATIC ausreichend. Wenn Sie bei einer bestimmten Auslastung das Gefühl haben, dass zu viele Agenten erstellt und beendet wurden, können Sie den Wert von *num_poolagents* erhöhen; der Parameter sollte allerdings auf AUTOMATIC gesetzt bleiben.

numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und System i-Datenbanken

Mit diesem Parameter wird die Anzahl der lokalen Datenbanken, die gleichzeitig aktiv sein können (d. h., auf die von Anwendungen zugegriffen werden kann), oder die maximale Anzahl verschiedener Datenbankaliasnamen angegeben, die in einem DB2 Connect-Server katalogisiert werden können.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

UNIX 8 [1 — 256]

Windows-Datenbankserver mit lokalen und fernen Clients
8 [1 — 256]

Windows-Datenbankserver mit lokalen Clients
3 [1 — 256]

Maßeinheit
Zähler

Jede Datenbank belegt Speicher, und eine aktive Datenbank verwendet ein neues Segment des gemeinsam benutzten Speichers.

Empfehlung: In der Regel ist es am besten, diesen Wert auf die tatsächliche Anzahl der bereits für den Datenbankmanager definierten Datenbanken zu setzen und um ca. 10 % für mögliches Wachstum zu erhöhen.

Eine Änderung des Parameters *numdb* kann Auswirkungen auf die Gesamtmenge an Speicher haben, der zugeordnet wird. Daher ist es nicht empfehlenswert, diesen Parameter häufig zu aktualisieren. Beim Aktualisieren dieses Parameters sollten Sie die anderen Konfigurationsparameter beachten, die Speicher für eine Datenbank oder für eine mit der Datenbank verbundenen Anwendung zuordnen können.

query_heap_sz - Größe des Abfragezwischenspeichers

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
1 000 [2 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn eine Anwendung (lokal oder fern) eine Verbindung zur Datenbank herstellt

Freigabe
Wenn die Anwendung die Verbindung zur Datenbank oder zur Instanz trennt

Mit diesem Parameter wird die **maximale** Speichermenge angegeben, die für den Abfragezwischenspeicher zugeordnet werden kann. Dabei wird sichergestellt, dass eine Anwendung nicht unnötig große virtuelle Speicherbereiche innerhalb eines Agenten belegt.

Ein Abfragezwischenspeicher wird zur Speicherung jeder Abfrage im privaten Speicher des Agenten verwendet. Die Informationen für jede Abfrage bestehen aus dem Eingabe- und Ausgabe-SQL-Deskriptorbereich, dem Text der Anweisung, dem SQL-Kommunikationsbereich, dem Paketnamen, dem Ersteller, der Abschnittsnummer und dem Konsistenz-Token.

Aus dem Abfragezwischenspeicher wird auch der Speicher für Blockcursor zugeordnet. Dieser Speicher besteht aus einem Cursorsteuerungsblock und einem vollständig formatierten Ausgabe-SQL-Deskriptorbereich.

Zu Anfang ist der zugeordnete Abfragezwischenspeicher ebenso groß wie der Zwischenspeicher für die Anwendungsunterstützungsebene, der durch den Parameter *aslheapsz* definiert ist. Der Abfragezwischenspeicher muss größer oder gleich 2 sowie größer oder gleich dem Wert des Parameters *aslheapsz* sein. Wenn dieser Abfragezwischenspeicher zur Verarbeitung einer Anforderung nicht ausreicht, wird neuer Speicher in der für die Anforderung erforderlichen Größe (nicht über den Wert *query_heap_sz* hinaus) zugeordnet. Wenn dieser neue Abfragezwischenspeicher mehr als anderthalbmal so groß wie der Wert für *aslheapsz* ist, wird der Abfragezwischenspeicher in der Größe von *aslheapsz* neu zugeordnet, wenn die Abfrage beendet ist.

Empfehlung: In den meisten Fällen ist der Standardwert ausreichend. Als Minimalwert sollte für *query_heap_sz* ein Wert angegeben werden, der mindestens fünfmal so groß wie der Wert für *aslheapsz* ist. Dadurch können Abfragen größer als *aslheapsz* sein, und es wird zusätzlicher Speicher für drei oder vier Blockcursor bereitgestellt, die gleichzeitig geöffnet sein können.

Wenn Sie sehr umfangreiche LOBs (große Objekte) haben, müssen Sie möglicherweise den Wert dieses Parameters erhöhen, damit der Abfragezwischenspeicher groß genug für diese LOBs ist.

release - Release-Level der Datenbankkonfiguration

Dieser Parameter gibt den Release-Level der Konfigurationsdatei an.

Konfigurationstyp
Datenbankmanager, Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Informativ

resync_interval - Intervall für Transaktionsresynchronisation

Mit diesem Parameter wird das Zeitintervall in Sekunden angegeben, während dessen ein Transaktionsmanager (TM), ein Ressourcenmanager (RM) oder ein Synchronisationspunktmanager (SPM) versuchen soll, jede ausstehende unbestätigte Transaktion, die in dem TM, RM oder SPM gefunden wird, wiederherzustellen. Dieser Parameter ist gültig, wenn Sie Transaktionen in einer DUOW-Umgebung (DUOW = Distributed Unit of Work, verteilte Arbeitseinheit) ausführen. Dieser Parameter gilt ebenfalls für die Recovery von Systemen mit föderierten Datenbanken.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
180 [1 - 60 000]

Maßeinheit
Sekunden

Empfehlung: Wenn in Ihrer Umgebung unbestätigte Transaktionen keine Störungen anderer Transaktionen für Ihre Datenbank verursachen, können Sie den Wert dieses Parameters auch erhöhen. Wenn Sie ein DB2 Connect-Gateway zum Zugriff auf DRDA2-Anwendungsserver verwenden, sollten Sie die Auswirkungen bedenken, die unbestätigte Transaktionen auf Anwendungsservern haben könnten, auch wenn lokal keine Störungen beim Datenzugriff zu erwarten sind. Gibt es keine unbestätigten Transaktionen, sind die Auswirkungen auf die Leistung minimal.

rqrioblk - E/A-Blockgröße für Clients

Mit diesem Parameter wird die Größe des Puffers für die Kommunikation zwischen fernen Anwendungen und ihren Datenbankagenten auf dem Datenbankserver festgelegt. Dieser Parameter wird auch verwendet, um die E/A-Blockgröße auf dem Data Server Runtime Client festzulegen, wenn ein Blockcursor geöffnet wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
32 767 [4 096 - 65 535]

Maßeinheit
Byte

Zuordnung

- Wenn eine ferne Client-Anwendung eine Verbindungsanforderung für eine Server-Datenbank absetzt
- Wenn ein Blockcursor geöffnet wird, werden zusätzliche Blöcke auf dem Client geöffnet

Freigabe

- Wenn die ferne Anwendung die Verbindung zur Server-Datenbank trennt
- Wenn der Blockcursor geschlossen wird

Wenn ein Data Server Runtime Client die Verbindung zu einer fernen Datenbank anfordert, wird dieser Kommunikationspuffer auf dem Client zugeordnet. Auf dem Datenbankserver wird zu Anfang ein Kommunikationspuffer von 32.767 Byte zugeordnet, bis eine Verbindung hergestellt ist und der Server den Wert des Parameters *rqrioblk* auf dem Client ermitteln kann. Wenn der Server diesen Wert ermittelt hat, wird der Kommunikationspuffer auf dem Server neu zugeordnet, sofern der Clientpuffer nicht 32.767 Byte groß ist.

Der Speicher für Blockcursor wird aus dem privaten Adressraum der Anwendung zugeordnet. Sie sollten daher die optimale Größe des privaten Speichers ermitteln, der jedem Anwendungsprogramm zugeordnet werden soll. Wenn der Data Server Runtime Client keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.

Empfehlung: Bei Cursorn ohne Blockung könnte ein Grund zur Erhöhung des Werts für diesen Parameter darin bestehen, dass die Daten (z. B. LOB-Daten), die durch eine einzige Abfrageanweisung übertragen werden sollen, so umfangreich sind, dass der Standardwert nicht ausreicht.

Beachten Sie außerdem, welche Auswirkung dieser Parameter auf die Anzahl und die mögliche Größe von Blockcursorn hat. Große Zeilenblöcke können zu einer höheren Leistung führen, wenn die Anzahl oder die Größe der Zeilen, die übertragen werden, groß ist (z. B., wenn die Datenmenge größer als 4.096 Byte ist). Dies hat jedoch den Nachteil, dass größere Datensatzblöcke die Größe des für jede Verbindung benötigten Arbeitsspeichers erhöhen.

Größere Satzblöcke können außerdem mehr Abrufanforderungen verursachen, als tatsächlich für die Anwendung erforderlich wären. Sie können die Anzahl der Abrufanforderungen mit der Klausel OPTIMIZE FOR in der Anweisung SELECT in Ihrer Anwendung steuern.

sheapthres - Schwellenwert für Sortierspeicher

Dieser Parameter ein instanzweiter, veränderlicher Grenzwert für den maximalen Speicherbereich, der zu einem beliebigen Zeitpunkt von privaten Sortiervorgängen beansprucht werden kann. Wenn die Gesamtnutzung an privatem Sortierspeicherbereich für eine Instanz diesen Grenzwert erreicht, wird der für weitere eingehende private Sortieranforderungen zugeordnete Speicherbereich beträchtlich verkleinert.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients
- OLAP-Funktionen

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX-Plattformen (32-Bit)
0 [0 - 2 097 152]

Windows-Plattformen (32-Bit)
0 [0 - 2 097 152]

64-Bit-Plattformen
0 [0 - 2 147 483 647]

Maßeinheit
Seiten (4 KB)

Der Sortierspeicher wird beispielsweise bei folgenden Operationen verwendet: Sortierungen, Hash-Joins, dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND (Index ANDing) und Sternjoins verwendet werden) und Operationen, bei denen sich die Tabelle im Speicher befindet.

Durch die explizite Angabe des Schwellenwerts wird verhindert, dass der Datenbankmanager übermäßig große Speichermengen für sehr viele Sortiervorgänge verwendet.

Es gibt keinen Grund, den Wert dieses Parameter beim Versetzen von einer Umgebung mit nicht partitionierten Datenbanken in eine Umgebung mit partitionierten Datenbanken zu erhöhen. Wenn Sie die Konfigurationsparameter der Datenbank und des Datenbankmanagers in einer Umgebung mit einer nicht partitionierten Datenbank optimiert haben, funktionieren diese Werte in einer Umgebung mit partitionierten Datenbanken zumeist ebenfalls einwandfrei. Sie können diesen Parameter in anderen Knoten oder Datenbankpartitionen nur auf unterschiedliche Werte setzen, indem Sie mehrere DB2-Instanzen erstellen. Dies erfordert das Verwalten verschiedener DB2-Datenbanken in verschiedenen Datenbankpartitionsgruppen. Ein solches Vorgehen macht viele Vorteile einer Umgebung mit partitionierten Datenbanken zunichte.

Wenn der Parameter *sheapthres* der Instanzebene auf den Wert 0 gesetzt ist, erfolgt die Verfolgung der Sortierspeichernutzung nur auf der Datenbankebene und die Speicherzuordnung für Sortiervorgänge wird durch den Wert des Konfigurationsparameters *sheapthres_shr* auf der Datenbankebene begrenzt.

Die automatische Optimierung von *sheapthres_shr* ist nur zulässig, wenn der Datenbankkonfigurationsparameter *sheapthres* auf den Wert 0 gesetzt ist.

Dieser Parameter ist nicht dynamisch aktualisierbar, wenn eine der folgenden Bedingungen zutrifft:

- Der Anfangswert für *sheapthres* ist 0, und der Zielwert ist ungleich 0.
- Der Anfangswert für *sheapthres* ist ungleich 0, und der Zielwert ist 0.

Empfehlung: Im Idealfall sollten Sie den Wert dieses Parameters möglichst auf ein angemessenes Vielfaches des Parameters *sortheap* mit dem größten Wert setzen, der in Ihrer Datenbankmanagerinstanz definiert ist. Der Wert dieses Parameters sollte **mindestens** zweimal so groß sein wie der größte Sortierspeicher (*sortheap*), der für eine Datenbank in der Instanz definiert ist.

Wenn Sie private Sortiervorgänge ausführen und Ihr System über genügend Speicher verfügt, kann der Idealwert für diesen Parameter auf folgende Weise berechnet werden:

1. Berechnen Sie die normale Sortierspeicherbelegung für jede Datenbank:

   ```
   (normale Anzahl Agenten, die gleichzeitig für die Datenbank ausgeführt werden)
    * (sortheap, wie für die Datenbank definiert)
   ```

2. Berechnen Sie die Summe der obigen Ergebnisse. Dies ergibt einen Wert für den gesamten Sortierspeicher, der unter normalen Umständen für alle Datenbanken innerhalb der Instanz verwendet werden kann.

Sie sollten Vergleichstests (Benchmark-Tests) ausführen, um die optimale Einstellung für diesen Parameter zu ermitteln, die Sortierleistung und Speicherbedarf gleichermaßen berücksichtigt.

Mithilfe des Datenbanksystemmonitors können Sie unter Verwendung des Monitorelements *post_threshold_sorts* (Sortierungen nach Erreichen des Schwellenwerts) die Sortieraktivitäten verfolgen.

spm_log_file_sz - Protokolldateigröße für SPM

Mit diesem Parameter wird die Größe der Protokolldatei für den Synchronisationspunktmanager (SPM) in 4-KB-Seiten angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
256 [4 - 1000]

Maßeinheit
Seiten (4 KB)

Die Protokolldatei befindet sich im Unterverzeichnis `spmlog` im Verzeichnis `sqllib` und wird erstellt, wenn der Synchronisationspunktmanager zum ersten Mal gestartet wird.

Empfehlung: Die Protokolldatei des Synchronisationspunktmanagers sollte einerseits groß genug sein, um die Leistung zu gewährleisten, andererseits aber auch klein genug, um die Verschwendung von Speicherbereich zu vermeiden. Die erforderliche Größe hängt von der Anzahl der Transaktionen ab, die geschützte Dialoge verwenden, und von der Häufigkeit, mit der Commits oder Rollbacks ausgeführt werden.

Gehen Sie wie folgt vor, um die Größe der SPM-Protokolldatei zu ändern:

1. Verwenden Sie den Befehl LIST DRDA INDOUBT TRANSACTIONS, um festzustellen, ob es unbestätigte Transaktionen gibt.
2. Wenn es keine unbestätigten Transaktionen gibt, stoppen Sie den Datenbankmanager.
3. Aktualisieren Sie die Konfiguration des Datenbankmanagers mit dem neuen Wert für die Größe der SPM-Protokolldatei.
4. Wechseln Sie in das Verzeichnis `$HOME/sqllib`, und löschen Sie die aktuelle SPM-Protokolldatei mit dem Befehl `rm -fr spmlog`. (Anmerkung: Dies ist der AIX-Befehl. Auf anderen Systemen sind wahrscheinlich andere Löschbefehle erforderlich.)
5. Starten Sie den Datenbankmanager. Beim Start des Datenbankmanagers wird eine neue SPM-Protokolldatei in der angegebenen Größe erstellt.

spm_log_path - Protokolldateipfad für SPM

Dieser Parameter gibt an, in welches Verzeichnis die SPM-Protokolle geschrieben werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
sqllib/spmlog [gültiger Pfad oder gültige Einheit]

Standardmäßig werden die Protokolle in das Verzeichnis sqllib/spmlog geschrieben; dies kann in Umgebungen mit hohem Transaktionsaufkommen zu E/A-Engpässen führen. Verwenden Sie diesen Parameter, damit SPM-Protokolldateien auf eine Platte mit kürzeren Zugriffszeiten als das aktuelle Verzeichnis sqllib/spmlog geschrieben werden. Dadurch wird der gemeinsame Zugriff der SPM-Agenten optimiert.

spm_max_resync - Maximale Anzahl von SPM-Resynchronisationsagenten

Mit diesem Parameter wird die Anzahl der Agenten angegeben, die gleichzeitig Resynchronisationsoperationen ausführen können.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
20 [10 — 256]

spm_name - Name des Synchronisationspunktmanagers

Mit diesem Parameter wird der Name der Instanz des Synchronisationspunktmanagers (SPM) gegenüber dem Datenbankmanager identifiziert.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Vom TCP/IP-Hostnamen abgeleitet

srvcon_auth - Authentifizierungstyp für ankommende Verbindungen auf dem Server

Mit diesem Parameter wird angegeben, wie und wo die Benutzerauthentifizierung bei der Verarbeitung ankommender Verbindungen auf dem Server stattfinden soll. Er dient zum Überschreiben des aktuellen Authentifizierungstyps.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [CLIENT; SERVER; SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

Wenn kein Wert angegeben wird, verwendet DB2 den Wert des Konfigurationsparameters *authentication* des Datenbankmanagers.

Eine Beschreibung der Authentifizierungstypen finden Sie unter „authentication - Authentifizierungstyp" auf Seite 535.

srvcon_gssplugin_list - Liste der GSS-API-Plug-ins für ankommende Verbindungen auf dem Server

Mit diesem Parameter werden die GSS-API-Plug-in-Bibliotheken angegeben, die vom Datenbankserver unterstützt werden. Dieser Parameter verarbeitet ankommende Verbindungen auf dem Server, wenn der Parameter *srvcon_auth* mit dem Wert KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN oder GSS_SERVER_ENCRYPT angegeben ist oder wenn *srvcon_auth* nicht angegeben ist und *authentication* mit dem Wert KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN oder GSS_SERVER_ENCRYPT angegeben ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Dieser Wert ist standardmäßig null. Wenn der Authentifizierungstyp GSSPLUGIN ist und dieser Parameter NULL ist, wird ein Fehler zurückgegeben. Wenn der Authentifizierungstyp KERBEROS ist und dieser Parameter NULL ist, wird das von DB2 bereitgestellte Kerberos-Modul bzw. die Kerberos-Bibliothek verwendet. Bei einem anderen Authentifizierungstyp wird dieser Parameter nicht verwendet.

Wenn der Authentifizierungstyp KERBEROS ist und der Wert dieses Parameters nicht NULL ist, muss die angegebene Liste exakt ein Kerberos-Plug-in enthalten, und dieses Plug-in wird zur Authentifizierung verwendet (alle anderen GSS-Plug-ins in der Liste werden ignoriert). Enthält die Liste mehr als ein Kerberos-Plug-in, wird ein Fehler zurückgegeben.

Jeder Name eines GSS-API-Plug-ins muss durch ein Komma (,) ohne Leerzeichen vor oder nach dem Komma getrennt werden. Die Plug-in-Namen sollten in der bevorzugten Reihenfolge aufgelistet werden.

srvcon_pw_plugin - Plug-in für Benutzer-ID/Kennwort für ankommende Verbindungen auf dem Server

Mit diesem Parameter wird der Name der Benutzer-ID/Kennwort-Plug-in-Standardbibliothek angegeben, die für die serverseitige Authentifizierung zu verwenden ist. Er verarbeitet ankommende Verbindungen auf dem Server, wenn der Parameter *srvcon_auth* mit dem Wert CLIENT, SERVER, SERVER_ENCRYPT oder DATA_ENCRYPT angegeben ist oder wenn *srvcon_auth* nicht angegeben ist und *authentication* mit dem Wert CLIENT, SERVER, SERVER_ENCRYPT oder DATA-_ENCRYPT angegeben ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Standardmäßig ist dieser Wert null und die von DB2 bereitgestellte Benutzer-ID/Kennwort-Plug-in-Bibliothek wird verwendet. Das Plug-in wird für alle Gruppensuchen verwendet. Für Nichtrootinstallationen gilt, dass bei Verwendung der DB2-Bibliothek des Plug-ins für Benutzer-ID und Kennwort der Befehl db2rfe ausgeführt werden muss, bevor das DB2-Produkt verwendet wird.

srv_plugin_mode - Server-Plug-in-Modus

Mit diesem Parameter wird angegeben, ob Plug-ins im abgeschirmten oder nicht abgeschirmten Modus auszuführen sind. Es wird nur der nicht abgeschirmte Modus (UNFENCED) unterstützt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
UNFENCED

start_stop_time - Zeitlimit für DB2START und DB2STOP

Mit diesem Parameter wird die Zeit (in Minuten) angegeben, innerhalb welcher alle Datenbankpartitionsserver auf den Befehl DB2START oder DB2STOP reagieren müssen. Außerdem wird er als Zeitlimitwert für ADD DBPARTITIONNUM-Operationen verwendet.

Konfigurationstyp
Datenbankmanager

Gilt für
Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [1 - 1 440]

Maßeinheit
Minuten

Datenbankpartitionsserver, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2START reagieren, senden eine Nachricht an das Fehlerprotokoll für `db2start`, das sich im Unterverzeichnis `log` des Unterverzeichnisses `sqllib` im Ausgangsverzeichnis für die Instanz befindet. Vor dem Neustart dieser Knoten sollten Sie auf diesen Knoten einen Befehl DB2STOP absetzen.

Datenbankpartitionsserver, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2STOP reagieren, senden eine Nachricht an das Fehlerprotokoll für `db2stop`, das sich im Unterverzeichnis `log` des Unterverzeichnisses `sqllib` im Ausgangsverzeichnis für die Instanz befindet. Sie können `db2stop` entweder für jeden oder einmal für alle nicht reagierenden Datenbankpartitionsserver absetzen. (Die bereits gestoppten Server melden, dass sie bereits gestoppt sind.)

Wenn eine `db2start`- oder eine `db2stop`-Operation in einer Mehrpartitionsdatenbank nicht innerhalb des Zeitwerts ausgeführt wird, der durch den Konfigurationsparameter *start_stop_time* des Datenbankmanagers angegeben wurde, werden die Datenbankpartitionen intern abgebrochen, für die das zulässige Zeitlimit überschritten wurde. Bei Umgebungen mit vielen Datenbankpartitionen und einem geringen Wert für *start_stop_time* kann es zu diesem Verhalten kommen. Erhöhen Sie den Wert für *start_stop_time*, um diesem Verhalten entgegenzuwirken.

Beim Hinzufügen einer neuen Datenbankpartition mithilfe des Befehls DB2START, START DATABASE MANAGER bzw. ADD DBPARTITIONNUM muss die Operation zum Hinzufügen der Datenbankpartition bestimmen, ob die einzelnen Datenbanken in der Instanz für dynamischen Speicher eingerichtet sind. Dies geschieht durch Kommunikation mit der Katalogpartition für jede einzelne Datenbank. Wenn der dynamische Speicher aktiviert ist, werden die Speicherpfaddefinitionen im Rahmen dieser Kommunikation abgerufen. Ähnlich gilt für den Fall, dass Tabellenbereiche für temporäre Tabellen zusammen mit den Datenbankpartitionen zu erstellen sind, dass die Operation möglicherweise mit einem anderen Datenbankpartitionsserver kommunizieren muss, um die Tabellenbereichsdefinitionen für die

Datenbankpartitionen abzurufen, die sich auf diesem Server befinden. Diese Faktoren müssen bei der Festlegung des Werts für den Parameter *start_stop_time* berücksichtigt werden.

svcename - TCP/IP-Servicename

Dieser Parameter enthält den Namen des TCP/IP-Ports, an dem ein Datenbankserver auf Datenübertragungen von fernen Clientknoten wartet. Dieser Name muss der für die Verwendung durch den Datenbankmanager reservierte Port sein.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Damit Verbindungsanforderungen von einem Data Server Runtime Client mithilfe von TCP/IP empfangen werden können, muss der Datenbankserver an einem Port empfangsbereit sein, der diesem Server zugewiesen wurde. Der Systemadministrator für den Datenbankserver muss einen Port (Nummer *n*) reservieren und den zugeordneten TCP/IP-Servicenamen in der Servicedatei auf dem Server definieren.

Der Port (Nummer *n*) des Datenbankservers und sein TCP/IP-Servicename müssen in der Servicedatei auf dem Datenbankclient definiert werden.

Auf Linux- und UNIX-Systemen befindet sich die Servicedatei in folgendem Verzeichnis: `/etc/services`

Der Parameter *svcename* sollte auf den Servicenamen gesetzt werden, der dem Hauptverbindungsport zugeordnet ist, sodass der Datenbankserver nach dem Start ermitteln kann, an welchem Port er für eingehende Verbindungsanforderungen empfangsbereit sein muss.

sysadm_group - SYSADM-Gruppenname

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSADM für die Datenbankmanagerinstanz besitzt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Die Berechtigung SYSADM (Systemadministrator) stellt die höchste Ebene der Berechtigungen innerhalb des Datenbankmanagers dar, von der aus sämtliche Datenbankobjekte gesteuert werden.

Die Berechtigung SYSADM wird von den Sicherheitseinrichtungen festgelegt, die in einer bestimmten Betriebsumgebung verwendet werden.

- Unter dem Windows-Betriebssystem kann dieser Parameter auf eine beliebige lokale Gruppe gesetzt werden. Er wird in der Sicherheitsdatenbank von Windows definiert. Gruppennamen müssen die Längenbegrenzungen einhalten, die in SQL- und XML-Begrenzungen angegeben sind. Wenn für diesen Parameter der Wert „NULL" angegeben wird, haben alle Mitglieder der Administratorengruppe die Berechtigung SYSADM.
- Wenn bei Linux- und UNIX-Systemen der Wert „NULL" für diesen Parameter angegeben wird, gilt die Primärgruppe des Instanzeigners standardmäßig als SYSADM-Gruppe.

  Ist der Wert nicht „NULL", kann als SYSADM-Gruppe jeder gültige UNIX-Gruppenname definiert werden.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSADM_GROUP NULL aus. Das Schlüsselwort „NULL" muss in Großbuchstaben angegeben werden.

sysctrl_group - SYSCTRL-Gruppenname

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSCTRL (Systemsteuerung) besitzt. Die Berechtigung SYSCTRL umfasst Zugriffsrechte, die sich auf Systemressourcen auswirkende Operationen ermöglichen, aber keinen direkten Zugriff auf Daten zulassen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Gruppennamen werden auf allen Plattformen akzeptiert, solange sie die in SQL- und XML-Begrenzungen angegebenen Längenbegrenzungen einhalten.

Achtung: Dieser Parameter muss für Windows-Clients den Wert NULL besitzen, wenn die Systemsicherheit verwendet wird (d. h., **authentication** hat den Wert CLIENT, SERVER oder den Wert einer anderen gültigen Authentifizierung). Der Grund hierfür ist, dass Windows-Betriebssysteme keine Gruppeninformationen speichern und somit keine Möglichkeit bieten festzustellen, ob ein Benutzer ein Mitglied einer bestimmten SYSCTRL-Gruppe ist. Bei Angabe eines Gruppennamens kann keiner der Benutzer dieser Gruppe angehören.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSCTRL_GROUP NULL aus. Das Schlüsselwort NULL muss in Großbuchstaben angegeben werden.

sysmaint_group - SYSMAINT-Gruppenname

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSMAINT (Systempflege) besitzt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Mit der Berechtigung SYSMAINT können Operationen zur Pflege aller Datenbanken, die zu einer Instanz gehören, durchgeführt werden, jedoch ohne direkten Zugriff auf Daten.

Gruppennamen werden auf allen Plattformen akzeptiert, solange sie die in SQL- und XML-Begrenzungen angegebenen Längenbegrenzungen einhalten.

Achtung: Dieser Parameter muss für Windows-Clients den Wert NULL besitzen, wenn die Systemsicherheit verwendet wird (d. h., **authentication** hat den Wert CLIENT, SERVER oder den Wert einer anderen gültigen Authentifizierung). Der Grund hierfür ist, dass Windows-Betriebssysteme keine Gruppeninformationen speichern und somit keine Möglichkeit bieten festzustellen, ob ein Benutzer ein Mitglied einer bestimmten SYSMAINT-Gruppe ist. Bei Angabe eines Gruppennamens kann keiner der Benutzer dieser Gruppe angehören.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSMAINT_GROUP NULL aus. Das Schlüsselwort NULL muss in Großbuchstaben angegeben werden.

sysmon_group - Berechtigungsgruppenname für Systemmonitor

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSMON (Systemmonitor) besitzt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
NULL

Benutzer, die über die Berechtigung SYSMON auf der Instanzebene verfügen, können über den Datenbanksystemmonitor Momentaufnahmen von einer Datenbankmanagerinstanz oder den zugehörigen Datenbanken erstellen. Die Berechtigung SYSMON ermöglicht die Ausführung der folgenden Befehle:

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

Benutzer mit der Berechtigung SYSADM, SYSCTRL oder SYSMAINT haben automatisch die Möglichkeit, Momentaufnahmen über den Datenbanksystemmonitor zu erstellen und diese Befehle zu verwenden.

Gruppennamen werden auf allen Plattformen akzeptiert, solange sie die in SQL- und XML-Begrenzungen angegebenen Längenbegrenzungen einhalten.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSMON_GROUP NULL aus. Das Schlüsselwort NULL muss in Großbuchstaben angegeben werden.

tm_database - Name für Transaktionsmanagerdatenbank

Mit diesem Parameter wird der Name der Transaktionsmanagerdatenbank (TMD) für jede DB2-Instanz angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
1ST_CONN [beliebiger gültiger Datenbankname]

Eine Transaktionsmanagerdatenbank kann eine der folgenden Datenbanken sein:

- Eine lokale DB2-Datenbank
- Eine ferne DB2-Datenbank, die sich nicht auf einem Host oder System AS/400 befindet

- Eine Datenbank von DB2 für OS/390 Version 5, wenn auf sie über TCP/IP zugegriffen und der Synchronisationspunktmanager (SPM) nicht verwendet wird

Es handelt sich dabei um eine Datenbank, die zu Protokoll- und Koordinierungsfunktionen verwendet wird und die zur Recovery unbestätigter Transaktionen dient.

Dieser Parameter kann auf den Wert **1ST_CONN** gesetzt werden, wodurch festgelegt wird, dass die Transaktionsmanagerdatenbank die erste Datenbank ist, zu der ein Benutzer eine Verbindung herstellt.

Empfehlung: Zur Vereinfachung der Verwaltung und des Betriebs können Sie einige Datenbanken in verschiedenen Instanzen erstellen und diese Datenbanken ausschließlich als Transaktionsmanagerdatenbanken verwenden.

tp_mon_name - Name des Transaktionsprozessormonitors

Mit diesem Parameter wird der Name des verwendeten Monitors für das Transaktionsprogramm (TP) angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Kein Standardwert

Gültige Werte

- CICS
- MQ
- ENCINA
- CB
- SF
- TUXEDO
- TOPEND
- Kein Eintrag oder ein weiterer Wert (unter UNIX und Windows; keine weiteren gültigen Werte unter Solaris oder SINIX)

- Sind Anwendungen in einer WebSphere Enterprise Server Edition CICS-Umgebung aktiv, muss für den Parameter „CICS" angegeben werden.
- Sind Anwendungen in einer WebSphere Enterprise Server Edition Encina-Umgebung aktiv, muss für den Parameter „ENCINA" angegeben werden.
- Sind Anwendungen in einer WebSphere Enterprise Server Edition Component Broker-Umgebung aktiv, muss für den Parameter „CB" angegeben werden.
- Sind Anwendungen in einer IBM MQSeries-Umgebung aktiv, muss für den Parameter „MQ" angegeben werden.
- Sind Anwendungen in einer BEA Tuxedo-Umgebung aktiv, muss für den Parameter „TUXEDO" angegeben werden.

- Sind Anwendungen in einer IBM San Francisco-Umgebung aktiv, muss für den Parameter „SF" angegeben werden.

Benutzer von **IBM WebSphere EJB** und **Microsoft Transaction Server** müssen für diesen Parameter keinen Wert konfigurieren.

Wenn keines der oben genannten Produkte verwendet wird, sollte dieser Parameter nicht konfiguriert, sondern ohne Angabe eines Werts belassen werden.

In vorangehenden Versionen von IBM DB2 unter Windows enthielt dieser Parameter den Pfad und den Namen der DLL, in der die Funktionen *ax_reg* und *ax_unreg* des XA-Transaktionsmanagers gespeichert waren. Dieses Format wird noch immer unterstützt. Wenn der Wert für diesen Parameter mit keinem der oben genannten TP-Monitornamen übereinstimmt, wird angenommen, dass der angegebene Wert der Name einer Bibliothek ist, die die Funktionen *ax_reg* und *ax_unreg* enthält. Dies gilt für UNIX- und Windows-Umgebungen.

Benutzer von TXSeries CICS und Encina: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „libEncServer:C" oder „libEncServer:E" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „CICS" oder „ENCINA" konfiguriert wird, ist dies ausreichend.

Benutzer von MQSeries: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „mqmax" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „MQ" konfiguriert wird, ist dies ausreichend.

Benutzer von Component Broker: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „somtrx1i" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „CB" konfiguriert wird, ist dies ausreichend.

Benutzer von San Francisco: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „ibmsfDB2" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „SF" konfiguriert wird, ist dies ausreichend.

Die maximale Länge der Zeichenfolge, die für diesen Parameter angegeben werden kann, beträgt 19 Zeichen.

Diese Informationen können auch in der Zeichenfolge XA OPEN von IBM DB2 Version 9.1 konfiguriert werden. Wenn mehrere TP-Monitore in einer einzigen DB2-Instanz verwendet werden, muss diese Funktion verwendet werden.

trust_allclnts - Alle Clients akzeptieren

Dieser Parameter und der Parameter *trust_clntauth* werden verwendet, um festzustellen, wo Benutzerberechtigungen für die Datenbankumgebung überprüft werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
YES [NO, YES, DRDAONLY]

Dieser Parameter ist nur aktiviert, wenn der Parameter *authentication* auf CLIENT gesetzt ist.

Durch Übernehmen des Standardwerts „YES" für diesen Parameter werden alle Clients als gesicherte Clients akzeptiert. Dies bedeutet, der Server geht davon aus, dass auf jedem Client eine Sicherheitsstufe vorhanden ist und auf jedem Client eine Gültigkeitsprüfung für Benutzer ausgeführt werden kann.

Dieser Parameter kann nur auf „NO" gesetzt werden, wenn der Parameter *authentication* auf CLIENT gesetzt ist. Ist dieser Parameter auf „NO" gesetzt, müssen die ungesicherten Clients beim Herstellen einer Verbindung zum Server eine Benutzer-ID mit Kennwort angeben. Ungesicherte Clients sind Betriebssystemplattformen, die nicht über ein Sicherheitssubsystem zur Gültigkeitsprüfung für Benutzer verfügen.

Das Setzen dieses Parameters auf „DRDAONLY" schützt vor allen Clients mit Ausnahme von Clients von DB2 für OS/390 und z/OS, DB2 für VM und VSE sowie DB2 für OS/400. Nur diese Clients dürfen die Authentifizierung auf der Clientseite ausführen. Alle anderen Clients müssen eine Benutzer-ID und ein Kennwort bereitstellen, wobei die Authentifizierung vom Server durchgeführt wird.

Wenn *trust_allclnts* auf „DRDAONLY" gesetzt ist, wird mit dem Parameter *trust_clntauth* ermittelt, wo die Authentifizierung der Clients erfolgt. Wenn *trust_clntauth* auf „CLIENT" gesetzt ist, findet die Authentifizierung auf dem Client statt. Wenn *trust_clntauth* auf „SERVER" gesetzt ist, erfolgt die Authentifizierung auf dem Client, sofern kein Kennwort angegeben ist, und auf dem Server, sofern ein Kennwort angegeben ist.

trust_clntauth - Authentifizierung gesicherter Clients

Dieser Parameter gibt an, ob eine Authentifizierung gesicherter Clients auf dem Server oder auf dem Client erfolgt, wenn der Client eine Benutzer-ID mit Kennwort für eine Verbindung angibt. Dieser Parameter (und *trust_allclnts*) ist nur aktiviert, wenn der Parameter *authentication* auf CLIENT gesetzt ist. Wird keine Benutzer-ID mit Kennwort angegeben, geht das System davon aus, dass die Authentifizierung des Benutzers vom Client ausgeführt wurde, d. h., auf dem Server erfolgt keine weitere Überprüfung.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
CLIENT [CLIENT, SERVER]

Ist dieser Parameter auf CLIENT (die Standardeinstellung) gesetzt, kann der gesicherte Client eine Verbindung herstellen, ohne eine Benutzer-ID mit Kennwort anzugeben, und es wird angenommen, dass die Authentifizierung des Benutzers bereits vom Betriebssystem vorgenommen wurde. Ist der Parameter auf SERVER gesetzt, werden Benutzer-ID und Kennwort auf dem Server überprüft.

Der numerische Wert für CLIENT ist 0. Der numerische Wert für SERVER ist 1.

util_impact_lim - Richtlinie für Instanzauslastungswirkung

Dieser Parameter ermöglicht dem Datenbankadministrator (DBA), die Leistungseinbußen für eine Auslastung durch ein gedrosseltes Dienstprogramm zu begrenzen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen Clients
- Datenbankserver mit lokalen und fernen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [1 - 100]

Maßeinheit
Prozentsatz der zulässigen Wirkung auf die Auslastung

Wenn die Leistungseinbußen begrenzt sind, kann der Datenbankadministrator während kritischer Auslastungszeiten des Produktionssystems Onlinedienstprogramme ausführen und so sicher sein, dass die Auswirkung auf die Produktionsleistung innerhalb annehmbarer Grenzen bleibt.

Zum Beispiel kann ein Datenbankadministrator, der den Parameter *util_impact_lim* (Auslastungswirkung) auf den Wert 10 setzt, davon ausgehen, dass ein gedrosselter BACKUP-Aufruf die Auslastung mit nicht mehr als 10 Prozent beansprucht.

Wenn der Parameter *util_impact_lim* auf den Wert 100 gesetzt ist, findet keine Drosselung von Dienstprogrammaufrufen statt. In diesem Fall können die Dienstprogramme die Auslastung in willkürlichem (und unerwünschtem) Ausmaß beanspruchen. Wenn der Parameter *util_impact_lim* auf einen Wert unter 100 gesetzt wird, ist es möglich, Dienstprogramme im gedrosselten Modus aufzurufen. Zur Ausführung im gedrosselten Modus muss ein Dienstprogramm außerdem mit einer Priorität ungleich Null aufgerufen werden.

Empfehlung: Für die meisten Benutzer ist es wahrscheinlich vorteilhaft, wenn der Parameter *util_impact_lim* auf einen niedrigen Wert (zum Beispiel zwischen 1 und 10) gesetzt wird.

Ein gedrosseltes Dienstprogramm benötigt in der Regel mehr Ausführungszeit als ein ungedrosseltes Dienstprogramm. Wenn Sie feststellen, dass ein Dienst-

programm extrem viel Zeit benötigt, erhöhen Sie den Wert des Parameters *util_impact_lim* oder inaktivieren die Drosselung völlig, indem Sie den Parameter *util_impact_lim* auf den Wert 100 setzen.

Konfigurationsparameter der Datenbank

alt_collate - Alternative Sortierfolge

Mit diesem Parameter wird die Sortierfolge angegeben, die für Unicode-Tabellen in einer Nicht-Unicode-Datenbank zu verwenden ist.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [IDENTITY_16BIT]

Unicode-Tabellen und -Routinen können erst in einer Nicht-Unicode-Datenbank erstellt werden, wenn dieser Parameter definiert wurde. Wenn dieser Parameter einmal definiert ist, kann er nicht mehr geändert oder zurückgesetzt werden.

Dieser Parameter kann nicht für Unicode-Datenbank definiert werden.

app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert. In Version 9.5 wurde er durch den Konfigurationsparameter *appl_memory* ersetzt.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Bei partitionierten und nicht partitionierten Datenbanken mit aktivierter partitionsinterner Parallelität (intra_parallel=ON) gibt dieser Parameter die durchschnittliche Größe des gemeinsam genutzten Speicherbereichs an, der einer Anwendung zugeordnet ist. Bei nicht partitionierten Datenbanken, bei denen die partitionsinterne Parallelität nicht aktiviert ist (intra_parallel=OFF), ist dies der maximale private Speicher, der dem Zwischenspeicher zugeordnet wird. Pro Verbindung einer Datenbankpartition gibt es einen Zwischenspeicher für Anwendungssteuerung.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

Datenbankserver mit lokalen und fernen Clients

- 128 [1 - 64 000] wenn INTRA_PARALLEL nicht aktiviert wurde
- 512 [1 - 64 000] wenn INTRA_PARALLEL aktiviert wurde

Datenbankserver mit lokalen Clients

- 64 [1 - 64 000] (für Nicht-UNIX-Plattformen), wenn INTRA-_PARALLEL nicht aktiviert wurde
- 512 [1 - 64 000] (für Nicht-UNIX-Plattformen), wenn INTRA-_PARALLEL aktiviert wurde
- 128 [1 - 64 000] (für Linux- und UNIX-Plattformen) wenn INTRA_PARALLEL nicht aktiviert wurde
- 512 [1 - 64 000] (für Linux- und UNIX-Plattformen) wenn INTRA_PARALLEL aktiviert wurde

Partitionierter Datenbankserver mit lokalen und fernen Clients
512 [1 - 64 000]

Maßeinheit
Seiten (4 KB)

Zuordnung
Beim Starten einer Anwendung

Freigabe
Beim Beenden einer Anwendung

Der Zwischenspeicher für Anwendungssteuerung wird in erster Linie für die gemeinsame Nutzung von Informationen durch Agenten benötigt, die für dieselbe Anforderung arbeiten. Die Auslastung dieses Zwischenspeichers ist bei nicht partitionierten Datenbanken minimal, wenn Abfragen mit einem Parallelitätsgrad gleich 1 ausgeführt werden.

Dieser Zwischenspeicher wird auch zum Speichern von Deskriptorinformationen für deklarierte temporäre Tabellen verwendet. Die Deskriptorinformationen für alle deklarierten temporären Tabellen, die nicht explizit gelöscht wurden, werden in diesem Zwischenspeicher aufbewahrt und können erst nach dem Löschen der deklarierten temporären Tabelle gelöscht werden.

Empfehlung: Verwenden Sie zunächst den Standardwert. Sie müssen den Wert eventuell erhöhen, wenn Sie komplexe Anwendungen ausführen oder ein System mit zahlreichen Datenbankpartitionen bzw. deklarierte temporäre Tabellen verwenden. Die erforderliche Speicherkapazität erhöht sich mit der Anzahl deklarierter temporärer Tabellen, die gleichzeitig aktiv sind. Der Tabellendeskriptor einer deklarierten temporären Tabelle mit vielen Spalten ist größer als jener einer Tabelle mit wenigen Spalten. Daher erhöht eine große Anzahl Spalten in den deklarierten temporären Tabellen einer Anwendung auch den Bedarf an Zwischenspeicher zur Anwendungssteuerung.

appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert. In Version 9.5 wurde er durch den Konfigurationsparameter *appl_memory* ersetzt.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Dieser Parameter legt die Größe des von der Anwendungsgruppe gemeinsam genutzten Speichersegments fest.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

UNIX-Datenbankserver mit lokalen Clients (nicht 32-Bit-HP-UX)
20 000 [1 - 1 000 000]

32-Bit-HP-UX

- Datenbankserver mit lokalen Clients
- Datenbankserver mit lokalen und fernen Clients
- Partitionierter Datenbankserver mit lokalen und fernen Clients

10 000 [1 - 1 000 000]

Windows-Datenbankserver mit lokalen Clients
10 000 [1 - 1 000 000]

Datenbankserver mit lokalen und fernen Clients (nicht 32-Bit-HP-UX)
30 000 [1 - 1 000 000]

Partitionierter Datenbankserver mit lokalen und fernen Clients (nicht 32-Bit-HP-UX)
40 000 [1 - 1 000 000]

Maßeinheit
Seiten (4 KB)

Informationen, die den für dieselbe Anwendung arbeitenden Agenten gemeinsam zur Verfügung stehen müssen, werden im gemeinsam genutzten Speichersegment der Anwendungsgruppe gespeichert.

In einer partitionierten Datenbank oder in einer nicht partitionierten Datenbank mit aktivierter partitionsinterner Parallelität oder aktiviertem Konzentrator nutzen mehrere Anwendungen eine Anwendungsgruppe gemeinsam. Der Anwendungsgruppe wird ein gemeinsam genutztes Speichersegment zugeordnet. Innerhalb des gemeinsam genutzten Speichersegments der Anwendungsgruppe ist für jede Anwendung ein eigener Zwischenspeicher für die Anwendungssteuerung vorhanden, und alle Anwendungen der Anwendungsgruppe nutzen gemeinsam einen Zwischenspeicher.

Die Anzahl Anwendungen in einer Anwendungsgruppe wird wie folgt berechnet:

```
appgroup_mem_sz / app_ctl_heap_sz
```

Die Größe des von der Anwendungsgruppe gemeinsam genutzten Zwischenspeichers wird wie folgt berechnet:

```
appgroup_mem_sz * groupheap_ratio / 100
```

Die Größe des Zwischenspeichers für die Anwendungssteuerung wird für jede Anwendung wie folgt berechnet:

```
app_ctl_heap_sz * (100 - groupheap_ratio) / 100
```

Empfehlung: Behalten Sie den Standardwert für diesen Parameter bei, solange Sie keine Leistungsprobleme feststellen.

appl_memory - Anwendungsspeicher (Konfigurationsparameter)

Mit diesem Parameter können Datenbankadministratoren (DBAs) und unabhängige Softwareanbieter (ISVs) die maximale Menge an Anwendungsspeicher steuern, die Serviceanwendungsanforderungen von DB2-Datenbankagenten zugeordnet wird. Standardmäßig ist der Wert dieses Parameters auf AUTOMATIC gesetzt, was bedeutet, dass alle Anforderungen für Anwendungsspeicher zulässig sind, so lange die gesamte von der Datenbankpartition zugeordnete Speicherkapazität sich innerhalb der Grenzwerte des Parameters *instance_memory* bewegt.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Automatic [128 - 4 294 967 295]

Maßeinheit
Seiten (4 KB)

Zuordnung
Während der Aktivierung der Datenbank

Freigabe
Während der Inaktivierung der Datenbank

Anmerkung: Wenn *appl_memory* auf AUTOMATIC gesetzt wurde, ist die anfängliche Anwendungspeicherzuordnung bei Aktivierung der Datenbank nur gering, sie nimmt jedoch nach Bedarf zu (oder ab). Diese Änderung wird im Speicher angewendet. Der Wert des Parameters *appl_memory* ändert sich auf der Platte nicht, wie sich durch den Befehl db2 get db cfg show detail anzeigen lässt. Bei der nächsten Aktivierung wird der Wert neu berechnet. Wenn der Parameter *appl_memory* auf einen bestimmten Wert gesetzt wurde, wird die erforderliche Speicherkapazität von Anfang an bei Aktivierung der Datenbank zugeordnet, und die Größe des Anwendungsspeichers ändert sich nicht. Wenn die anfängliche Anwendungsspeicherkapazität vom Betriebssystem nicht zugeordnet werden kann oder den Grenzwert des Parameters *instance_memory* überschreitet, schlägt die Aktivierung der Datenbank mit dem folgenden SQL-Fehler fehl: SQL1084C Gemeinsam genutzte Speichersegmente können nicht zugeordnet werden.

applheapsz - Zwischenspeichergröße für Anwendungen

In früheren Releases verwies der Datenbankkonfigurationsparameter *applheapsz* auf die Größe des Anwendungsspeichers, die die einzelnen für die Anwendung aktiven Datenbankagenten in Anspruch nehmen durften. Seit Version 9.5 verweist *applheapsz* auf die gesamte Anwendungsspeichergröße, die von der gesamten Anwendung in Anspruch genommen werden darf. Für DPF, den Konzentrator oder SMP-Konfigurationen bedeutet dies, dass der in früheren Releases benutzte

Wert für den Parameter *applheapsz* trotz gleicher Arbeitslast erhöht werden muss, es sei denn, die Einstellung AUTOMATIC wird verwendet.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass er nach Bedarf erhöht wird, bis entweder der Grenzwert des Parameters *appl_memory* oder des Parameters *instance_memory* erreicht ist.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Automatic [16 - 60 000]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn eine Anwendung einer Datenbank zugeordnet wird oder zu ihr eine Verbindung herstellt.

Freigabe
Wenn die Zuordnung der Anwendung zur Datenbank aufgehoben oder die Verbindung der Anwendung zur Datenbank unterbrochen wird.

Anmerkung: Dieser Parameter definiert die maximale Größe des Anwendungszwischenspeichers. Pro Datenbankanwendung wird ein Anwendungszwischenspeicher zugeordnet, wenn die Anwendung das erste Mal mit der Datenbank verbunden wird. Der Zwischenspeicher wird von allen Datenbankagenten, die für die Anwendung aktiv sind, gemeinsam genutzt. (In früheren Releases ordneten sich die Datenbankagenten jeweils ihre eigenen Anwendungszwischenspeicher zu). Der Anwendungszwischenspeicher ordnet die Speicherkapazität, die für die Verarbeitung der Anwendung notwenig ist, gemäß dem Grenzwert zu, der über den Parameter definiert ist. Wenn die Einstellung AUTOMATIC gewählt wurde, wird der Anwendungszwischenspeicher bei Bedarf vergrößert, bis entweder der Grenzwert *appl_memory* für die Datenbank oder der Grenzwert *instance_memory* für die Datenbankpartition erreicht ist. Wenn die Verbindung der Anwendung zur Datenbank unterbrochen wird, wird der gesamte Anwendungszwischenspeicher freigegeben.

Der online geänderte Wert wird beim Grenzwert einer Anwendungsverbindung wirksam, d. h. wenn der Wert dynamisch geändert wurde, verwenden die zu dem Zeitpunkt verbundenen Anwendungen noch den alten Wert, während alle neu verbundenen Anwendungen den neuen Wert benutzen.

archretrydelay - Wiederholungsintervall für Archivierung bei Fehler

Mit diesem Parameter wird die Anzahl von Sekunden angegeben, die nach einem fehlgeschlagenen Archivierungsversuch abzuwarten ist, bevor die Archivierung der Protokolldatei erneut versucht wird.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients

- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
20 [0 - 65 535]

Nachfolgende Wiederholungen finden nur statt, wenn der Datenbankkonfigurationsparameter *numarchretry* mindestens auf den Wert 1 gesetzt ist.

auto_del_rec_obj - Automatisches Löschen von Recovery-Objekten (Konfigurationsparameter)

Dieser Parameter gibt an, ob Datenbankprotokolldateien, Backup-Images und Ladekopieimages gelöscht werden sollen, wenn ihr zugeordneter Eintrag in der Datei des Recoveryprotokolls bereinigt wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
OFF [ON; OFF]

Sie können die Einträge in der Datei des Recoveryprotokolls mithilfe des Befehls PRUNE HISTORY oder der API db2Prune bereinigen. Sie können auch den IBM Data Server-Datenbankmanager so konfigurieren, dass die Datei des Recoveryprotokolls nach jedem Datenbank-Backup automatisch bereinigt wird. Wenn Sie den Datenbankkonfigurationsparameter *auto_del_rec_obj* auf ON setzen, löscht der Datenbankmanager beim Bereinigen der Protokolldatei auch die entsprechenden physischen Protokolldateien, Backup-Images sowie Ladekopieimages. Der Datenbankmanager kann nur Recoveryobjekte, wie z. B. Datenbankprotokolle, Backup-Images sowie Ladekopieimages, löschen, wenn es sich bei Ihrem Speichermedium um eine Platte handelt oder wenn Sie einen Speichermanager, wie z. B. Tivoli Storage Manager, verwenden.

auto_maint - Automatische Verwaltung

Dieser Parameter ist das übergeordnete Element für alle anderen Datenbankkonfigurationsparameter der automatischen Verwaltung (*auto_db_backup*, *auto_tbl_maint*, *auto_runstats*, *auto_stats_prof*, *auto_stmt_stats*, *auto_prof_upd* und *auto_reorg*).

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
ON [ON; OFF]

Wenn dieser Parameter inaktiviert wird, werden alle untergeordneten Parameter ebenfalls inaktiviert, jedoch ändern sich ihre Einstellungen, die in der Datenbankkonfigurationsdatei aufgezeichnet sind, nicht. Wenn dieser übergeordnete Parameter aktiviert wird, werden die aufgezeichneten Werte für die untergeordneten Parameter wirksam. Auf diese Weise kann die automatische Verwaltung global aktiviert bzw. inaktiviert werden.

Standardmäßig ist dieser Parameter auf den Wert ON gesetzt.

Sie können einzelne Funktionen der automatischen Verwaltung unabhängig voneinander aktivieren oder inaktivieren, indem Sie die folgenden Parameter definieren:

auto_db_backup
Dieser Parameter der automatischen Verwaltung aktiviert bzw. inaktiviert automatische Backup-Operationen für eine Datenbank. Eine Backup-Richtlinie (eine definierte Gruppe von Regeln oder Bestimmungen) kann zur Angabe der automatischen Funktionsweise verwendet werden. Die Backup-Richtlinie hat den Zweck, sicherzustellen, dass die Datenbank regelmäßig gesichert wird. Die Backup-Richtlinie für eine Datenbank wird automatisch erstellt, wenn der DB2-Diagnosemonitor zum ersten Mal ausgeführt wird. Standardmäßig ist dieser Parameter auf den Wert OFF gesetzt. Zur Aktivierung muss dieser Parameter auf den Wert ON gesetzt werden und der übergeordnete Parameter muss ebenfalls aktiviert sein.

auto_tbl_maint
Dieser Parameter ist das übergeordnete Element für alle Tabellenverwaltungsparameter (*auto_runstats*, *auto_stats_prof*, *auto_prof_upd* und *auto_reorg*). Wenn dieser Parameter inaktiviert wird, werden alle untergeordneten Parameter ebenfalls inaktiviert, jedoch ändern sich ihre Einstellungen, die in der Datenbankkonfigurationsdatei aufgezeichnet sind, nicht. Wenn dieser übergeordnete Parameter aktiviert wird, werden die aufgezeichneten Werte für die untergeordneten Parameter wirksam. Auf diese Weise kann die Tabellenverwaltung global aktiviert bzw. inaktiviert werden.

Standardmäßig ist dieser Parameter auf den Wert ON gesetzt.

auto_runstats
Dieser Parameter der automatischen Tabellenverwaltung aktiviert bzw. inaktiviert automatische RUNSTATS-Operationen für Tabellen in einer Datenbank. Eine Richtlinie (eine definierte Gruppe von Regeln oder Bestimmungen) für die Statistikerstellung kann zur Angabe der automatischen Funktionsweise verwendet werden. Die vom Dienstprogramm RUNSTATS erfassten Statistiken werden vom Optimierungsprogramm zur Bestimmung des effizientesten Plans für den Zugriff auf die physischen Daten verwendet. Zur Aktivierung muss dieser Parameter auf den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf den Wert ON gesetzt.

auto_stats_prof

Dieser Parameter der automatischen Tabellenverwaltung aktiviert die Statistikprofilgenerierung, die dazu dient, Anwendungen zu unterstützen, deren Auslastungen komplexe Abfragen, zahlreiche Vergleichselemente, Joins und Gruppierungen unter Einbeziehung verschiedener Tabellen beinhalten. Zur Aktivierung muss dieser Parameter auf den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf den Wert OFF gesetzt.

auto_stmt_stats

Mit diesem Parameter wird die Erfassung von Echtzeitstatistiken aktiviert und inaktiviert. Dieser Parameter ist ein untergeordneter Parameter des Konfigurationsparameters *auto_runstats*. Diese Funktion wird nur aktiviert, wenn auch die übergeordnete Funktion, also der Konfigurationsparameter *auto_runstats*, aktiviert ist. Beispiel: Wenn *auto_stmt_stats* aktiviert werden soll, müssen *auto_maint*, *auto_tbl_maint* und *auto_runstats* auf ON gesetzt werden. Um den untergeordneten Wert beizubehalten, kann der Konfigurationsparameter *auto_runstats* auf ON gesetzt sein, während der Konfigurationsparameter *auto_maint* auf OFF gesetzt ist. Die entsprechende Funktion für die automatische Statistikerstellung ist weiterhin auf OFF gesetzt.

Unter der Annahme, dass die Funktionen für die automatische Statistikerstellung und für die automatische Reorganisation aktiviert sind, gelten die folgenden Einstellungen:

```
Automatische Verwaltung                        (AUTO_MAINT) = ON
  Automatisches Datenbankbackup              (AUTO_DB_BACKUP) = OFF
    Automatische Tabellenverwaltung          (AUTO_TBL_MAINT) = ON
      Automatische Statistikerstellung        (AUTO_RUNSTATS) = ON
        Automatische Anweisungsstatistik   (AUTO_STMT_STATS) = OFF
      Automatische Statistikprofilgener.    (AUTO_STATS_PROF) = OFF
        Automatische Profilaktualisierungen   (AUTO_PROF_UPD) = OFF
        Automatische Reorganisation              (AUTO_REORG) = ON
```

Sie können sowohl die Funktion für die automatische Statistikerstellung als auch für die automatische Reorganisation vorübergehend inaktivieren; hierfür müssen Sie *auto_tbl_maint* auf OFF setzen. Beide Funktionen können später wieder aktiviert werden; hierfür müssen Sie *auto_tbl_maint* wieder auf ON setzen. Damit die Änderungen wirksam werden, ist es nicht notwendig, die Befehle db2stop oder db2start abzusetzen.

Standardmäßig ist dieser Parameter auf den Wert OFF gesetzt.

auto_prof_upd

Dieser Parameter der automatischen Tabellenverwaltung (ein untergeordneter Parameter von *auto_stats_prof*) gibt an, dass das RUNSTATS-Profil mit Empfehlungen zu aktualisieren ist. Wenn dieser Parameter inaktiviert wird, werden Empfehlungen in der Tabelle opt_feedback_ranking gespeichert, die Sie einsehen können, wenn Sie das RUNSTATS-Profil manuell aktualisieren. Zur Aktivierung muss dieser Parameter auf den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf den Wert OFF gesetzt.

auto_reorg

Dieser Parameter der automatischen Tabellenverwaltung aktiviert bzw. inaktiviert die automatische Tabellen- und Indexreorganisation für eine Datenbank. Eine Reorganisationsrichtlinie (eine definierte Gruppe von

Regeln oder Bestimmungen) kann zur Angabe der automatischen Funktionsweise verwendet werden. Zur Aktivierung muss dieser Parameter auf den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf den Wert OFF gesetzt.

autorestart - Automatischer Neustart aktiviert

Mit diesem Parameter wird festgelegt, ob der Datenbankmanager bei einer abnormalen Beendigung der Datenbank automatisch das Dienstprogramm zum Neustarten der Datenbank aufrufen darf, solange eine Anwendung mit der Datenbank verbunden ist.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
On [On; Off]

Das Dienstprogramm zum Neustarten der Datenbank führt eine *Recovery nach Systemabsturz* durch, wenn die Datenbank abnormal beendet wurde, solange Anwendungen mit ihr verbunden waren; die Ursache für eine abnormale Beendigung kann z. B. ein Netzausfall oder ein Systemsoftwarefehler sein. Bei der Recovery werden festgeschriebene Transaktionen, die sich im Pufferpool befanden, jedoch zum Zeitpunkt des Fehlers nicht auf die Festplatte geschrieben waren, in der Datenbank nachvollzogen. Außerdem werden alle nicht festgeschriebenen Transaktionen, die eventuell auf Platte geschrieben wurden, wieder zurückgesetzt.

Wenn der Parameter *autorestart* nicht aktiviert ist, empfängt eine Anwendung den Fehler SQL1015N, wenn sie versucht, die Verbindung zu einer Datenbank herzustellen, für die eine Recovery nach einem Systemabsturz ausgeführt werden muss (für die die Datenbank neu gestartet werden muss). In diesem Fall kann die Anwendung das Dienstprogramm zum Neustarten der Datenbank aufrufen, oder Sie können die Datenbank neu starten, indem Sie die Funktion zum Neustarten im Recoveryprogramm auswählen.

avg_appls - Durchschnittliche Anzahl aktiver Anwendungen

Mithilfe dieses Parameters versucht das Abfrageoptimierungsprogramm zu ermitteln, wie viel Pufferpool zur Laufzeit für den ausgewählten Zugriffsplan verfügbar ist.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
Automatic [1 – maxappls]

Maßeinheit
Zähler

Empfehlung: Wenn DB2 in einer Mehrbenutzerumgebung ausgeführt wird, kann es vorteilhaft sein, wenn das Abfrageoptimierungsprogramm weiß, dass mehrere Abfragebenutzer das System verwenden (besonders bei komplexen Abfragen und einem großen Pufferpool), sodass das Optimierungsprogramm in seinen Annahmen zur Verfügbarkeit von Pufferpool weniger großzügig ist.

Bei der Einstellung dieses Parameters sollten Sie die Anzahl der Anwendungen mit komplexen Abfragen berechnen, die die Datenbank durchschnittlich verwenden. Aus dieser Berechnung sollten alle einfachen OLTP-Anwendungen (Online-Transaktionsprogramme) ausgeklammert werden. Wenn die Bestimmung dieser Anzahl schwierig ist, können Sie folgende Werte miteinander multiplizieren:

- Eine Durchschnittsanzahl aller Anwendungen, die für Ihre Datenbank ausgeführt werden. Der Datenbanksystemmonitor kann Informationen zur Anzahl der Anwendungen zu einem gegebenen Zeitpunkt liefern, sodass Sie anhand mehrerer Probewerte die Durchschnittsanzahl über einen gewissen Zeitraum hinweg berechnen können. Die Informationen des Datenbanksystemmonitors umfassen sowohl OLTP- als auch Nicht-OLTP-Anwendungen.
- Den von Ihnen geschätzten Prozentsatz an Anwendungen mit komplexen Abfragen.

Wie bei der Anpassung anderer Konfigurationsparameter, die das Optimierungsprogramm beeinflussen, sollten Sie auch den Wert dieses Parameters in kleinen Schritten ändern. Dadurch können Sie die Differenzen bei der Pfadauswahl minimieren.

Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl REBIND PACKAGE).

backup_pending - Backup anstehend

Dieser Parameter weist darauf hin, dass Sie ein Gesamtbackup der Datenbank ausführen müssen, bevor Sie auf sie zugreifen.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter ist nur aktiviert, wenn die Datenbankkonfiguration geändert wurde, sodass die Datenbank jetzt nicht mehr nicht wiederherstellbar sondern wiederherstellbar ist (das bedeutet, die Parameter *logretain* und *userexit* waren zunächst auf NO gesetzt, später wird jedoch einer der Parameter (oder beide) auf YES gesetzt und die Aktualisierung der Datenbankkonfiguration wird akzeptiert).

blk_log_dsk_ful - Bei voller Protokollplatte blockieren

Dieser Parameter kann so definiert werden, dass keine Fehler aufgrund erschöpfter Festplattenkapazität generiert werden, wenn DB2 keine neue Protokolldatei im aktiven Protokollpfad erstellen kann.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
No [Yes; No]

Anstatt einen Fehler aufgrund erschöpfter Festplattenkapazität zu generieren, versucht DB2 alle fünf Minuten, die Protokolldatei zu erstellen, bis diese Datei erfolgreich erstellt wurde. Nach jedem Versuch schreibt DB2 eine Nachricht in das Protokoll mit Benachrichtigungen für die Systemverwaltung. Das Überwachen dieses Protokolls mit Benachrichtigung für die Systemverwaltung stellt die einzige Möglichkeit dar, zu bestätigen, dass eine Anwendung blockiert ist. Bis zur erfolgreichen Erstellung der Protokolldatei kann keine Benutzeranwendung, die eine Aktualisierung von Tabellendaten ausführen will, eine Transaktion festschreiben. Auf Lesezugriff beschränkte Abfragen sind u. U. nicht direkt betroffen. Wenn jedoch eine Abfrage auf Daten zugreifen muss, die aufgrund einer Aktualisierungsanforderung gesperrt sind, oder auf eine Datenseite, die im Pufferpool von der aktualisierenden Anwendung korrigiert wird, erscheinen auch auf Lesezugriff beschränkte Abfragen blockiert.

Wenn *blk_log_dsk_ful* auf *yes* gesetzt wird, werden Anwendungen blockiert, sobald DB2 einen Fehler aufgrund erschöpfter Festplattenkapazität feststellt. Dadurch können Sie den Fehler beheben und die Transaktion anschließend beenden. Bei erschöpfter Festplattenkapazität können Sie beispielsweise alte Protokolldateien auf ein anderes Dateisystem versetzen oder das Dateisystem vergrößern, sodass blockierte Anwendungen beendet werden können.

Wenn *blk_log_dsk_ful* auf *no* gesetzt ist und eine Transaktion einen Fehler aufgrund erschöpfter Festplattenkapazität feststellt, schlägt die Transaktion fehl und wird mit ROLLBACK rückgängig gemacht. In einigen Situationen kann die Datenbank abstürzen, wenn eine Transaktion einen Fehler aufgrund erschöpfter Festplattenkapazität verursacht.

catalogcache_sz - Katalogcachegröße

Mit diesem Parameter wird der maximale Speicher in Seiten angegeben, die der Katalogcache vom Datenbankzwischenspeicher verwenden kann.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
-1 [MAXAPPLS*5]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank initialisiert wird

Freigabe
Wenn die Datenbank heruntergefahren wird

Dieser Parameter wird aus dem gemeinsam genutzten Datenbankspeicher zugeordnet und für das Caching (Zwischenspeichern) von Systemkataloginformationen verwendet. In einem partitionierten Datenbanksystem gibt es für jede Datenbankpartition einen Katalogcache.

Das Caching von Kataloginformationen in Einzeldatenbankpartitionen ermöglicht dem Datenbankmanager, den internen Systemaufwand zu verringern, da auf die Systemkataloge (oder auf den Katalogknoten in einer Umgebung mit partitionierten Datenbanken) zum Abrufen von Informationen, die bereits abgerufen wurden, nicht mehr zugegriffen werden muss. Das Verwenden des Katalogcache kann helfen, die Gesamtleistung folgender Operationen und Anwendungen zu erhöhen:

- Binden von Paketen und Kompilieren von SQL- und XQuery-Anweisungen
- Operationen, bei denen Zugriffsrechte auf Datenbankebene, Routinenzugriffsrechte, Zugriffsrechte für globale Variablen und Rollenberechtigungen überprüft werden
- Anwendungen, die mit Nicht-Katalogknoten in einer Umgebung mit partitionierten Datenbanken verbunden sind

Durch Definieren des Standardwerts (-1) in einer Serverumgebung oder in einer Umgebung mit partitionierten Datenbanken wird als Wert für die Berechnung der Seitenzuordnung das Fünffache des Werts für den Konfigurationsparameter *maxappls* genommen. Wenn das Fünffache von *maxappls* jedoch kleiner als 8 ist, gilt dies nicht. In diesem Fall setzt der Standardwert -1 den Parameter *catalogcache_sz* auf den Wert 8.

Empfehlung: Verwenden Sie zu Beginn den Standardwert, und optimieren Sie den Wert mithilfe des Datenbanksystemmonitors. Beim Optimieren dieses Parameters sollten Sie überlegen, ob der zusätzliche Speicher, der für den Katalogcache reserviert wird, möglicherweise besser für einen anderen Zweck zugeordnet werden sollte, z. B. für den Pufferpool oder den Paketcache.

Eine Optimierung dieses Parameters ist besonders wichtig, wenn die Auslastung für kurze Zeit viele SQL- oder XQuery-Kompilierungen umfasst und anschließend nur noch wenige oder gar keine Kompilierungen stattfinden. Wenn der Cache zu groß ist, kann Speicher für Kopien nicht mehr benötigter Informationen verschwendet werden.

Überlegen Sie, ob in einer Umgebung mit partitionierten Datenbanken für *catalogcache_sz* auf dem Katalogknoten ein größerer Wert festgelegt werden muss, da die auf Nicht-Katalogknoten benötigten Kataloginformationen immer zuerst auf dem Katalogknoten zwischengespeichert werden.

Mithilfe der Monitorelemente *cat_cache_lookups* (Suchfunktionen des Katalogcache), *cat_cache_inserts* (Einfügungen im Katalogcache), *cat_cache_overflows* (Überläufe des Katalogcache) und *cat_cache_size_top* (Obere Grenze des Katalogcache) können Sie ermitteln, ob dieser Konfigurationsparameter angepasst werden sollte.

Anmerkung: Der Katalogcache ist auf allen Knoten in einer Umgebung mit partitionierten Datenbanken vorhanden. Da für jeden Knoten eine lokale Datenbankkonfigurationsdatei vorhanden ist, definiert der Wert des Parameters *catalogcache_sz* für jeden Knoten die Größe des lokalen Katalogcache. Um ein effizientes Caching zu gewährleisten und Überlaufszenarios zu vermeiden, müssen Sie den Wert für *catalogcache_sz* auf jedem Knoten explizit festlegen und dabei die Möglichkeit in Betracht ziehen, den Wert für *catalogcache_sz* auf Nicht-Katalogknoten kleiner zu wählen als auf Katalogknoten. Bedenken Sie, dass die Informationen, die auf

Nicht-Katalogknoten zwischengespeichert werden müssen, aus dem Cache des Katalogknotens abgerufen werden. Ein Katalogcache auf Nicht-Katalogknoten ist wie eine Untermenge der Informationen des Katalogcache auf dem Katalogknoten.

Im Allgemeinen ist ein größerer Cachespeicher erforderlich, wenn eine UOW (Unit of Work, Arbeitseinheit) dynamische SQL- oder XQuery-Anweisungen enthält oder wenn Sie Pakete binden, die eine große Anzahl statischer SQL- oder XQuery-Anweisungen enthalten.

chngpgs_thresh - Schwellenwert für geänderte Seiten

Dieser Parameter gibt den Prozentsatz der geänderten Seiten an, bei dem die asynchronen Seitenlöschfunktionen gestartet werden, wenn sie zum gegebenen Zeitpunkt nicht aktiv sind.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
60 [5 – 99]

Maßeinheit
Prozent

Asynchrone Seitenlöschfunktionen schreiben geänderte Seiten aus dem Pufferpool (oder den Pufferpools) auf Platte, bevor der Bereich im Pufferpool von einem Datenbankagenten angefordert wird. Daher müssen Datenbankagenten in der Regel nicht die Auslagerung geänderter Seiten abwarten, bevor sie den Speicherbereich im Pufferpool nutzen können. Dadurch wird die Gesamtleistung der Datenbankanwendungen verbessert.

Wenn die Seitenlöschfunktionen gestartet werden, erstellen sie eine Liste der Seiten, die auf Platte zu schreiben sind. Wenn das Schreiben dieser Seiten auf Platte abgeschlossen ist, werden die Seitenlöschfunktionen wieder inaktiv und warten auf den nächsten Starttrigger.

Wenn die Registrierdatenbankvariable DB2_USE_ALTERNATE_PAGE_CLEANING definiert ist (d. h., die alternative Methode der Seitenbereinigung verwendet wird), hat der Parameter *chngpgs_thresh* keine Wirkung und der Datenbankmanager bestimmt automatisch, wie viele genutzte Seiten im Pufferpool zu behalten sind.

Empfehlung: Bei Datenbanken mit hohem Aufkommen an aktualisierenden Transaktionen können Sie allgemein sicherstellen, dass genügend freie Seiten im Pufferpool verfügbar sind, indem Sie diesen Parameter auf einen Wert setzen, der gleich groß wie oder kleiner als der Standardwert ist. Ein Prozentsatz über dem Standardwert kann sich positiv auf die Leistung auswirken, wenn in Ihrer Datenbank eine kleine Anzahl sehr großer Tabellen gespeichert ist.

codepage - Codepage für die Datenbank

Dieser Parameter zeigt die Codepage an, die zur Erstellung der Datenbank verwendet wurde. Der Wert des Parameters *codepage* wird auf der Basis des Werts für den Parameter *codeset* abgeleitet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

codeset - Codierter Zeichensatz für die Datenbank

Dieser Parameter zeigt den codierten Zeichensatz an, der zur Erstellung der Datenbank verwendet wurde. Der codierte Zeichensatz wird vom Datenbankmanager zur Bestimmung des Parameters *codepage* verwendet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

collate_info - Informationen zur Sortierfolge

Dieser Parameter legt die Sortierfolge der Datenbank fest. Bei einer sprachenbasierten Sortierfolge enthalten die ersten 256 Byte die Zeichenfolgedarstellung des Namens der Sortierfolge (z. B. ″SYSTEM_819_US″).

Dieser Parameter kann nur mit der API db2CfgGet angezeigt werden. Er kann **nicht** mithilfe des Befehlszeilenprozessors oder der Steuerzentrale angezeigt werden.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter enthält 260 Byte mit Informationen zur Sortierfolge der Datenbank. Die ersten 256 Byte geben die Sortierfolge der Datenbank an, wobei Byte „n″ die Sortierwertigkeit des Codepunkts enthält, dessen zugrundeliegende dezimale Darstellung „n″ in der Codepage der Datenbank ist.

Die letzten 4 Byte enthalten interne Informationen zur Art der Sortierfolge. Bei den letzten vier Byte des Parameters handelt es sich um eine ganze Zahl. Die ganze Zahl ist abhängig von der Endian-Folge der Plattform. Folgende Werte sind möglich:

- **0** – Die Sortierfolge enthält nicht eindeutige Wertigkeiten.
- **1** – Die Sortierfolge enthält ausschließlich eindeutige Wertigkeiten.
- **2** – Die Sortierfolge ist die Identitätssortierfolge, nach der Zeichenfolgen Byte für Byte verglichen werden.
- **3** – Die Sortierfolge lautet NLSCHAR und wird für das Sortieren von Zeichen in der Datenbank für Thailändisch TIS620-1 (Codepage 874) verwendet.
- **4** – Die Sortierfolge lautet IDENTITY_16BIT und implementiert den Algorithmus „CESU-8 Compatibility Encoding Scheme for UTF–16: 8–Bit″, wie er im Unicode Technical Report 26 angegeben wird, der auf der Website des Unicode Technical Consortium unter der Adresse http://www.unicode.org zur Verfügung steht.
- **X'8001'** – Die Sortierfolge lautet UCA400_NO und implementiert den UCA-Algorithmus (Unicode Collation Algorithm) auf der Grundlage von Unicode Standard Version 4.00 mit implizit aktivierter Normalisierung.
- **X'8002'** – Die Sortierfolge lautet UCA400_LTH und implementiert den UCA-Algorithmus (Unicode Collation Algorithm) auf der Grundlage von Unicode Standard Version 4.00. Sie sortiert alle Zeichen der thailändischen Sprache in der Reihenfolge des Royal Thai Dictionary.

- **X'8003'** – Die Sortierfolge lautet UCA400_LSK und implementiert den UCA-Algorithmus (Unicode Collation Algorithm) auf der Grundlage von Unicode Standard Version 4.00. Sie sortiert alle Zeichen der slowakischen Sprache in der richtigen Reihenfolge.

Anmerkung: Bei einer sprachenbasierten Sortierfolge enthalten die ersten 256 Byte die Zeichenfolgedarstellung des Namens der Sortierfolge.

Wenn Sie diese internen Informationen zur Art der Sortierfolge verwenden, müssen Sie eine Bytefolgeumkehrung in Betracht ziehen, wenn Informationen zu einer Datenbank auf einer anderen Plattform abgerufen werden.

Sie können die Sortierfolge bei der Erstellung der Datenbank angeben.

country/region - Gebietscode der Datenbank

Dieser Parameter zeigt den *Gebietscode* an, der beim Erstellen der Datenbank verwendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

database_consistent - Datenbank ist konsistent

Dieser Parameter gibt an, ob die Datenbank in einem konsistenten Zustand ist.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

YES gibt an, dass alle Transaktionen mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht wurden, sodass die Daten konsistent sind. Wenn es zu einem Systemabsturz kommt, während die Datenbank konsistent ist, sind keinerlei Maßnahmen erforderlich, um die Datenbank wieder verwendbar zu machen.

NO gibt an, dass noch eine Transaktion ansteht oder eine andere Funktion in der Datenbank noch nicht abgeschlossen wurde, sodass die Daten zu diesem Zeitpunkt nicht konsistent sind. Wenn es zu einem Systemabsturz kommt, während die Datenbank nicht konsistent ist, müssen Sie die Datenbank mit dem Befehl RESTART DATABASE erneut starten, um sie wieder verwendbar zu machen.

database_level - Release-Level der Datenbank

Dieser Parameter gibt den Release-Level des Datenbankmanagers an, der die Datenbank verwenden kann.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Im Fall einer nicht abgeschlossenen oder fehlgeschlagenen Migration (Umstellung) spiegelt dieser Parameter den Release-Level der nicht umgestellten Datenbank wider und kann daher vom Parameter *release* (Release-Level der Datenbank-

konfiguration) abweichen. Ansonsten ist der Wert des Parameters *database_level* mit dem Wert des Parameters *release* identisch.

database_memory - Größe des gemeinsam genutzten Datenbankspeichers

Dieser Parameter gibt die Menge des Speichers an, die für den gemeinsam genutzten Speicherbereich einer Datenbank reserviert ist. Ist diese Menge geringer als die aus den einzelnen Speicherparametern berechnete Menge (beispielsweise aus Sperrenlisten, Zwischenspeichern von Dienstprogrammen, Pufferpools usw.) wird die größere Menge verwendet.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Automatic [Computed, 0 - 4 294 967 295]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank aktiviert wird

Freigabe
Wenn die Datenbank inaktiviert wird

Wenn dieser Parameter auf AUTOMATIC gesetzt wird, aktiviert dies die automatische Optimierung. Wenn sie aktiviert ist, bestimmt die Speicheroptimierungsfunktion den Gesamtspeicherbedarf für die Datenbank und erhöht oder verringert auf der Basis des aktuellen Bedarfs der Datenbank die Speichergröße, die für den gemeinsam genutzten Datenbankspeicher zugeordnet ist. Wenn zum Beispiel der aktuelle Bedarf der Datenbank hoch ist und ausreichend freier Speicher auf dem System zur Verfügung steht, wird mehr Speicher für den gemeinsam genutzten Datenbankspeicher in Anspruch genommen. Wenn der Bedarf an Datenbankspeicher sinkt oder die Größe des freien Speichers auf dem System auf einen zu niedrigen Wert zurückgeht, wird ein Teil des gemeinsam genutzten Datenbankspeichers freigegeben.

Die Speicheroptimierungsfunktion lässt je nach dem berechneten Vorteil einer Bereitstellung zusätzlichen Speichers für die Instanz immer eine minimale Menge an Speicher frei. Wenn eine Bereitstellung von mehr Speicher für eine Instanz einen großen Vorteil erzielt, hält die Speicheroptimierungsfunktion eine kleinere Größe an Speicher frei. Wenn der Vorteil kleiner ist, wird mehr freier Speicher behalten. Dadurch können Datenbanken in der Verteilung von Systemspeicher kooperieren.

Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherkonsumenten verteilt, müssen mindestens zwei Speicherkonsumenten für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **self_tuning_mem** muss auf ON gesetzt sein).

Um die Verwaltung dieses Parameters zu vereinfachen, wird der Datenbankmanager durch die Einstellung COMPUTED angewiesen, die erforderliche Speicherkapazität zu berechnen und den Speicher bei der Aktivierung der Datenbank zuzuordnen. Der Datenbankmanager ordnet außerdem zusätzlichen Speicher zu, um dem Speicherbedarf in Zeiten hoher Auslastung für die Zwischenspeicher im gemeinsam genutzten Datenbankspeicherbereich gerecht zu werden, wenn ein Zwischenspeicher seine konfigurierte Größe überschreitet. Andere Operationen, wie zum Beispiel dynamische Aktualisierungen der Konfiguration, haben ebenfalls Zugriff auf diesen zusätzlichen Speicher. Der Befehl db2pd mit der Option **-memsets** kann zur Überwachung des im gemeinsam genutzten Datenbankspeicherbereich verbliebenen nicht genutzten Speichers verwendet werden.

Empfehlung: Dieser Wert bleibt in der Regel bei der Einstellung AUTOMATIC. Für Umgebungen, die die Einstellung AUTOMATIC nicht unterstützten, sollte die Einstellung COMPUTED verwendet werden. Zum Beispiel kann der zusätzliche Speicher zur Erstellung neuer Pufferpools oder zur Vergrößerung vorhandener Pufferpools verwendet werden.

Anmerkung: Wenn Sie in Version 9.5 den Konfigurationsparameter **database_memory** auf AUTOMATIC setzen, ist die anfängliche gemeinsam genutzte Datenbankspeicherzuordnung die konfigurierte Größe aller für die Datenbank definierten Zwischenspeicher und Pufferpools, und der Speicher wird nach Bedarf vergrößert. Wenn der Parameter **database_memory** auf einen bestimmten Wert gesetzt wurde, wird die erforderliche Speicherkapazität von Beginn an während der Aktivierung der Datenbank zugeordnet. Wenn die anfängliche Speicherkapazität vom Betriebssystem nicht zugeordnet werden kann oder den Grenzwert des Parameters **instance_memory** überschreitet, schlägt die Aktivierung der Datenbank mit dem folgenden SQL-Fehler fehl: SQL1084C Gemeinsam genutzte Speichersegmente können nicht zugeordnet werden.

Ab DB2 Version 9.5 Fixpack 2 verwendet der Datenbankmanager umlagerbaren Speicher für den gemeinsam genutzten Datenbankspeicher, wenn in der Solaris-Betriebsumgebung für **database_memory** die Einstellung AUTOMATIC definiert wird. Durch diese Verwendung kleinerer Speicherseiten kann es zu Leistungseinbußen kommen. Bei Solaris-Betriebssystemen unter UltraSPARC versucht der Datenbankmanager, 64-KB-Speicherseiten zu verwenden, falls diese verfügbar sind. Sind keine 64-KB-Speicherseiten verfügbar, verwendet der Datenbankmanager 8-KB-Speicherseiten. Bei Solaris-Betriebssystemen auf Sun-x64-Sytemen verwendet der Datenbankmanager 4-KB-Speicherseiten. Wenn die Verwendung großer Speicherseiten im gemeinsam genutzten Speicher unter Solaris beibehalten werden soll, definieren Sie für **database_memory** die Einstellung COMPUTED oder einen numerischen Wert.

DB2-Speicherbelegung steuern:

Wenn der Parameter **instance_memory** auf AUTOMATIC gesetzt wurde, wird beim Starten der Instanz (db2start) eine feste Obergrenze für die gesamte Speicherbelegung durch die Instanz festgelegt. Die tatsächliche Speicherbelegung durch den Datenbankmanager variiert je nach Auslastung. Wenn der Speichermanager für eine automatische Leistungsoptimierung für die Optimierung des Datenbankspeichers (Parameter **database_memory**) aktiviert wurde (standardmäßig für neue Datenbanken der

Fall), aktualisiert der Speichermanager für eine automatische Leistungsoptimierung während der Laufzeit dynamisch die Größe der für die Leistung kritischen Zwischenspeicher innerhalb der gemeinsam genutzten Speicher der Datenbank. Dies geschieht in Abhängigkeit von dem freien physischen Hauptspeicher im System, während gleichzeitig sichergestellt wird, dass ausreichend freier Instanzspeicher (Parameter **instance_memory**) für funktionalen Speicherbedarf zur Verfügung steht. Weitere Informationen finden Sie im Abschnitt zum Konfigurationsparameter **instance_memory**.

Einschränkungen in einigen Linux[1]-Kernels:
Aufgrund von Betriebssystemeinschränkungen in einigen Linux-Kerneln lässt der Speichermanager für eine automatische Leistungsoptimierung derzeit die Einstellung AUTOMATIC für den Parameter **database_memory** nicht zu. Jedoch ist diese Einstellung jetzt für diese Kernel unter der Voraussetzung zulässig, dass **instance_memory** auf einen bestimmten Wert gesetzt wird, nicht jedoch auf AUTOMATIC. Wenn **database_memory** auf AUTOMATIC gesetzt wurde und **instance_memory** später zurück auf AUTOMATIC gesetzt wird, wird der Konfigurationsparameter **database_memory** bei der nächsten Aktivierung der Datenbank automatisch mit der Einstellung COMPUTED aktualisiert. Wenn einige Datenbanken bereits aktiv sind, stoppt der Speichermanager für die automatische Leistungsoptimierung die Optimierung der gesamten Datenbankspeicherkapazitäten (Parameter **database_memory**).

[1]Unter Linux unterstützt dieser Parameter die Einstellung AUTOMATIC für RHEL5 und SUSE 10 SP1 und höher. Alle anderen validierten Linux-Varianten kehren zu der Einstellung COMPUTED zurück, wenn der Kernel diese Funktion nicht unterstützt.

db_mem_thresh - Schwellenwert für Datenbankspeicher

Dieser Parameter gibt den maximalen Prozentsatz des festgeschriebenen, jedoch momentan ungenutzten gemeinsamen Datenbankspeichers an, der vom Datenbankmanager zugelassen wird, bevor damit begonnen wird, festgeschriebene Speicherseiten freizugeben und an das Betriebssystem zurückzugeben.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [0–100]

Maßeinheit
Prozent

Dieser Datenbankkonfigurationsparameter bezieht sich darauf, wie der Datenbankmanager überschüssigen gemeinsam genutzten Speicher der Datenbank behandelt. Wenn Seiten von Speicher durch einen Prozess beansprucht werden, werden sie in der Regel für den Prozess fest reserviert. Das heißt, eine Speicherseite wird vom Betriebssystem zugeordnet und belegt einen Bereich entweder im physischen Speicher oder in einer Auslagerungsdatei auf der Platte. Abhängig von der Auslastung der Datenbank kann es zu gewissen Tageszeiten zu Spitzenbelastungen der Daten-

bank mit gemeinsam genutztem Speicher kommen. Wenn das Betriebssystem einmal genügend Speicher für diese Spitzenlastzeiten reserviert hat, bleibt dieser Speicher reserviert, auch wenn der Speicherbedarf später wieder nachlässt.

Gültige Werte sind ganze Zahlen aus dem Bereich von 0 (sofortige Freigabe jedes nicht genutzten gemeinsamen Datenbankspeichers) bis 100 (keine Freigabe von ungenutztem gemeinsamen Datenbankspeicher). Der Standardwert ist 10 (Freigabe von ungenutztem Speicher erst, wenn mehr als 10 % des gemeinsam genutzten Datenbankspeichers zurzeit nicht genutzt werden); dieser Wert sollte für die meisten Auslastungen geeignet sein.

Dieser Konfigurationsparameter kann dynamisch aktualisiert werden. Dieser Parameter muss mit Vorsicht aktualisiert werden, da ein zu niedrig eingestellter Wert zur einer übermäßigen Speicherbelastung für das System führen kann (da ständig Speicherseiten reserviert und wieder freigegeben werden), während ein zu hoch eingestellter Wert möglicherweise verhindert, dass der Datenbankmanager überhaupt gemeinsam genutzten Datenbankspeicher an das Betriebssystem zur Verwendung durch andere Prozesse zurückgibt.

Dieser Konfigurationsparameter wird ignoriert (d. .h. ungenutzte Seiten des gemeinsam genutzten Datenbankspeichers bleiben reserviert), wenn der Speicherbereich des gemeinsam genutzten Datenbankspeichers durch die Registrierdatenbankvariable DB2_PINNED_BP fixiert ist, für große Seiten durch die Registrierdatenbankvariable DB2_LARGE_PAGE_MEM konfiguriert ist oder die Freigabe von Speicher durch die Registrierdatenbankvariable DB2MEMDISCLAIM explizit inaktiviert ist.

Einige Versionen von Linux unterstützen die Freigabe von Unterbereichen eines gemeinsam genutzten Speichersegments an das Betriebssystem nicht. Auf solchen Plattformen wird dieser Parameter ignoriert.

dbheap - Zwischenspeicher für Datenbank

Dieser Parameter definiert die maximale Speicherkapazität, die vom Zwischenspeicher für die Datenbank verwendet werden kann.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass der Datenbankzwischenspeicher nach Bedarf erhöht wird, bis entweder der Grenzwert des Parameters *database_memory* oder des Parameters *instance_memory* erreicht ist.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Automatic [32 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank aktiviert wird

Freigabe

Wenn die Datenbank inaktiviert wird

Für jede Datenbank gibt es einen Datenbankzwischenspeicher, der vom Datenbankmanager für alle Anwendungen, die auf die Datenbank zugreifen, verwendet wird. Er enthält Steuerblockdaten für Tabellen, Indizes, Tabellenbereiche und Pufferpools. Er enthält außerdem Speicherbereich für den Protokollpuffer (*logbufsz*) sowie temporären Speicher, der von Dienstprogrammen verwendet wird. Daher ist die Größe des Zwischenspeichers von vielen Variablen abhängig. Die Steuerblockdaten werden im Zwischenspeicher gehalten, bis alle Anwendungen die Verbindung zur Datenbank getrennt haben.

Der Mindestspeicherbereich, den der Datenbankmanager zu Beginn benötigt, wird beim Herstellen der ersten Verbindung zugeordnet. Der Datenbereich wird nach Bedarf erweitert, bis entweder die konfigurierte Obergrenze erreicht ist oder, wenn die Einstellung AUTOMATIC gewählt wurde, bis der Speicher für den Parameter *database_memory* oder *instance_memory* oder für beide ausgeschöpft ist.

Als grobe Orientierung können Sie die folgende Formel verwenden, wenn Sie dem Konfigurationsparameter *dbheap* einen Wert zuordnen möchten:

```
10 KB pro Tabellenbereich + 4 KB pro Tabelle + (1 KB + 4*verwendete Speicherbereiche)
  pro Bereichsclustertabelle (RCT - Range-Clustered Table)
```

Der von Ihnen konfigurierte Wert *dbheap* steht nur für einen Teil des zugeordneten Datenbankzwischenspeicher. Der Datenbankzwischenspeicher ist der Hauptspeicherbereich, über den der globale Datenbankspeicherbedarf befriedigt wird. Die Größe ist so berechnet, dass zusätzlich zum Wert *dbheap* auch grundlegende Zuordnungen für den Start der Datenbank enthalten sind. Tools mit Speicherbelegung wie der Memory Tracker, Snapshot Monitor und db2pd berichten über die statistischen Daten des größeren Datenbankzwischenspeichers. Es findet keine separate Verfolgung der Zuordnungen statt, die durch den Konfigurationsparameter *dbheap* repräsentiert werden. Daher ist es normal, dass die von den Tools dokumentierten statistischen Daten zur Speicherbelegung durch den Datenbankzwischenspeicher den für den Parameter *dbheap* konfigurierten Wert überschreiten.

Mithilfe des Datenbanksystemmonitors können Sie unter Verwendung des Elements *db_heap_top* (Maximal zugeordneter Datenbankzwischenspeicher) die größte Speichermenge ermitteln, die für den Datenbankzwischenspeicher verwendet wurde.

Anmerkung:

- Arbeitsklassensets und Arbeitsaktionssets von Workload Management (WLM) werden im Datenbankzwischenspeicher gespeichert. Jedoch wird dafür nur ein sehr kleiner Teil des Speichers benötigt.
- Informationen zu gesicherten Kontexten, zum Workload-Management sowie zu Prüfrichtlinien werden zwecks schneller Verarbeitung im Hauptspeicher zwischengespeichert. Dieser Speicher wird aus dem Zwischenspeicher für die Datenbank zugeordnet. Daher stellen benutzerdefinierte Objekte für gesicherte Kontexte, für das Workload-Management und für Prüfrichtlinien größere Speicheranforderungen an den Datenbankzwischenspeicher. In diesem Fall wird empfohlen, den Konfigurationsparameter *dbheap* auf den Wert AUTOMATIC zu setzen, sodass der Datenbankmanager die Größe des Zwischenspeichers für die Datenbank automatisch verwalten kann.

decflt_rounding - Rundungsmodus für dezimale Gleitkommawerte (Konfigurationsparameter)

Mit diesem Parameter können Sie den Rundungsmodus für dezimale Gleitkommawerte (DECFLOAT) angeben. Der Rundungsmodus betrifft Operationen mit dezimalen Gleitkommawerten auf dem Server sowie den Ladevorgang (LOAD).

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Siehe „Auswirkungen einer Änderung von decflt_rounding" auf Seite 613 unten.

Standardwert [Bereich]
ROUND_HALF_EVEN [ROUND_CEILING, ROUND_FLOOR, ROUND-_HALF_UP, ROUND_DOWN]

DB2 unterstützt fünf IEEE-konforme Rundungsmodi für dezimale Gleitkommawerte. Der Rundungsmodus gibt an, wie das Ergebnis einer Berechnung zu runden ist, wenn das Ergebnis die zulässige Anzahl Stellen überschreitet. Die Rundungsmodi sind wie folgt definiert:

ROUND_CEILING
Aufrunden (gegen positiv Unendlich runden). Wenn alle verworfenen Stellen null sind oder wenn das Vorzeichen negativ ist, bleibt das Ergebnis unverändert. Andernfalls wird der Ergebniskoeffizient um 1 erhöht (aufgerundet).

ROUND_FLOOR
Abrunden (gegen negativ Unendlich runden). Wenn alle verworfenen Stellen null sind oder wenn das Vorzeichen positiv ist, bleibt das Ergebnis unverändert. Andernfalls bleibt das Vorzeichen negativ und der Ergebniskoeffizient wird um 1 erhöht.

ROUND_HALF_UP
Zum näherliegenden Wert der höheren Dezimalstelle auf- oder abrunden. Bei gleichem Abstand, um 1 aufrunden. Wenn die verworfenen Stellen einen Wert größer-gleich der Hälfte des Werts 1 (0,5) an der nächsthöheren Dezimalstelle darstellen, wird der Ergebniskoeffizient um 1 erhöht (aufrunden). Andernfalls werden die verworfenen Stellen (0,5 oder weniger) ignoriert.

ROUND_HALF_EVEN
Zum näherliegenden Wert der nächsthöheren Dezimalstelle auf- oder abrunden. Bei gleichem Abstand so runden, dass die letzte Ziffer gerade ist. Wenn die verworfenen Stellen einen Wert größer als die Hälfte des Werts 1 (0,5) an der nächsthöheren Dezimalstelle darstellen, wird der Ergebniskoeffizient um 1 erhöht (aufgerundet). Wenn sie einen Wert kleiner als 0,5 darstellen, wird der Ergebniskoeffizient nicht angepasst, d. h., die verworfenen Stellen werden ignoriert. Wenn sie genau die Hälfte (0,5) darstellen, bleibt der Ergebniskoeffizient unverändert, wenn die letzte rechte Ziffer gerade ist. Sie wird um 1 erhöht (aufgerundet), wenn die letzte rechte Ziffer ungerade ist, damit eine gerade Ziffer entsteht. Dieser Rundungsmodus ist laut Spezifikation für IEEE-konforme dezimale Gleitkommawerte der Standardrundungsmodus und wird auch in DB2-Produkten als Standardrundungsmodus verwendet.

ROUND_DOWN
Abrunden (Abschneiden). Die verworfenen Stellen werden ignoriert.

Tabelle 72 zeigt die Ergebnisse des Auf-/Abrundens der Werte 12,341, 12,345, 12,349, 12,355 und -12,345 auf jeweils vier Stellen für die verschiedenen Rundungsmodi:

Tabelle 72. Rundungsmodi für dezimale Gleitkommawerte

Rundungsmodus	**12,341**	**12,345**	**12,349**	**12,355**	**-12,345**
ROUND_DOWN	12,34	12,34	12,34	12,35	-12,34
ROUND_HALF_UP	12,34	12,35	12,35	12,36	-12,35
ROUND_HALF_EVEN	12,34	12,34	12,35	12,36	-12,34
ROUND_FLOOR	12,34	12,34	12,34	12,35	-12,35
ROUND_CEILING	12,35	12,35	12,35	12,36	-12,34

Auswirkungen einer Änderung von decflt_rounding

- MQTs, die früher erstellt wurden, könnten Ergebnisse enthalten, die sich von den Ergebnissen unterscheiden, die mit dem neuen Rundungsmodus erstellt werden. Um dieses Problem zu lösen, sollten Sie potenziell betroffene MQTs aktualisieren.
- Die Ergebnisse eines Triggers können durch den neuen Rundungsmodus beeinträchtigt werden. Eine Änderung hat keinerlei Auswirkungen auf die bereits geschriebenen Daten.
- Integritätsbedingungen, die das Einfügen von Daten in eine Tabelle ermöglichten, weisen (im Falle einer erneuten Auswertung) genau diese Daten möglicherweise zurück. Im Gegensatz dazu akzeptieren Integritätsbedingungen, die das Einfügen von Daten in eine Tabelle nicht ermöglichten, (im Falle einer erneuten Auswertung) genau diese Daten. Verwenden Sie die Anweisung SET INTEGRITY, um nach solchen Problemen zu suchen und sie zu beheben. Der Wert einer generierten Spalte, deren Berechnung von *decflt_rounding* abhängig ist, kann für zwei Zeilen, die abgesehen vom generierten Spaltenwert identisch sind, unterschiedlich sein, wenn die eine Zeile vor der Änderung an *decflt_rounding* und die andere nach der Änderung an *decflt_rounding* eingefügt wurde.
- Der Rundungsmodus wird nicht in Abschnitte hinein kompiliert. Aus diesem Grund muss statisches SQL nach der Änderung von *decflt_rounding* nicht erneut kompiliert werden.

Anmerkung: Der Wert dieses Konfigurationsparameters wird nicht dynamisch geändert, sondern wird erst wirksam, wenn alle Anwendungen ihre Verbindung zur Datenbank getrennt haben. Wenn die Datenbank aktiviert ist, muss sie inaktiviert werden.

dft_degree - Grad der Parallelität

Mit diesem Parameter wird der Standardwert für das Sonderregister CURRENT DEGREE und die Bindeoption DEGREE angegeben.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Verbindung

Standardwert [Bereich]
1 [-1(ANY), 1 - 32 767]

Der Standardwert ist 1.

Bei Angabe des Werts 1 wird keine partitionsinterne Parallelität verwendet. Der Wert -1 (oder ANY) bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität je nach Anzahl der Prozessoren und Art der Abfrage festlegt.

Der Grad der partitionsinternen Parallelität für eine SQL-Anweisung wird während der Kompilierung der Anweisung mithilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption DEGREE angegeben. Der maximale Grad der partitionsinternen Parallelität zur Laufzeit für eine aktive Anwendung wird mit dem Befehl SET RUNTIME DEGREE angegeben. Der Konfigurationsparameter Maximaler Grad der Parallelität bei Abfragen (*max_querydegree*) gibt den maximalen Grad der partitionsinternen Parallelität für alle SQL-Abfragen an.

Der zur Laufzeit tatsächlich verwendete Parallelitätsgrad ist der niedrigste der folgenden Werte:

- Konfigurationsparameter *max_querydegree*
- Grad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen

Mit diesem Parameter wird der Standardwert für EXTENTSIZE bei Tabellenbereichen festgelegt.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
32 [2 – 256]

Maßeinheit
Seiten

Bei der Erstellung eines Tabellenbereichs kann wahlfrei `EXTENTSIZE n` angegeben werden, wobei n die Speicherbereichsgröße ist. Wenn Sie EXTENTSIZE in der Anweisung CREATE TABLESPACE nicht angeben, verwendet der Datenbankmanager den Wert, der durch diesen Parameter definiert wird.

Empfehlung: In vielen Fällen geben Sie die Speicherbereichsgröße bei der Erstellung eines Tabellenbereichs explizit an. Bevor Sie einen Wert für diesen Parameter wählen, sollten Sie verstehen, wie die Speicherbereichsgröße in der Anweisung CREATE TABLESPACE explizit gewählt wird.

dft_loadrec_ses - Standardanzahl von Sitzungen für Recovery

Mit diesem Parameter wird die Standardanzahl von Sitzungen angegeben, die während des Wiederabrufens einer Tabellenladekopie verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
1 [1 - 30 000]

Maßeinheit
Zähler

Der Wert sollte auf eine optimale Anzahl von E/A-Sitzungen gesetzt werden, die zum Wiederabrufen einer Ladekopie verwendet werden sollen. Das Wiederabrufen einer Ladekopie ist eine ähnliche Operation wie ein Restore. Sie können diesen Parameter durch Einträge überschreiben, die sich in der Datei mit den Angaben zur Speicherposition der exportierten Daten befinden. Diese Datei wird von der Umgebungsvariablen DB2LOADREC angegeben.

Die Standardanzahl der Puffer, die für den Abruf der Ladekopien verwendet werden, ist um zwei größer als der Wert dieses Parameters. Die Anzahl der Puffer kann ebenfalls in der Datei mit den Angaben zur Speicherposition der exportierten Daten überschrieben werden.

Dieser Parameter ist nur gültig, wenn die aktualisierende Recovery aktiviert ist.

dft_mttb_types - Standardtypen von verwalteten Tabellen zur Optimierung

Mit diesem Parameter wird der Standardwert für das Sonderregister CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION angegeben. Der Wert dieses Registers bestimmt, welche Typen mit REFRESH DEFERRED definierter MQTs der Abfrageoptimierung verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
SYSTEM [ALL, NONE, FEDERATED_TOOL, SYSTEM, USER oder eine Liste von Werten]

Sie können eine Liste von Werten durch Kommata getrennt angeben. Zum Beispiel: 'USER,FEDERATED_TOOL'. ALL und NONE können nicht zusammen mit anderen Werten angegeben werden, und Sie können denselben Wert nur einmal angeben. Zur Verwendung in Verbindung mit den APIs db2CfgSet und db2CfgGet lauten die akzeptierbaren Parameterwerte wie folgt: 8 (ALL), 4 (NONE), 16 (FEDERATED_TOOL), 1 (SYSTEM) und 2 (USER). Es können mehrere Werte zusammen angegeben werden; dazu müssen Sie das bitweise ODER verwenden; Beispiel: 18 wäre das Äquivalent zu USER,FEDERATED_TOOL. Wie zuvor dürfen die Werte 4 und 8 nicht mit anderen Werten verwendet werden.

dft_prefetch_sz - Standardwert für PREFETCHSIZE

Mit diesem Parameter wird der Standardwert für PREFETCHSIZE bei Tabellenbereichen festgelegt.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Automatic [0 - 32 767]

Maßeinheit
Seiten

Bei der Erstellung eines Tabellenbereichs kann optional PREFETCHSIZE *n* angegeben werden, wobei *n* die Anzahl der Seiten ist, die der Datenbankmanager liest, wenn ein Vorablesezugriff erfolgt. Wenn Sie in der Anweisung CREATE TABLESPACE keinen Wert für PREFETCHSIZE angeben, verwendet der Datenbankmanager den aktuellen Wert des Parameters *dft_prefetch_sz*.

Wenn ein Tabellenbereich mit der Angabe AUTOMATIC DFT_PREFETCH_SZ erstellt wird, wird die Größe des Vorablesezugriffs des Tabellenbereichs automatisch ermittelt. Dies bedeutet, dass DB2 den Wert für PREFETCHSIZE des Tabellenbereichs unter Verwendung der folgenden Gleichung entsprechend automatisch berechnet und aktualisiert:

```
PREFETCHSIZE = (Anzahl Container) * (Anzahl physischer Spindeln) * EXTENTSIZE
```

Dabei gilt für die Anzahl der physischen Spindeln der Standardwert 1. Diese Anzahl kann über die DB2-Registrierdatenbankvariable DB2_PARALLEL_IO definiert werden. Diese Berechnung erfolgt zu folgenden Zeitpunkten:

- Beim Starten der Datenbank
- Bei der anfänglichen Erstellung eines Tabellenbereichs mit AUTOMATIC DFT_PREFETCH_SZ
- Bei einer Änderung der Anzahl von Containern für einen Tabellenbereich durch die Ausführung einer Anweisung ALTER TABLESPACE
- Bei einer Änderung des Werts für PREFETCHSIZE für einen Tabellenbereich in AUTOMATIC durch eine Anweisung ALTER TABLESPACE

Der Status AUTOMATIC für PREFETCHSIZE kann aktiviert oder inaktiviert werden, wenn der PREFETCHSIZE-Wert manuell durch eine Anweisung ALTER TABLESPACE aktualisiert wird.

Empfehlung: Mit Tools zur Systemüberwachung können Sie feststellen, ob Ihre CPU ausgelastet ist, während das System auf Ein-/Ausgaben wartet. Die Erhöhung dieses Parameterwerts kann nützlich sein, wenn für die Tabellenbereiche, die verwendet werden, kein Wert für PREFETCHSIZE definiert ist.

Dieser Parameter enthält den Standardwert für die gesamte Datenbank und ist eventuell nicht für alle Tabellenbereiche innerhalb der Datenbank geeignet. Beispielsweise kann der Wert 32 für einen Tabellenbereich mit dem Wert 32 Seiten für

EXTENTSIZE geeignet sein, aber nicht für einen Tabellenbereich mit dem Wert 25 Seiten für EXTENTSIZE. Im Idealfall sollten Sie den Wert für PREFETCHSIZE für jeden Tabellenbereich explizit angegeben.

Sie sollten als Wert für diesen Parameter einen Faktor oder ein ganzzahliges Vielfaches des Wertes des Parameters *dft_extent_sz* angeben, um die E/A-Operationen für Tabellenbereiche zu minimieren, die mit dem Standardwert für EXTENTSIZE (Parameter *dft_extent_sz*) definiert sind. Wenn z. B. für den Parameter *dft_extent_sz* der Wert 32 angegeben ist, könnten Sie den Wert von *dft_prefetch_sz* auf 16 (einen Bruchteil von 32) oder auf 64 (ein ganzzahliges Vielfaches von 32) setzen. Wenn für PREFETCHSIZE ein Vielfaches des Werts für EXTENTSIZE angegeben ist, kann der Datenbankmanager parallele E/A-Operationen ausführen, wenn die folgenden Bedingungen erfüllt sind:

- Die Bereiche, die vorab gelesen werden, befinden sich auf verschiedenen physischen Einheiten.
- Es sind mehrere E/A-Server konfiguriert (*num_ioservers*).

dft_queryopt - Standardabfrageoptimierungsklasse

Mit der Abfrageoptimierungsklasse können Sie das Optimierungsprogramm anweisen, beim Kompilieren von SQL- und XQuery-Abfragen verschiedene Grade der Optimierung zu verwenden. Dieser Parameter bietet zusätzliche Flexibilität, indem die Standardabfrageoptimierungsklasse für den Fall festgelegt wird, dass weder die Anweisung SET CURRENT QUERY OPTIMIZATION noch die Option QUERYOPT mit dem Bindebefehl verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Verbindung

Standardwert [Bereich]
5 [0 — 9]

Maßeinheit
Abfrageoptimierungsklasse (siehe unten)

Derzeit sind folgende Abfrageoptimierungsklassen definiert:

- 0 - Minimale Abfrageoptimierung
- 1 - Abfrageoptimierung in etwa vergleichbar mit DB2 Version 1
- 2 - Geringe Optimierung
- 3 - Mittlere Abfrageoptimierung
- 5 - Starke Abfrageoptimierung, die über heuristische Methoden den Aufwand bei der Auswahl eines Zugriffsplans reduziert. Dies ist der Standardwert.
- 7 - Starke Abfrageoptimierung
- 9 - Maximale Abfrageoptimierung

dft_refresh_age - Standardaktualisierungsalter

Dieser Parameter stellt die maximale Dauer seit der Verarbeitung einer Anweisung REFRESH TABLE für eine bestimmte MQT (Materialized Query Table, gespeicherte Abfragetabelle) dar. Nach der Überschreitung dieses Zeitlimits wird die MQT erst zur Erfüllung von Abfragen verwendet, wenn die MQT aktualisiert ist.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0, 99999999999999 (ANY)]

Für diesen Parameter wird der für REFRESH AGE verwendete Standardwert eingesetzt, wenn das Sonderregister CURRENT REFRESH AGE nicht angegeben wird. Dieser Parameter gibt einen Zeitmarkendifferenzwert mit dem Datentyp DECIMAL(20,6) an. Wenn für CURRENT REFRESH AGE der Wert 99999999999999 (ANY) gilt und die Abfrageoptimierungsklasse (QUERY OPTIMIZATION) den Wert zwei (oder fünf und höher) hat, werden MQTs der Art REFRESH DEFERRED zum Optimieren der Verarbeitung einer dynamischen Abfrage herangezogen.

dft_sqlmathwarn - Bei arithmetischen Ausnahmebedingungen fortsetzen

Mit diesem Parameter wird der Standardwert festgelegt, der bestimmt, ob arithmetische Fehler und Fehler bei Abfrageumwandlungen beim Kompilieren von SQL-Anweisungen als Fehler oder als Warnungen behandelt werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [No, Yes]

Bei statischen SQL-Anweisungen wird der Wert dieses Parameters dem Paket während des Bindens zugeordnet. Bei dynamischen SQL-DML-Anweisungen wird der Wert dieses Parameters verwendet, wenn die Anweisung vorbereitet wird (PREPARE).

Achtung: Wenn Sie den Wert des Parameters *dft_sqlmathwarn* für eine Datenbank ändern, kann sich das Verhalten von Prüfungen auf Integritätsbedingung, Triggern und Sichten, die arithmetische Ausdrücke enthalten, ändern. Dies wiederum kann sich auf die Datenintegrität der Datenbank auswirken. Sie sollten die Einstellung des Parameters *dft_sqlmathwarn* für eine Datenbank nur nach einer gründlichen Analyse der Auswirkungen der neuen Behandlung arithmetischer Ausnahmebedingungen auf Integritätsprüfungen, Trigger und Sichten ändern. Auch alle weiteren Änderungen sind mit derselben Sorgfalt vorzunehmen.

Betrachten Sie beispielsweise die folgende Prüfung auf Integritätsbedingung, die eine arithmetische Divisionsoperation enthält:

```
A/B > 0
```

Wenn *dft_sqlmathwarn* gleich „No" ist und eine Einfügeoperation (INSERT) mit B=0 versucht wird, wird die Division durch 0 als arithmetischer Fehler verarbeitet. Die Einfügeoperation schlägt fehl, weil DB2 die Integritätsbedingung nicht prüfen kann. Wenn der Parameter *dft_sqlmathwarn* auf „Yes" gesetzt wird, wird die Division durch 0 als arithmetische Warnung mit dem Ergebnis NULL verarbeitet. Das Ergebnis NULL führt dazu, dass das Vergleichselement mit dem Wert UNKNOWN ausgewertet wird und die Einfügeoperation erfolgreich ist. Wenn *dft_sqlmathwarn*

wieder auf „No" gesetzt wird, schlägt der Versuch, dieselbe Zeile einzufügen, fehl, weil der Fehler der Division durch 0 DB2 daran hindert, die Integritätsbedingung auszuwerten. Die Zeile, die mit B=0 eingefügt wurde, als der Parameter *dft_sqlmathwarn* den Wert „Yes" hatte, bleibt in der Tabelle und kann ausgewählt werden. Aktualisierungen der Zeile, für die die Integritätsbedingung ausgewertet wird, schlagen fehl, während Aktualisierungen, die keine erneute Prüfung der Integritätsbedingung erfordern, erfolgreich ausgeführt werden.

Bevor Sie den Parameter *dft_sqlmathwarn* von „No" in „Yes" ändern, sollten Sie in Betracht ziehen, die Integritätsbedingung so umzuschreiben, dass sie Nullwerte aus arithmetischen Ausdrücken gesondert behandelt. Beispiel:

```
( A/B > 0 ) AND ( CASE
                    WHEN A IS NULL THEN 1
                    WHEN B IS NULL THEN 1
                    WHEN A/B IS NULL THEN 0
                    ELSE 1
                    END
                = 1 )
```

Diese Bedingung kann verwendet werden, wenn A und B Nullwerte haben können, d. h. die Dateneingabe optional ist. Wenn A oder B keinen Nullwert haben kann, kann die entsprechende Klausel WHEN...IS NULL entfernt werden.

Bevor Sie den Parameter *dft_sqlmathwarn* von „Yes" in „No" ändern, sollten Sie zunächst prüfen, welche Daten inkonsistent werden könnten, indem Sie zum Beispiel Vergleichselemente wie die folgenden verwenden:

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

Wenn inkonsistente Zeilen isoliert sind, können Sie geeignete Maßnahmen zur Korrektur der Inkonsistenz ergreifen, bevor Sie den Parameter *dft_sqlmathwarn* ändern. Sie können Integritätsbedingungen mit arithmetischen Ausdrücken nach der Änderung auch manuell gegenprüfen. Dazu setzen Sie die betroffenen Tabellen in den Status *Überprüfung anstehend* (mit der Klausel OFF der Anweisung SET CONSTRAINTS) und fordern anschließend eine Prüfung an (mit der Klausel IMMEDIATE CHECKED der Anweisung SET CONSTRAINTS). Inkonsistente Daten werden durch einen arithmetischen Fehler angezeigt, der verhindert, dass die Integritätsbedingung ausgewertet wird.

Empfehlung: Verwenden Sie die Standardeinstellung "No", sofern Sie keine Abfragen benötigen, deren Verarbeitung arithmetische Ausnahmebedingungen einschließt. In diesem Fall geben Sie den Wert "Yes" an. Dieser Fall kann eintreten, wenn Sie SQL-Anweisungen verarbeiten, die auf anderen Datenbankmanagern unabhängig von möglichen arithmetischen Ausnahmebedingungen Ergebnisse liefern.

discover_db - Discovery-Unterstützung für diese Datenbank

Mit diesem Parameter kann verhindert werden, dass Informationen zu einer Datenbank an einen Client zurückgegeben werden, wenn eine Discovery-Anforderung auf dem Server empfangen wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Enable [Disable, Enable]

Standardmäßig ist dieser Parameter so eingestellt, dass Discovery-Unterstützung für diese Datenbank aktiviert ist.

Durch Ändern dieses Parameterwerts in „Disable" können Datenbanken, die sensible Daten enthalten, vor dem Discovery-Prozess verdeckt werden. Diese Maßnahme kann zusätzlich zu anderen Datenbanksicherheitsfunktionen für die Datenbank ergriffen werden.

dlchktime - Zeitintervall für Prüfung von Deadlocks

Dieser Parameter definiert die Häufigkeit, mit der alle mit der Datenbank verbundenen Anwendungen vom Datenbankmanager auf Deadlocks überprüft werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 000 (10 Sekunden) [1 000 - 600 000]

Maßeinheit
Millisekunden

Beim Auftreten von Deadlocks müssen zwei oder mehr Anwendungen, die auf dieselbe Datenbank zugreifen, für einen unbegrenzten Zeitraum auf eine Ressource warten. Die Anwendungen warten unendlich lange, da jede Anwendung eine Ressource gesperrt hat, die von der anderen Anwendung zur Fortsetzung der Verarbeitung benötigt wird.

Anmerkung:

1. In einer Umgebung mit partitionierten Datenbanken gilt dieser Parameter nur für den Katalogknoten.
2. In einer Umgebung mit partitionierten Datenbanken wird ein Deadlock erst nach der zweiten Iteration markiert.

Empfehlung: Durch Erhöhung des Werts für diesen Parameter wird die Häufigkeit der Prüfung auf Deadlocks verringert, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung von Deadlocks warten müssen, erhöht wird.

Durch Angabe eines niedrigeren Werts für diesen Parameter wird die Häufigkeit der Prüfung auf Deadlocks erhöht, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung von Deadlocks warten müssen, verkürzt, die Zeit, die der Datenbankmanager für die Prüfung auf Deadlocks verwendet, jedoch verlängert wird. Wenn das Prüfintervall zu klein ist, kann dies zu einer Verschlechterung der Laufzeitleistung führen, weil der Datenbankmanager häufig nach Deadlocks sucht. Wenn dieser Parameter mit einem niedrigeren Wert versehen wird, um den gemeinsamen Zugriff zu verbessern, sollten Sie darauf achten, dass die Parameter *maxlocks* und *locklist* entsprechend eingestellt sind, um unnötige Sperreneskalationen zu vermeiden, die zu Zugriffskonflikten und weiteren Deadlocks führen können.

dyn_query_mgmt - Dynamische SQL- und XQuery-Abfrageverwaltung

Mit diesem Parameter wird festgelegt, ob Query Patroller Informationen zu übergebenen Abfragen erfasst.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
0 (DISABLE) [1(ENABLE), 0 (DISABLE)]

Dieser Parameter ist dort relevant, wo DB2 Query Patroller installiert ist. Wenn dieser Parameter auf den Wert „ENABLE" gesetzt wird, erfasst Query Patroller Informationen über die Abfrage, wie zum Beispiel die ID des übergebenden Benutzers und den geschätzten Ausführungsaufwand, der durch das Optimierungsprogramm berechnet wurde. Anhand dieser Werte wird unter Beachtung der Schwellenwerte auf Benutzer- und Systemebene bestimmt, ob die Abfrage durch Query Patroller verwaltet werden sollte.

Wenn dieser Parameter auf den Wert „DISABLE" gesetzt wird, erfasst Query Patroller keine Informationen zu übergebenen Abfragen, und es findet keine Abfrageverwaltung statt.

enable_xmlchar - Ermöglichen der Konvertierung in XML (Konfigurationsparameter)

Dieser Parameter legt fest, ob für Nicht-BIT DATA CHAR-Ausdrücke (bzw. CHAR-Ausdrücke) in einer SQL-Anweisung XMLPARSE-Operationen durchgeführt werden können.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Yes [Yes; No]

Wenn pureXML-Funktionen in einer Nicht-Unicode-Datenbank verwendet werden, kann es durch die XMLPARSE-Funktion zu Zeichensubstitutionen kommen, da SQL-Zeichenfolgedaten von der Client-Codepage in die Datenbankcodepage und anschließend in die Unicode-Codepage zur internen Speicherung umgesetzt werden. Wenn *enable_xmlchar* auf NO gesetzt wird, wird die Verwendung von Zeichendatentypen bei der XML-Syntaxanalyse blockiert, und bei sämtlichen Versuchen, Zeichentypen in eine Nicht-Unicode-Datenbank einzufügen, wird ein Fehler generiert. Der BLOB-Datentyp und FOR BIT DATA-Datentypen sind weiterhin zulässig, wenn *enable_xmlchar* auf NO gesetzt wird, da keine Codepagekonvertierung erfolgt, wenn diese Datentypen zum Einfügen von XML-Daten in eine Datenbank verwendet werden.

Standardmäßig ist *enable_xmlchar* auf YES gesetzt, sodass die Syntaxanalyse von Zeichendatentypen möglich ist. In diesem Fall müssen Sie sicherstellen, dass sämtliche XML-Daten, die eingefügt werden sollen, ausschließlich Codepunkte enthal-

ten, die Teil der Datenbankcodepage sind, damit keine Substitutionszeichen während des Einfügevorgangs der XML-Daten eingeführt werden.

Anmerkung: Der Client muss die Verbindung zum Agenten unterbrechen und anschließend die Verbindung wiederherstellen, damit diese Änderung wirksam wird.

failarchpath - Protokollarchivpfad für Funktionsübernahme

Mit diesem Parameter wird ein Pfad angegeben, in dem DB2 versucht, Protokolldateien zu archivieren, wenn die Protokolldateien weder am primären noch am sekundären (falls definiert) Archivierungsziel archiviert werden können, weil diese Ziele von einem Datenträgerproblem betroffen sind. Der angegebene Pfad muss auf eine Platte verweisen.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Null [ ]

Wenn sich Protokolldateien in dem durch den aktuellen Wert von *failarchpath* angegebenen Pfad befinden, werden sämtliche Aktualisierungen für *failarchpath* nicht sofort wirksam. Die Aktualisierung wird dann wirksam, wenn die Verbindung zu allen Anwendungen getrennt ist.

groupheap_ratio - Für Anwendungsgruppenzwischenspeicher vorgesehener Speicher in Prozent

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert. In Version 9.5 wurde er durch den Konfigurationsparameter *appl_memory* ersetzt.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Dieser Parameter gibt den Prozentsatz an Speicher im gemeinsam genutzten Speicher zur Anwendungssteuerung an, der für den von der Anwendungsgruppe gemeinsam genutzten Zwischenspeicher vorgesehen ist.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
70 [1 – 99]

Maßeinheit
Prozent

Dieser Parameter hat keine Auswirkungen auf eine nicht partitionierte Datenbank mit inaktiviertem Konzentrator und inaktivierter partitionsinterner Parallelität.

Empfehlung: Behalten Sie den Standardwert für diesen Parameter bei, solange Sie keine Leistungsprobleme feststellen.

hadr_db_role - Rolle der HADR-Datenbank

Dieser Parameter gibt die aktuelle Rolle einer Datenbank an, unabhängig davon, ob sie online oder offline ist.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Informativ

Gültige Werte: STANDARD, PRIMARY oder STANDBY.

Anmerkung: Wenn eine Datenbank aktiv ist, kann die HADR-Rolle der Datenbank auch mit dem Befehl GET SNAPSHOT FOR DATABASE ermittelt werden.

hadr_local_host - Name des lokalen Hosts für HADR

Mit diesem Parameter wird der lokale Host für die HADR-TCP-Kommunikation (High Availability Disaster Recovery) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Es kann entweder ein Hostname oder eine IP-Adresse verwendet werden. Wenn ein Hostname angegeben wird und mehreren IP-Adressen zugeordnet ist, wird ein Fehler zurückgegeben und HADR wird nicht gestartet. Wenn der Hostname mehreren IP-Adressen zugeordnet ist (selbst wenn Sie den gleichen Hostnamen auf dem Primärsystem und dem Bereitschaftssystem angeben), besteht die Möglichkeit, dass das Primärsystem und das Bereitschaftssystem diesen Hostnamen verschiedenen IP-Adressen zuordnen, weil einige DNS-Server IP-Adresslisten in nicht deterministischer Reihenfolge zurückgeben.

Ein Hostname besitzt das Format: meinserver.ibm.com. Eine IP-Adresse besitzt das Format: ″12.34.56.78″.

hadr_local_svc - Lokaler HADR-Servicename

Mit diesem Parameter wird der TCP-Servicename oder die Portnummer angegeben, für die der HADR-Prozess (High Availability Disaster Recovery) Verbindungen akzeptiert.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Der Wert für *hadr_local_svc* auf dem primären Datenbanksystem oder dem Bereitschaftsdatenbanksystem darf nicht mit dem Wert für *svcename* bzw. mit *svcename + 1* auf ihren jeweiligen Knoten übereinstimmen.

hadr_peer_window - HADR-Peerfenster (Konfigurationsparameter)

Wenn Sie den Konfigurationsparameter **hadr_peer_window** auf einen Zeitwert ungleich null setzen, verhält sich das HADR-Datenbankpaar aus Primär- und Bereitschaftsdatenbank über den konfigurierten Zeitraum weiterhin so, als befände es sich im Peerzustand, wenn die Primärdatenbank die Verbindung zur Bereitschaftsdatenbank verliert. Dies hilft bei der Gewährleistung der Datenkonsistenz.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0 – 4 294 967 295]

Maßeinheit
Sekunden

Hinweise:

- Der Wert muss in beiden Datenbanken, d. h. in der Primär- und der Bereitschaftsdatenbank, gleich sein.
- Ein empfohlener Mindestwert beträgt 120 Sekunden.
- Der Wert des Parameters **hadr_peer_window** wird ignoriert, wenn der Parameter **hadr_syncmode** auf den Wert ASYNC gesetzt ist. Das heißt, der Wert wird wie null (0) behandelt, da er im asynchronen Modus irrelevant ist.
- Um eine Beeinträchtigung der Verfügbarkeit der Primärdatenbank zu vermeiden, wenn die Bereitschaftsdatenbank mit Absicht, zum Beispiel zu Wartungszwecken, heruntergefahren wird, wird das Peerfenster nicht aufgerufen, wenn die Bereitschaftsdatenbank explizit inaktiviert wird, während sich das HADR-Datenbankpaar im Peerzustand befindet.

- Der Befehl TAKEOVER HADR mit der Option **PEER WINDOW ONLY** startet eine Übernahmeoperation nur, wenn sich die HADR-Bereitschaftsdatenbank innerhalb des definierten Peerfensters befindet.
- Bei der Übernahmeoperation mit der Option **PEER WINDOW ONLY** kann es zu einer fehlerhaften Funktionsweise kommen, wenn die Uhr der Primärdatenbank und die Uhr der Bereitschaftsdatenbank nicht innerhalb einer Toleranz von höchstens 5 Sekunden miteinander synchronisiert sind. Das heißt, die Operation ist möglicherweise erfolgreich, obwohl sie fehlschlagen sollte, oder sie schlägt fehl, obwohl sie erfolgreich sein sollte. Verwenden Sie einen Zeitsynchronisationsservice (z. B. NTP), um die Uhren über dieselbe Quelle zu synchronisieren.
- In den Bereitschaftsdatenbanken basiert die Endzeit des Peerfensters auf der letzten Überwachungssignalnachricht (Heartbeat), die von der Primärdatenbank empfangen wird, und nicht auf dem Zeitpunkt der Verbindungstrennung. Daher reicht die verbleibende Zeit, die sich die Bereitschaftsdatenbank im Status 'S-DisconnectedPeer' befindet, bevor sie in den Status 'S-RemoteCatchupPending' wechselt, von (**hadr_peer_window - hadr_timeout**) Sekunden bis (**hadr_peer_window**) Sekunden, je nachdem, wann und wie die Verbindung getrennt wurde.

hadr_remote_host - Name des fernen HADR-Hosts

Mit diesem Parameter wird der TCP/IP-Hostname oder die IP-Adresse des fernen HADR-Datenbankservers (High Availability Disaster Recovery) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Ähnlich wie der Parameter *'hadr_local_host'* darf dieser Parameter nur einer IP-Adresse zugeordnet werden.

hadr_remote_inst - HADR-Instanzname des fernen Servers

Mit diesem Parameter wird der Instanzname des fernen Servers angegeben. Verwaltungstools wie die DB2-Steuerzentrale verwenden diesen Parameter, um eine Verbindung zu dem fernen Server herzustellen. Ebenso prüft HADR (High Availability Disaster Recovery), ob eine ferne Datenbank, die eine Verbindung anfordert, zu der deklarierten fernen Instanz gehört.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

hadr_remote_svc - Ferner HADR-Servicename

Mit diesem Parameter wird der TCP-Servicename oder die Portnummer angegeben, die vom fernen HADR-Datenbankserver (High Availability Disaster Recovery) verwendet wird.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus

Mit diesem Parameter wird der Synchronisationsmodus angegeben, der bestimmt, wie die Protokollschreiboperationen der Primärdatenbank mit der Bereitschaftsdatenbank synchronisiert werden, wenn sich die Systeme im Peerstatus befinden.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NEARSYNC [ASYNC; SYNC]

Gültige Werte für diesen Parameter sind:

SYNC
Dieser Modus bietet den größten Schutz gegen Transaktionsverlust, jedoch auf Kosten einer längeren Transaktionsantwortzeit.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben wurden und wenn die Primärdatenbank von der Bereitschaftsdatenbank eine Empfangsbestätigung erhalten hat, dass die Protokolle auch in Protokolldateien in der Bereitschaftsdatenbank geschrieben wurden. So wird sichergestellt, dass die Protokolldaten an beiden Standorten gespeichert werden.

NEARSYNC
Dieser Modus bietet einen etwas geringeren Schutz gegen Transaktionsverlust, jedoch im Gegenzug auch eine kürzere Transaktionsantwortzeit als der Modus SYNC.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben wurden und wenn die Primärdatenbank von der Bereitschaftsdatenbank eine Empfangsbestätigung erhalten hat, dass die Protokolle auch in den Hauptspeicher des Bereitschaftssystems geschrieben wurden. Zu Datenverlust kann es nur dann kommen, wenn beide Standorte gleichzeitig ausfallen und wenn der Zielstandort nicht alle empfangenen Protokolldaten an nicht flüchtigen Speicher übertragen hat.

ASYNC

Dieser Modus hat das größte Risiko eines Transaktionsverlusts im Fall eines Ausfalls der Primärdatenbank, bietet allerdings auch die kürzeste Transaktionsantwortzeit unter den drei Modi.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben und an die TCP-Schicht der Hostmaschine des primären Systems übergeben wurden. Da das primäre System nicht auf eine Empfangsbestätigung vom Bereitschaftssystem wartet, können Transaktionen bereits als festgeschrieben betrachtet werden, während sie noch an das Bereitschaftssystem weitergeleitet werden.

hadr_timeout - HADR-Zeitlimitwert

Dieser Parameter gibt die Zeit (in Sekunden) an, die der HADR-Prozess (High Availability Disaster Recovery) wartet, bevor ein Kommunikationsversuch als fehlgeschlagen eingestuft wird.

Konfigurationstyp

Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

Konfigurierbar

Standardwert [Bereich]

120 [1 - 4 294 967 295]

indexrec - Zeitpunkt für Indexneuerstellung

Dieser Parameter gibt an, wann der Datenbankmanager versucht, ungültige Indizes neu zu erstellen und ob eine Indexerstellung während einer aktualisierenden DB2-Recovery oder einer Wiedergabe des HADR-Protokolls in der Bereitschaftsdatenbank wiederholt wird.

Konfigurationstyp

Datenbank und Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp

Online konfigurierbar

Weitergabeklasse

Sofort

Standardwert [Bereich]

UNIX Datenbankmanager
restart [restart; restart_no_redo; access; access_no_redo]

Windows Datenbankmanager
restart [restart; restart_no_redo; access; access_no_redo]

Datenbank
Systemeinstellung verwenden [system; restart; restart_no_redo; access; access_no_redo]

Für diesen Parameter sind fünf Einstellungen möglich:

SYSTEM
Verwenden Sie die Systemeinstellung, die in der Konfigurationsdatei des Datenbankmanagers angegeben wurde, um festzulegen, wann ungültige Indizes neu erstellt werden und ob Protokollsätze zur Indexerstellung während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt werden müssen. (Anmerkung: Diese Einstellung ist nur für Datenbankkonfigurationen gültig.)

ACCESS
Ungültige Indizes werden neu erstellt, wenn zum ersten Mal auf den Index zugegriffen wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt.

ACCESS_NO_REDO
Ungültige Indizes werden erneut erstellt, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird. Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt; diese Indizes bleiben ungültig. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt.

RESTART
Die Standardeinstellung für *indexrec*. Ungültige Indizes werden neu erstellt, wenn der Befehl RESTART DATABASE explizit oder implizit abgesetzt wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme neu erstellt.

Beachten Sie, dass der Befehl RESTART DATABASE implizit abgesetzt wird, wenn der Parameter *autorestart* aktiviert ist.

RESTART_NO_REDO
Ungültige Indizes werden neu erstellt, wenn der Befehl RESTART DATABASE explizit oder implizit abgesetzt wird. (Der Befehl RESTART DATABASE wird implizit abgesetzt, wenn der Parameter *autorestart* aktiviert ist.)Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden DB2-Recovery oder während einer Wiedergabe des

HADR-Protokolls wiederholt; stattdessen werden diese Indizes neu erstellt, wenn die aktualisierende Recovery beendet wird oder wenn die HADR-Übernahme stattfindet.

Indizes können ungültig werden, wenn nicht behebbare Plattenfehler auftreten. Wenn dabei die Daten selbst beschädigt werden, können die Daten verloren gehen. Wird jedoch ein Index beschädigt, kann der Index durch Neuerstellung wiederhergestellt werden. Wird ein Index neu erstellt, während Benutzer mit der Datenbank verbunden sind, können zwei Probleme auftreten:

- Während der Erstellung der Indexdatei könnte es zu einer unerwarteten Verschlechterung der Antwortzeit kommen. Benutzer, die auf die Tabelle zugreifen und diesen bestimmten Index verwenden, müssen auf die Neuerstellung des Index warten.
- Nach der Indexneuerstellung könnten unerwartete, aktive Sperren beibehalten werden, besonders dann, wenn die Benutzertransaktion, die die Indexneuerstellung ausgelöst hat, keine Commit- oder Rollbackoperation ausgeführt hat.

Empfehlung: Die beste Option für diesen Parameter auf einem Server mit vielen Benutzern, wenn die Zeit für den Neustart keine kritische Rolle spielt, ist die Indexneuerstellung beim Neustart der Datenbank (DATABASE RESTART) als Teil des Vorgangs, mit dem die Datenbank nach einem Systemabsturz wiederhergestellt und online verfügbar gemacht wird.

Wenn dieser Parameter auf „ACCESS" oder auf „ACCESS_NO_REDO" gesetzt wird, verschlechtert sich während der Indexneuerstellung die Leistung des Datenbankmanagers. Jeder Benutzer, der auf den Index oder die Tabelle zugreift, der bzw. die gerade neu erstellt wird, muss zunächst auf die Beendigung der Indexneuerstellung warten.

Wenn dieser Parameter auf „RESTART" gesetzt wird, dauert der Neustart der Datenbank wegen der Indexneuerstellung länger, jedoch wird die normale Verarbeitung nicht mehr beeinträchtigt, sobald die Datenbank wieder online verfügbar ist.

Anmerkung: Bei der Datenbankrecovery werden alle ausführbaren Dateien von SQL-Prozeduren im Dateisystem entfernt, die zu der gerade wiederhergestellten Datenbank gehören. Wenn der Parameter *indexrec* auf RESTART gesetzt wird, werden alle ausführbaren Dateien von SQL-Prozeduren aus dem Datenbankkatalog extrahiert und in das Dateisystem zurückgestellt, wenn die nächste Verbindung zur Datenbank hergestellt wird. Wenn der Parameter *indexrec* nicht auf RESTART gesetzt ist, wird die ausführbare Datei einer SQL-Prozedur erst in das Dateisystem extrahiert, wenn diese SQL-Prozedur zum ersten Mal ausgeführt wird.

Der Unterschied zwischen den Werten RESTART und RESTART_NO_REDO bzw. zwischen den Werten ACCESS und ACCESS_NO_REDO ist nur von Bedeutung, wenn die vollständige Protokollierung für Indexerstellungsoperationen, wie z. B. die Operationen CREATE INDEX und REORG INDEX, oder für eine Indexneuerstellung aktiviert ist. Sie können die Protokollierung aktivieren, indem Sie den Datenbankkonfigurationsparameter *logindexbuild* oder LOG INDEX BUILD beim Ändern einer Tabelle aktivieren. Wenn Sie *indexrec* entweder auf RESTART oder auf ACCESS setzen, kann für Operationen, die eine protokollierte Indexerstellung einbeziehen, eine aktualisierende Recovery durchgeführt werden, ohne dass das Indexobjekt einen ungültigen Status aufweist; dies hätte zur Folge, dass der Index später neu erstellt werden müsste.

jdk_64_path - Installationspfad für 64-Bit Software Developer's Kit für Java auf DAS

Dieser Parameter gibt das Verzeichnis an, in dem das 64-Bit-Software Developer's Kit (SDK) für Java installiert ist, das zu verwenden ist, um DB2-Verwaltungsserver-funktionen auszuführen.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Null [beliebiger gültiger Pfad]

Anmerkung: Dieser Parameter unterscheidet sich vom Konfigurationsparameter **jdk_path**, der ein 32-Bit-SDK für Java angibt.

Die vom Java-Interpreter verwendeten Umgebungsvariablen werden aus dem Wert dieses Parameters berechnet. Dieser Parameter wird nur auf den Plattformen verwendet, die 32- und 64-Bit-Instanzen unterstützen.

Bei den Plattformen handelt es sich um folgende:

- 64-Bit-Kernel für AIX, HP-UX und Solaris Operating System
- 64-Bit-Windows unter X64 und IPF
- 64-Bit-Linux-Kernel auf AMD64- und Intel EM64T-Systemen (x64), POWER und zSeries

Bei allen anderen Plattformen wird nur **jdk_path** verwendet.

Da es für diesen Parameter keinen Standardwert gibt, sollten Sie beim Installieren des SDK für Java einen Wert angeben.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

locklist - Maximaler Speicher für Sperrenliste

Dieser Parameter gibt die Speichermenge an, die für die Sperrenliste zugeordnet wird. Für jede Datenbank gibt es eine Sperrenliste, die die Sperren aller gleichzeitig mit der Datenbank verbundenen Anwendungen enthält.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX Automatic [4 - 524 288]

Windows-Datenbankserver mit lokalen und fernen Clients
Automatic [4 - 524 288]

Windows-64-Bit-Datenbankserver mit lokalen Clients
Automatic [4 - 524 288]

Windows-32-Bit-Datenbankserver mit lokalen Clients
Automatic [4 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die erste Anwendung die Verbindung zur Datenbank herstellt

Freigabe
Wenn die letzte Anwendung die Verbindung zur Datenbank beendet

Das Sperren ist eine Funktion des Datenbankmanagers zur Steuerung des gleichzeitigen Zugriffs auf Daten der Datenbank durch mehrere Anwendungen. Sowohl Zeilen als auch Tabellen können gesperrt werden. Der Datenbankmanager fordert eventuell auch Sperren zur internen Verwendung an.

Der Wert AUTOMATIC für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch anzupassen. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherverbraucher verteilt, müssen mindestens zwei Speicherverbraucher für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Der Wert des Parameters *locklist* wird zusammen mit dem Parameter *maxlocks* optimiert, sodass eine Inaktivierung der automatischen Optimierung des Parameters *locklist* automatisch auch eine Inaktivierung der automatischen Optimierung des Parameters *maxlocks* bewirkt. Die Aktivierung der automatischen Optimierung für den Parameter *locklist* führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters *maxlocks*.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter *self_tuning_mem* muss auf "ON" gesetzt sein).

Auf 32-Bit-Plattformen erfordert jede Sperre 48 oder 96 Byte der Sperrenliste, und zwar in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind:

- 96 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, für das keine anderen Sperren aktiv sind.
- 48 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Auf 64-Bit-Plattformen (außer HP-UX/PA-RISC) erfordert jede Sperre 64 oder 128 Byte der Sperrenliste in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind:

- 128 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, für das keine anderen Sperren aktiviert sind.

- 64 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Auf HP-UX/PA-RISC-Plattformen (64-Bit) erfordert jede Sperre 80 oder 160 Byte der Sperrenliste in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind.

Wenn der Prozentsatz der Sperrenliste, der von einer Anwendung verwendet wird, den Wert des Parameters *maxlocks* erreicht, führt der Datenbankmanager eine Sperreneskalation von Zeilenebene auf Tabellenebene für die Sperren aus, die von dieser Anwendung aktiviert wurden. Obwohl der Eskalationsprozess an sich nicht viel Zeit in Anspruch nimmt, wird durch Sperren ganzer Tabellen (im Vergleich zum Sperren einzelner Zeilen) der gemeinsame Zugriff eingeschränkt und die allgemeine Datenbankleistung kann bei nachfolgenden Zugriffen auf die betroffenen Tabellen beeinträchtigt werden. Es wird empfohlen, die Größe der Sperrenliste folgendermaßen klein zu halten:

- Führen Sie häufig Commits aus, um Sperren freizugeben.
- Wenn viele Aktualisierungen ausgeführt werden sollen, sperren Sie vor den Aktualisierungen die ganze Tabelle (mit der SQL-Anweisung LOCK TABLE). Dadurch wird nur eine Sperre verwendet und der Zugriff anderer auf die zu aktualisierenden Daten verhindert. Der gemeinsame Zugriff auf die Daten wird jedoch eingeschränkt.

  Sie können die Option LOCKSIZE der Anweisung ALTER TABLE auch verwenden, um das Sperren einer bestimmten Tabelle zu steuern.

  Die Verwendung der Isolationsstufe RR (Wiederholtes Lesen) kann zu einer automatischen Tabellensperre führen.
- Verwenden Sie die Isolationsstufe der Cursorstabilität (CS), wenn möglich, um die Anzahl der Sperren für gemeinsamen Zugriff zu verringern. Wenn dadurch die durch die Anwendung festgelegten Integritätsanforderungen nicht beeinträchtigt werden, verwenden Sie anstatt der Cursorstabilität die Isolationsstufe UR (Nicht festgeschriebener Lesevorgang), um die Anzahl der Sperren weiter zu verringern.
- Setzen Sie *locklist* auf AUTOMATIC. Die Sperrenliste wird synchron vergrößert, um eine Sperreneskalation bzw. ein vollständiges Füllen der Sperrenliste zu vermeiden.

Wenn die Sperrenliste voll ist, kann die Leistung herabgesetzt werden, da die Sperreneskalation mehr Tabellensperren und weniger Zeilensperren erzeugt und auf diese Weise den gemeinsamen Zugriff auf gemeinsam benutzte Objekte in der Datenbank einschränkt. Zudem kann es mehr Deadlocks zwischen Anwendungen geben (da sie alle auf eine verringerte Anzahl von Tabellensperren warten), was dazu führt, dass Transaktionen mit ROLLBACK rückgängig gemacht werden. Ihre Anwendung empfängt einen SQLCODE-Wert -912, wenn die maximale Anzahl von Sperranforderungen für die Datenbank erreicht wurde.

Empfehlung: Wenn Sperreneskalationen zu Leistungseinbußen führen, müssen Sie eventuell den Wert dieses Parameters oder den Wert des Parameters *maxlocks* erhöhen. Mit dem Datenbanksystemmonitor können Sie feststellen, ob Sperreneskalationen auftreten. Weitere Informationen finden Sie in der Beschreibung des Monitorelements *lock_escals* (Sperreneskalationen).

Die folgenden Schritte können Ihnen bei der Ermittlung der Anzahl der Seiten, die für Ihre Sperrenliste erforderlich sind, vielleicht helfen:

1. Berechnen Sie eine Untergrenze für die Größe der Sperrenliste, und verwenden Sie dazu *eine* der folgenden Formeln, abhängig von Ihrer jeweiligen Umgebung:

a.

```
(512 * x * maxappls) / 4096
```

b. Bei aktiviertem Konzentrator:

```
(512 * x * max_coordagents) / 4096
```

c. In einer partitionierten Datenbank mit aktiviertem Konzentrator:

```
(512 * x * max_coordagents * Anzahl Datenbankpartitionen) / 4096
```

Dabei ist 512 ein Schätzwert für die durchschnittliche Anzahl von Sperren pro Anwendung, und x ist die Anzahl der benötigten Byte für jede Sperre eines Objekts, für das bereits eine Sperre aktiv ist (40 Byte auf 32-Bit-Plattformen, 64 Byte auf 64-Bit-Plattformen).

2. Berechnen Sie eine Obergrenze für die Größe der Sperrenliste:

```
(512 * y * maxappls) / 4096
```

In dieser Formel ist y die Anzahl der benötigten Byte für die erste Sperre eines Objekts (80 Byte auf 32-Bit-Plattformen, 128 Byte auf 64-Bit-Plattformen).

3. Schätzen Sie das Aufkommen an gemeinsamem Zugriff durch Anwendungen auf Ihre Daten, und wählen Sie nach Ihren Schätzungen einen Anfangswert für *locklist*, der zwischen der berechneten Ober- und Untergrenze liegt.
4. Mit dem Datenbanksystemmonitor können Sie, wie unten beschrieben, den Wert dieses Parameters optimieren.

Wenn *maxappls* oder *max_coordagents* in Ihrem Anwendungsszenario auf AUTOMATIC gesetzt ist, müssen Sie auch *locklist* auf AUTOMATIC setzen.

Sie können mit dem Datenbanksystemmonitor die maximale Anzahl der von einer bestimmten Transaktion aktivierten Sperren feststellen. Weitere Informationen finden Sie in der Beschreibung des Monitorelements *locks_held_top (Maximale Anzahl aktiver Sperren)*.

Anhand dieser Informationen können Sie die geschätzte Anzahl der Sperren pro Anwendung bestätigen oder anpassen. Um diese Auswertung durchzuführen, müssen Sie mehrere Probeanwendungen ausführen und dabei beachten, dass die Monitordaten auf einer Transaktionsebene und nicht auf einer Anwendungsebene geliefert werden.

Der Wert des Parameters *locklist* sollte eventuell auch dann heraufgesetzt werden, wenn der Parameter *maxappls* erhöht wird oder wenn die Anwendungen, die ausgeführt werden, nur relativ selten Commits ausführen.

Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl REBIND).

locktimeout - Zeitlimit für Sperren

Mit diesem Parameter wird die Anzahl der Sekunden angegeben, die eine Anwendung auf eine Sperre wartet; dadurch werden globale Deadlocks für Anwendungen vermieden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
-1 [-1; 0 - 32 767]

Maßeinheit
Sekunden

Wenn dieser Parameter auf `0` gesetzt wird, wird nicht auf Sperren gewartet. In diesem Fall erhält die Anwendung, wenn zum Zeitpunkt der Anforderung keine Sperre verfügbar ist, sofort die Rückmeldung -911.

Wenn dieser Parameter auf den Wert `-1` gesetzt wird, wird die Überwachung des Zeitlimits für Sperren inaktiviert. In diesem Fall wird auf eine Sperre gewartet (wenn zum Zeitpunkt der Anforderung keine verfügbar ist), bis eine der folgenden Situationen eintritt:

- Die Sperre wird erteilt
- Ein Deadlock tritt ein

Empfehlung: In einer Umgebung, in der Transaktionen verarbeitet werden (OLTP-Umgebung), können Sie einen Anfangswert von 30 Sekunden verwenden. In einer Umgebung, in der nur Abfragen durchgeführt werden, können Sie mit einem höheren Wert beginnen. In beiden Fällen sollten Sie diesen Parameter jedoch mithilfe von Vergleichstests (Benchmark-Tests) optimieren.

Der Wert sollte so eingestellt werden, dass schnell erkannt wird, wenn Wartezeiten aufgrund außergewöhnlicher Situationen auftreten, wie z. B. eine blockierte Transaktion (weil ein Benutzer seine Workstation verlassen hat). Sie sollten den Wert so hoch festlegen, dass gültige Sperrenanforderungen das Zeitlimit nicht aufgrund von Spitzenbelastungen überschreiten, da bei hoher Systemauslastung länger auf Sperren gewartet werden muss.

Mit dem Datenbanksystemmonitor können Sie die Häufigkeit verfolgen, mit der eine Anwendung (eine Verbindung) das Zeitlimit für Sperren überschreitet, oder erkennen, dass eine Datenbank eine Zeitlimitüberschreitung für alle verbundenen Anwendungen festgestellt hat.

Hohe Werte für das Monitorelement *lock_timeout* (Anzahl Zeitlimitüberschreitungen für Sperren) können folgende Ursachen haben:

- Ein zu niedriger Wert für diesen Konfigurationsparameter
- Eine Anwendung (Transaktion), die Sperren über einen längeren Zeitraum aufrechterhält. Mithilfe des Datenbanksystemmonitors können Sie solche Anwendungen näher untersuchen.
- Ein Problem in Bezug auf den gleichzeitigen Zugriff, verursacht durch Sperreneskalationen (von Sperren auf Zeilenebene zu Sperren auf Tabellenebene).

log_retain_status - Statusanzeiger für Beibehalten der Protokolle

Wenn dieser Parameter gesetzt ist (und die Parametereinstellung *logretain* auf Recovery gesetzt ist), zeigt er an, dass Protokolldateien für die aktualisierende Recovery gespeichert werden.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

logarchmeth1 - Primäre Protokollarchivierungsmethode

Mit diesem Parameter wird der Datenträgertyp des primären Ziels für archivierte Protokolle angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
OFF [LOGRETAIN, USEREXIT, DISK, TSM, VENDOR]

OFF Gibt an, dass die Protokollarchivierungsmethode nicht verwendet werden soll. Wenn sowohl *logarchmeth1* als auch *logarchmeth2* auf OFF gesetzt ist, heißt dies, dass die Datenbank Umlaufprotokolle verwendet. Sie kann in diesem Fall nicht aktualisierend wiederhergestellt werden. Dies ist der Standardwert.

LOGRETAIN
Dieser Wert kann nur für *logarchmeth1* verwendet werden und ist äquivalent zur Einstellung des Konfigurationsparameters *logretain* auf den Wert RECOVERY. Wenn Sie diesen Wert angeben, wird der Konfigurationsparameter *logretain* automatisch aktualisiert.

USEREXIT
Dieser Wert kann nur für *logarchmeth1* verwendet werden und ist äquivalent zur Einstellung des Konfigurationsparameters *userexit* auf den Wert ON. Wenn Sie diesen Wert angeben, wird der Konfigurationsparameter *userexit* automatisch aktualisiert.

DISK Auf diesen Wert muss ein Doppelpunkt (:) folgen und darauf der vollständig qualifizierte Name des Pfads, in dem die Protokolldateien archiviert werden sollen. Wenn Sie beispielsweise *logarchmeth1* auf DISK:/u/dbuser/archived_logs setzen, werden die archivierten Protokolldateien in ein Verzeichnis mit dem Namen /u/dbuser/archived_logs gestellt.

Anmerkung: Wenn Sie auf Band archivieren, können Sie zum Speichern und Abrufen der Protokolldateien das Dienstprogramm db2tapemgr verwenden.

TSM Wenn dieser Wert ohne weitere Konfigurationsparameter angegeben wird, werden Protokolldateien unter Verwendung der Standardmanagementklasse auf dem lokalen TSM-Server archiviert. Folgen auf diesen Wert ein Doppelpunkt (:) und eine TSM-Managementklasse, werden die Protokolldateien unter Verwendung der angegebenen Managementklasse archiviert.

Wenn Protokolle mit TSM archiviert werden, versucht TSM vor der Verwendung der durch den Datenbankkonfigurationsparameter angegebenen Managementklasse zunächst, das Objekt an die Managementklasse zu binden, die in der Liste INCLUDE-EX-

CLUDE angegeben ist, die sich in der Datei mit den TSM-Clientoptionen befindet. Wird keine Übereinstimmung gefunden, wird die TSM-Standardmanagementklasse verwendet, die auf dem TSM-Server angegeben ist. Anschließend bindet TSM das Objekt erneut an die mit dem Datenbankkonfigurationsparameter angegebene Managementklasse.

Daher muss sowohl die Standardmanagementklasse als auch die durch den Datenbankkonfigurationsparameter angegebene Managementklasse eine Archivierungskopiengruppe enthalten, da die Archivierungsoperation sonst fehlschlägt.

VENDOR
Gibt an, dass zur Archivierung der Protokolldateien eine Bibliothek eines anderen Lieferanten verwendet wird. Auf diesen Wert müssen ein Doppelpunkt (:) und der Name der Bibliothek folgen. Die in der Bibliothek zur Verfügung gestellten APIs müssen die Backup- und Restore-APIs für Produkte anderer Lieferanten verwenden.

Hinweis:

1. Wenn *logarchmeth1* oder *logarchmeth2* auf einen anderen Wert als OFF gesetzt ist, ist die Datenbank für die aktualisierende Recovery konfiguriert.
2. Wenn Sie die Konfigurationsparameter *userexit* oder *logretain* aktualisieren, wird *logarchmeth1* automatisch aktualisiert und umgekehrt. Wenn Sie *userexit* oder *logretain* verwenden, muss *logarchmeth2* auf den Wert OFF gesetzt werden.

logarchmeth2 - Sekundäre Protokollarchivierungsmethode

Mit diesem Parameter wird der Datenträgertyp des sekundären Ziels für archivierte Protokolle angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
OFF [LOGRETAIN, USEREXIT, DISK, TSM, VENDOR]

OFF Gibt an, dass die Protokollarchivierungsmethode nicht verwendet werden soll. Wenn sowohl *logarchmeth1* als auch *logarchmeth2* auf OFF gesetzt ist, heißt dies, dass die Datenbank Umlaufprotokolle verwendet. Sie kann in diesem Fall nicht aktualisierend wiederhergestellt werden. Dies ist der Standardwert.

LOGRETAIN
Dieser Wert kann nur für *logarchmeth1* verwendet werden und ist äquivalent zur Einstellung des Konfigurationsparameters *logretain*

auf den Wert RECOVERY. Wenn Sie diesen Wert angeben, werden die Konfigurationsparameter *logretain* automatisch aktualisiert.

USEREXIT
Dieser Wert kann nur für *logarchmeth1* verwendet werden und ist äquivalent zur Einstellung des Konfigurationsparameters *userexit* auf den Wert ON. Wenn Sie diesen Wert angeben, wird der Konfigurationsparameter *userexit* automatisch aktualisiert.

DISK Auf diesen Wert muss ein Doppelpunkt (:) folgen und darauf der vollständig qualifizierte Name des Pfads, in dem die Protokolldateien archiviert werden sollen. Wenn Sie beispielsweise *logarchmeth1* auf DISK:/u/dbuser/archived_logs setzen, werden die archivierten Protokolldateien in ein Verzeichnis mit dem Namen /u/dbuser/archived_logs gestellt.

Anmerkung: Wenn Sie auf Band archivieren, können Sie zum Speichern und Abrufen der Protokolldateien das Dienstprogramm db2tapemgr verwenden.

TSM Wenn dieser Wert ohne weitere Konfigurationsparameter angegeben wird, werden Protokolldateien unter Verwendung der Standardmanagementklasse auf dem lokalen TSM-Server archiviert. Folgen auf diesen Wert ein Doppelpunkt (:) und eine TSM-Managementklasse, werden die Protokolldateien unter Verwendung der angegebenen Managementklasse archiviert.

VENDOR
Gibt an, dass zur Archivierung der Protokolldateien eine Bibliothek eines anderen Lieferanten verwendet wird. Auf diesen Wert müssen ein Doppelpunkt (:) und der Name der Bibliothek folgen. Die in der Bibliothek zur Verfügung gestellten APIs müssen die Back-up- und Restore-APIs für Produkte anderer Lieferanten verwenden.

Hinweis:

1. Wenn *logarchmeth1* oder *logarchmeth2* auf einen anderen Wert als OFF gesetzt ist, ist die Datenbank für die aktualisierende Recovery konfiguriert.
2. Wenn Sie die Konfigurationsparameter *userexit* oder *logretain* aktualisieren, wird *logarchmeth1* automatisch aktualisiert und umgekehrt. Wenn Sie *userexit* oder *logretain* verwenden, muss *logarchmeth2* auf den Wert OFF gesetzt werden.

Wenn dieser Pfad angegeben wird, werden Protokolldateien sowohl an diesem Ziel als auch an dem Ziel archiviert, das durch den Datenbankkonfigurationsparameter *logarchmeth1* angegeben wird.

logarchopt1 - Optionen für primäres Protokollarchiv

Mit diesem Parameter wird das Feld der Optionen für das primäre Ziel für archivierte Protokolle (falls erforderlich) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client

- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
NULL [nicht anwendbar]

logarchopt2 - Optionen für sekundäres Protokollarchiv

Mit diesem Parameter wird das Feld der Optionen für das sekundäre Ziel für archivierte Protokolle (falls erforderlich) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
NULL [nicht anwendbar]

logbufsz - Protokollpuffergröße

Mit diesem Parameter kann die Menge an Datenbankzwischenspeicher (der durch den Parameter *dbheap* definiert wird) angegeben werden, die als Puffer für Protokollsätze verwendet werden soll, bevor diese Protokollsätze auf die Festplatte geschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

32-Bit-Plattformen
8 [4 - 4 096]

64-Bit-Plattformen
8 [4 - 65 535]

Ab Version 9.5 Fixpack 3 gilt der folgende erweiterte Bereich: 8 [4 - 131 070]

Maßeinheit
Seiten (4 KB)

Protokollsätze werden in folgenden Fällen auf die Festplatte geschrieben:

- Eine Transaktion oder eine Gruppe von Transaktionen wird entsprechend der Angaben für den Konfigurationsparameter *mincommit* festgeschrieben.
- Der Protokollpuffer ist voll.

- Ein anderes internes Ereignis im Datenbankmanager macht das Schreiben auf die Festplatte erforderlich.

Der Wert dieses Parameters muss außerdem kleiner oder gleich dem Wert des Parameters *dbheap* sein. Durch Puffern der Protokollsätze werden E/A-Operationen für die Protokolldateien effektiver, da weniger häufig Schreiboperationen ausgeführt und bei jeder Schreiboperation mehr Protokollsätze auf die Festplatte geschrieben werden.

Empfehlung: Erhöhen Sie den Wert für die Größe dieses Pufferbereichs, wenn umfangreiche Leseaktivitäten auf einer dedizierten Protokollplatte auftreten oder die Festplatte in erheblichem Maße beansprucht wird. Wenn Sie den Wert dieses Parameters erhöhen, sollten Sie auch den Parameter *dbheap* berücksichtigen, da der Protokollpuffer einen Speicherbereich verwendet, der mit dem Parameter *dbheap* gesteuert wird.

Sie können mit dem Datenbanksystemmonitor feststellen, wie viel Speicherbereich des Protokollpuffers für eine bestimmte Transaktion (oder UOW = Unit of Work, Arbeitseinheit) verwendet wird. Weitere Informationen finden Sie in der Beschreibung des Monitorelements *log_space_used* (verwendeter Speicherbereich für UOW).

logfilsiz - Protokolldateigröße

Mit diesem Parameter wird die Größe jeder primären und sekundären Protokolldatei definiert. Die Größe dieser Protokolldateien beschränkt die Anzahl der Protokollsätze, die in die Dateien geschrieben werden können, bevor sie voll sind und eine neue Protokolldatei erforderlich wird.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

UNIX 1000 [4 - 524 286]

Ab Version 9.5 Fixpack 3 gilt der folgende erweiterte Bereich: 1000 [4 - 1 048 572]

Windows
1000 [4 - 524 286]

Ab Version 9.5 Fixpack 3 gilt der folgende erweiterte Bereich: 1000 [4 - 1 048 572]

Maßeinheit
Seiten (4 KB)

Die Verwendung primärer und sekundärer Protokolldateien sowie die Maßnahmen, die ausgeführt werden, wenn eine Protokolldatei voll ist, sind von der Art der ausgeführten Protokollierung abhängig:

- Umlaufprotokollierung

  Eine primäre Protokolldatei kann wieder verwendet werden, wenn die in ihr aufgezeichneten Änderungen mit COMMIT festgeschrieben wurden. Wenn die Größe der Protokolldatei beschränkt ist und Anwendungen eine große Anzahl von Änderungen an der Datenbank vorgenommen haben, ohne die Änderungen festzuschreiben, kann eine primäre Protokolldatei schnell voll werden. Wenn alle

primären Protokolldateien voll sind, ordnet der Datenbankmanager sekundäre Protokolldateien für die neuen Protokollsätze zu.

- Protokollierung mit Protokollspeicherung

  Wenn eine primäre Protokolldatei voll ist, wird das Protokoll archiviert und eine neue primäre Protokolldatei zugeordnet.

Empfehlung: Die Größe der Protokolldateien muss mit der Anzahl der primären Protokolldateien abgestimmt werden:

- Der Wert des Parameters *logfilsiz* sollte erhöht werden, wenn an der Datenbank eine große Anzahl von Aktualisierungs-, Lösch- oder Einfügetransaktionen ausgeführt wird, durch die die Protokolldatei sehr schnell voll wird.

  Anmerkung: Der obere Grenzwert für die Größe der Protokolldatei zusammen mit dem oberen Grenzwert für die Anzahl Protokolldateien (*logprimary + logsecond*) ergibt einen oberen Grenzwert von 512 GB für den aktiven Protokollspeicherbereich. Ab Version 9.5 Fixpack 3 steht für den aktiven Protokollspeicherbereich eine Größe von 1024 GB zur Verfügung.

  Eine zu kleine Protokolldatei kann sich auf die Systemleistung auswirken, da das Archivieren alter Protokolldateien, das Zuordnen neuer Protokolldateien sowie das Warten auf verwendbare Protokolldateien einen erhöhten Systemaufwand mit sich bringen.
- Der Wert des Parameters *logfilsiz* sollte verringert werden, wenn Plattenspeicherplatz knapp ist, da primäre Protokolldateien im Voraus mit dieser Größe zugeordnet werden.

  Eine zu große Protokolldatei kann Ihre Flexibilität bei der Verwaltung archivierter Protokolldateien bzw. beim Kopieren der Protokolldateien einschränken, da auf einigen Platten vielleicht keine vollständige Protokolldatei gespeichert werden kann.

Wenn Sie die Protokollspeicherung verwenden, wird die aktuelle aktive Protokolldatei geschlossen und abgeschnitten, wenn die letzte Anwendung die Verbindung zu einer Datenbank trennt. Für die nächste zur Datenbank hergestellte Verbindung wird die nächste Protokolldatei verwendet. Wenn Sie also die Protokollanforderungen Ihrer gleichzeitig ausgeführten Anwendungen kennen, können Sie vielleicht eine Protokolldateigröße festlegen, durch die nicht zu viel Speicher umsonst zugeordnet wird.

loghead - Erste aktive Protokolldatei

Dieser Parameter gibt den Namen der zurzeit aktiven Protokolldatei an.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

logindexbuild - Erstellte Indexseiten protokollieren

Mit diesem Parameter wird angegeben, ob Operationen zur Erstellung, erneuten Erstellung oder Reorganisation von Indizes protokolliert werden sollen, sodass Indizes bei DB2-Operationen zur aktualisierenden Recovery oder bei Prozeduren zum Nachvollziehen von HADR-Protokollen (High Availability Disaster Recovery) wiederhergestellt werden können.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Off [On; Off]

logpath - Pfad zu Protokolldateien

Dieser Parameter gibt den aktuellen Pfad an, der für die Protokolldateien verwendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter kann nicht direkt geändert werden, da der Wert vom Datenbankmanager festgelegt wird, wenn eine Änderung am Parameter *newlogpath* wirksam wird.

Bei der Erstellung einer Datenbank wird die zugehörige Datei des Recoveryprotokolls in einem Unterverzeichnis des Verzeichnisses erstellt, in dem sich die Datenbank befindet. Standardmäßig wird dieses Unterverzeichnis mit dem Namen SQLOGDIR in dem Verzeichnis angelegt, das für die Datenbank erstellt wurde.

logprimary - Anzahl primärer Protokolldateien

Mit diesem Parameter kann die Anzahl der im Voraus zugeordneten primären Protokolldateien festgelegt werden. Die primären Protokolldateien reservieren eine feste Menge Speicher, der den Recoveryprotokollen zugeordnet wird.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
3 [2 - 256]

Maßeinheit
Zähler

Zuordnung

- Wenn die Datenbank erstellt wird.
- Wenn ein Protokoll an eine andere Speicherposition versetzt wird (dies geschieht, wenn der Parameter *logpath* aktualisiert wird).
- Wenn die Datenbank das nächste Mal nach einer Erhöhung des Werts dieses Parameters (*logprimary*) gestartet wird, sofern die Datenbank nicht als HADR-Bereitschaftsdatenbank gestartet wird.

- Wenn nach der Archivierung einer Protokolldatei eine neue Protokolldatei zugeordnet wird. (Der Parameter *logretain* oder *userexit* muss aktiviert sein.)
- Wenn der Parameter *logfilsiz* geändert wurde, wird die Größe der Protokolldateien beim nächsten Start der Datenbank geändert, sofern die Datenbank nicht als HADR-Bereitschaftsdatenbank gestartet wird.

Freigabe

Eine Freigabe erfolgt, wenn der Wert für diesen Parameter herabgesetzt wird. Wenn der Wert herabgesetzt wird, werden nicht mehr benötigte Protokolldateien bei der nächsten Verbindung zur Datenbank gelöscht.

Bei der Umlaufprotokollierung werden die primären Protokolldateien nacheinander wiederholt verwendet. Das heißt, wenn ein Protokoll voll ist, wird die nächste primäre Protokolldatei in der Reihenfolge verwendet, wenn sie verfügbar ist. Ein Protokoll wird als verfügbar angesehen, wenn alle UOWs (UOW = Unit of Work, Arbeitseinheit) mit Protokollsätzen in diesem Protokoll mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht wurden. Wenn die nächste Protokolldatei in der Reihenfolge nicht verfügbar ist, wird eine sekundäre Protokolldatei zugeordnet und verwendet. Es werden solange weitere sekundäre Protokolldateien zugeordnet und verwendet, bis die nächste primäre Protokolldatei in der Reihenfolge verfügbar wird oder der durch den Parameter *logsecond* festgelegte Grenzwert erreicht wird. Sobald diese sekundären Protokolldateien vom Datenbankmanager nicht mehr benötigt werden, wird der für sie verwendete Speicher wieder freigegeben.

Für die Anzahl der primären und sekundären Protokolldateien muss Folgendes gelten:

- Wenn *logsecond* den Wert -1 besitzt, ist *logprimary* <= 256.
- Wenn *logsecond* nicht den Wert -1 besitzt, ist (*logprimary* + *logsecond*) <= 256.

Empfehlung: Der Wert, der für diesen Parameter gewählt wird, ist von einer Reihe von Faktoren abhängig, zu denen auch die Art der Protokollierung, die Größe der Protokolldateien und die Art der Verarbeitungsumgebung (z.B. die Länge der Transaktionen und die Häufigkeit der Commits) gehören.

Durch die Erhöhung dieses Werts wird mehr Plattenspeicher für die Protokolle erforderlich, weil die primären Protokolldateien bereits bei der ersten Verbindung zur Datenbank im Voraus zugeordnet werden.

Wenn sich herausstellt, dass häufig sekundäre Protokolldateien zugeordnet werden, kann die Systemleistung möglicherweise durch eine Vergrößerung der Protokolldateien (*logfilsiz*) oder durch eine Erhöhung der Anzahl primärer Protokolldateien verbessert werden.

Bei Datenbanken, auf die nicht oft zugegriffen wird, sollte zur Einsparung von Plattenspeicherplatz dieser Parameter auf den Wert 2 gesetzt werden. Bei Datenbanken, für die die aktualisierende Recovery aktiviert ist, sollte dieser Parameter auf einen größeren Wert gesetzt werden, um den erhöhten Systemaufwand aufgrund der beinahe sofort erforderlichen Zuordnung neuer Protokolle zu vermeiden.

Sie können den Datenbanksystemmonitor zur Ermittlung der geeigneten Größe für die primären Protokolldateien verwenden. Die Beobachtung der folgenden Monitorwerte über einen gewissen Zeitraum hinweg kann sich für die geeignete

Einstellung der Parameter als nützlich erweisen, da Durchschnittswerte Ihre tatsächlichen Anforderungen wahrscheinlich besser widerspiegeln.

- *sec_log_used_top* (Maximum des verwendeten sekundären Protokollspeichers)
- *tot_log_used_top* (Maximum des verwendeten Gesamtprotokollspeichers)
- *sec_logs_allocated* (Momentan zugeordnete sekundäre Protokolle)

logretain - Beibehalten von Protokollen aktivieren

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Dieser Parameter stellt fest, ob aktive Protokolldateien beibehalten werden und für die aktualisierende Recovery verfügbar sind.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Recovery; No]

Es gibt folgende Werte:

- ″No″ gibt an, dass die Protokolle nicht beibehalten werden.
- ″Recovery″ gibt an, dass die Protokolle beibehalten werden und zur aktualisierenden Recovery verwendet werden können.

Ist **logretain** auf Recovery oder **userexit** auf Yes gesetzt, werden die aktiven Protokolldateien gespeichert und als Online-Archivprotokolldateien in einer aktualisierenden Recovery eingesetzt. Dies wird Protokollierung mit Protokollspeicherung genannt.

Nach dem Setzen von **logretain** auf Recovery und/oder von **userexit** auf Yes müssen Sie ein Gesamtbackup der Datenbank vornehmen. Dieser Status wird durch den Markierungsparameter **backup_pending** angezeigt.

Anmerkung:

logarchmeth1 und **logretain** können beide die aktualisierende Recovery aktivieren. Es sollte jedoch nur eine Methode zurzeit für eine Datenbank aktiviert werden.

Wenn Sie **logarchmeth1** verwenden, dürfen Sie die Konfigurationsparameter **logretain** und **userexit** nicht setzen. Wenn der Konfigurationsparameter **logretain** auf Recovery gesetzt ist, wird der Wert für **logarchmeth1** automatisch auf **logretain** gesetzt.

Es wird empfohlen, **logarchmeth1** (und **logarchmeth2**) anstatt **logretain** und **userexit** zu verwenden, um die Archivprotokollierung und die aktualisierende Recovery zu aktivieren. Die Optionen **logretain** und **userexit** wurden beibehalten, um Benutzer zu unterstützen, die noch nicht auf **logarchmeth1** migriert haben.

logsecond - Anzahl sekundärer Protokolldateien

Dieser Parameter gibt die Anzahl der sekundären Protokolldateien an, die (nach Bedarf) erstellt und für Recoveryprotokolle verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
2 [-1; 0 – 254]

Maßeinheit
Zähler

Zuordnung
Nach Bedarf, wenn *logprimary* nicht ausreichend ist (siehe unten).

Freigabe
Wenn der Datenbankmanager feststellt, dass sie nicht mehr erforderlich sind.

Wenn die primären Protokolldateien voll sind, werden die sekundären Protokolldateien (in der Größe *logfilsiz*) nacheinander so zugeordnet, wie sie benötigt werden, und zwar höchstens so viele, wie durch diesen Parameter festgelegt wird. Wenn mehr sekundäre Protokolldateien erforderlich sind, als von diesem Parameter zugelassen werden, wird ein Fehlercode an die Anwendung zurückgegeben, und die Datenbank wird beendet.

Wenn Sie *logsecond* auf -1 setzen, wird die Datenbank mit unbegrenztem Speicherbereich für aktive Protokolle konfiguriert. Für die Größe oder Anzahl der unvollständigen Transaktionen, die in der Datenbank ausgeführt werden, gibt es keine Begrenzung. Wenn Sie *logsecond* auf den Wert -1 setzen, verwenden Sie weiter die Konfigurationsparameter *logprimary* und *logfilsiz*, um die Anzahl der Protokolldateien anzugeben, die der Datenbankmanager im Pfad für aktive Protokolldateien behalten soll. Wenn der Datenbankmanager Protokolldaten aus einer Protokolldatei lesen muss, die sich nicht im Pfad für aktive Protokolldateien befindet, ruft der Datenbankmanager die Protokolldatei aus dem Archiv in den Pfad für aktive Protokolldateien ab. (Der Datenbankmanager ruft die Dateien in den Überlaufprotokollpfad ab, falls Sie einen solchen Pfad konfiguriert haben.) Nach dem Abrufen der Protokolldatei wird diese Datei vom Datenbankmanager im Pfad für aktive Protokolldateien zwischengespeichert, sodass das Lesen weiterer Protokolldaten aus dieser Datei schneller erfolgen kann. Der Datenbankmanager verwaltet das Abrufen, Zwischenspeichern und Entfernen dieser Protokolldateien nach Bedarf.

Wenn Ihr Protokollpfad eine Roheinheit ist, müssen Sie den Konfigurationsparameter *overflowlogpath* konfigurieren, um *logsecond* auf den Wert -1 zu setzen.

Wenn Sie *logsecond* auf den Wert -1 setzen, besteht für die Größe der UOW (Unit of Work, Arbeitseinheit) oder die Anzahl gleichzeitig ablaufender UOWs keine Begrenzung. Allerdings können Rollbacks (sowohl auf Sicherungspunktebene als auch auf UOW-Ebene) viel Zeit in Anspruch nehmen, da Protokolldateien aus dem Archiv abgerufen werden müssen. Aus demselben Grund kann die Recovery nach einem Systemabsturz ebenfalls sehr zeitintensiv sein. Der Datenbankmanager schreibt eine Nachricht in das Protokoll mit Benachrichtigungen für die System-

verwaltung, um Sie darauf hinzuweisen, wenn die aktuelle Menge aktiver UOWs die primären Protokolldateien überschreitet. Dies weist zugleich darauf hin, dass Rollback- oder Recoveryoperationen nach einem Systemabsturz viel Zeit in Anspruch nehmen können.

Damit der Parameter *logsecond* auf den Wert -1 gesetzt werden kann, muss der Konfigurationsparameter *logarchmeth1* auf einen anderen Wert als OFF oder LOGRETAIN gesetzt sein.

Empfehlung: Verwenden Sie sekundäre Protokolldateien für Datenbanken, die in periodischen Zeitabständen immer wieder große Mengen an Protokollspeicher benötigen. Sekundäre Protokolldateien sollten beispielsweise eingesetzt werden, wenn eine Anwendung, die einmal im Monat ausgeführt wird, mehr Protokollspeicher benötigt, als durch die primären Protokolldateien zur Verfügung gestellt wird. Da sekundäre Protokolldateien keinen permanenten Dateibereich erfordern, sind sie für solche Zwecke gut geeignet.

max_log - Maximale Protokollgröße pro Transaktion

Dieser Parameter gibt an, ob für den Prozentsatz an Protokollspeicherbereich, der von einer Transaktion belegt werden kann, ein Grenzwert definiert wurde und wie dieser lautet.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
0 [0 — 100]

Maßeinheit
Prozent

Wenn der Wert nicht 0 ist, zeigt dieser Parameter den Prozentsatz des Speicherbereichs für primäre Protokolle an, der von einer Transaktion belegt werden kann.

Wird der Wert auf 0 gesetzt, besteht keine Begrenzung des Prozentsatzes des gesamten primären Protokollbereichs, den eine Transaktion belegen kann. Dies war die Funktionsweise von Transaktionen vor Version 8.

maxappls - Maximale Anzahl aktiver Anwendungen

Mit diesem Parameter wird die maximale Anzahl der (sowohl lokalen als auch fernen) Anwendungen angegeben, die gleichzeitig auf eine Datenbank zugreifen können. Da jede Anwendung, die die Verbindung zu einer Datenbank herstellt, die Zuordnung privaten Speichers erforderlich macht, entsteht durch Erhöhung dieses Parameters möglicherweise mehr Speicherbedarf.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Automatic [1 - 60 000]

Maßeinheit
Zähler

Wenn Sie den Parameter *maxappls* auf *automatic* setzen, ermöglichen Sie dadurch eine beliebige Anzahl verbundener Anwendungen. Der Datenbankmanager ordnet die zur Unterstützung neuer Anwendungen erforderlichen Ressourcen dynamisch zu.

Wenn Sie diesen Parameter nicht auf *automatic* setzen wollen, muss der Wert dieses Parameters größer oder gleich der Summe der verbundenen Anwendungen plus der Anzahl eben dieser Anwendungen sein, die sich gleichzeitig im Prozess eines zweiphasigen Commits oder eines Rollbacks befinden können. Addieren Sie zu dieser Summe die vorausgeschätzte Anzahl unbestätigter Transaktionen, die zu einem gegebenen Zeitpunkt vorhanden sein können.

Wenn eine Anwendung versucht, die Verbindung zu einer Datenbank herzustellen, jedoch der Wert des Parameters *maxappls* bereits erreicht wurde, wird an die Anwendung ein Fehler zurückgegeben, dass die maximale Anzahl von Anwendungen bereits mit der Datenbank verbunden ist.

In einer Umgebung mit partitionierten Datenbanken ist dies die maximale Anzahl von Anwendungen, die gleichzeitig auf einer Datenbankpartition aktiv sein können. Dieser Parameter beschränkt die Anzahl aktiver Anwendungen für eine Datenbankpartition auf einem Datenbankpartitionsserver unabhängig davon, ob der Server der Koordinatorknoten für die Anwendung ist oder nicht. Für den Katalogknoten in einer Umgebung mit partitionierten Datenbanken ist ein höherer Wert für *maxappls* erforderlich als für andere Umgebungstypen, weil in der Umgebung mit partitionierten Datenbanken jede Anwendung eine Verbindung zum Katalogknoten benötigt.

Empfehlung: Wenn der Wert dieses Parameters erhöht wird, ohne dass der Wert des Parameters *maxlocks* verringert oder der Wert des Parameters *locklist* erhöht wird, könnte die maximale Anzahl Sperren für die Datenbank (Parameter *locklist*) früher erreicht werden als die maximale Anzahl Anwendungen, was zu ständigen Problemen durch Sperreneskalation führen kann.

Bis zu einem gewissen Grad wird die maximale Anzahl von Anwendungen auch vom Paramater *max_coordagents* bestimmt. Eine Anwendung kann nur dann eine Verbindung zur Datenbank herstellen, wenn eine verfügbare Verbindung (*maxappls*) sowie ein koordinierender Agent (*max_coordagents*) vorhanden sind.

maxfilop - Maximale Anzahl offener Datenbankdateien pro Anwendung

Mit diesem Parameter wird die maximale Anzahl der Dateikennungen angegeben, die für jede einzelne Datenbank geöffnet sein können.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Transaktionsgrenzwert

Standardwert [Bereich]

AIX, Sun, HP und Linux (64 Bit)
61 440 [64 - 61 440]

Linux (32 Bit)
30 720 [64 - 30 720]

Windows (32 Bit)
32 768 [64 - 32 768]

Windows (64 Bit)
65 335 [64 - 65 335]

Maßeinheit
Zähler

Wenn durch das Öffnen einer Datei dieser Wert überschritten wird, werden einige von dieser Datenbank verwendete Dateien geschlossen. Wenn der Wert für den Parameter *maxfilop* zu klein ist, wächst der Systemaufwand für das Öffnen und Schließen von Dateien übermäßig an und kann die Leistung beeinträchtigen.

Sowohl SMS-Tabellenbereiche als auch DMS-Tabellenbereichsdateicontainer werden bei der Interaktion des Datenbankmanagers mit dem Betriebssystem als Dateien behandelt, sodass Dateikennungen für sie erforderlich sind. In der Regel werden von SMS-Tabellenbereichen vergleichsweise mehr Dateien verwendet, als von DMS-Dateitabellenbereichen Container verwendet werden. Daher wird für diesen Parameter ein höherer Wert benötigt, wenn SMS-Tabellenbereiche verwendet werden, als für DMS-Dateitabellenbereiche erforderlich wäre.

Mithilfe dieses Parameters kann außerdem sichergestellt werden, dass die Gesamtzahl der Dateikennungen, die vom Datenbankmanager verwendet werden, den Grenzwert des Betriebssystems nicht überschreitet, indem die Anzahl der Dateikennungen pro Datenbank auf einen bestimmten Wert gesetzt wird. Die tatsächliche Anzahl ist je nach Anzahl der gleichzeitig aktiven Datenbanken unterschiedlich.

maxlocks - Maximale Anzahl von Sperren vor Eskalation

Mit diesem Parameter wird definiert, wie viel Prozent der Sperrenliste eine Anwendung belegen muss, damit der Datenbankmanager eine Sperreneskalation ausführt.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Automatic [1 - 100]

Maßeinheit
Prozent

Eine Sperreneskalation ist der Prozess, durch den jeweils mehrere Zeilensperren für eine Tabelle durch eine Tabellensperre ersetzt werden, um die Anzahl der Sperren in der Liste zu verringern. Wenn die Anzahl der Sperren, die von einer Anwen-

dung aktiviert wurden, diesen Prozentwert der Gesamtgröße der Sperrenliste erreicht, wird für die Sperren dieser Anwendung eine Sperreneskalation ausgeführt. Eine Sperreneskalation wird auch dann ausgeführt, wenn in der Sperrenliste kein weiterer Platz mehr verfügbar ist.

Der Datenbankmanager bestimmt die zu eskalierenden Sperren, indem er die Sperrenliste nach der Anwendung durchsucht und die Tabelle mit den meisten gesperrten Zeilen ermittelt. Wenn nach der Ersetzung dieser Sperren durch eine einzige Tabellensperre der Wert für **maxlocks** nicht mehr überschritten wird, wird die Sperreneskalation beendet. Andernfalls wird die Eskalation fortgesetzt, bis der von dieser Anwendung belegte Prozentsatz der Sperrenliste unter den Wert von **maxlocks** gesunken ist. Das Produkt aus dem Wert des Parameters **maxlocks** und dem Wert des Parameters **maxappls** darf nicht kleiner als 100 sein.

Der Wert AUTOMATIC für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch anzupassen. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherverbraucher verteilt, müssen mindestens zwei Speicherverbraucher für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Der Wert des Parameters **locklist** wird zusammen mit dem Parameter **maxlocks** optimiert, sodass eine Inaktivierung der automatischen Optimierung des Parameters **locklist** automatisch auch eine Inaktivierung der automatischen Optimierung des Parameters **maxlocks** bewirkt. Die Aktivierung der automatischen Optimierung für den Parameter **locklist** führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters **maxlocks**.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **self_tuning_mem** muss auf ON gesetzt sein).

Auf 32-Bit-Plattformen erfordert jede Sperre 48 oder 96 Byte der Sperrenliste, und zwar in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind:

- 96 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, für das keine anderen Sperren aktiv sind.
- 48 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Auf 64-Bit-Plattformen (außer HP-UX/PA-RISC) erfordert jede Sperre 64 oder 128 Byte der Sperrenliste in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind:

- 128 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, für das keine anderen Sperren aktiviert sind.
- 64 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Auf HP-UX/PA-RISC-Plattformen (64-Bit) erfordert jede Sperre 80 oder 160 Byte der Sperrenliste in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind.

Empfehlung: Mit der folgenden Formel können Sie **maxlocks** so einstellen, dass eine Anwendung das Zweifache der durchschnittlichen Anzahl von Sperren aktivieren kann:

```
maxlocks = 2 * 100 / maxappls
```

Dabei wird 2 verwendet, um das Zweifache der durchschnittlichen Anzahl zu erzielen, und 100 ist der höchste zulässige Prozentwert. Wenn Sie nur wenige Anwendungen haben, die gleichzeitig aktiv sind, können Sie alternativ zu der ersten Formel auch die folgende Formel verwenden:

```
maxlocks = 2 * 100 / (durchschnittliche Anzahl der gleichzeitig aktiven
Anwendungen)
```

Bei der Festlegung von **maxlocks** können Sie eine gleichzeitige Verwendung von **locklist** (Größe der Sperrenliste) in Erwägung ziehen. Die maximale Anzahl der Sperren, die eine Anwendung aktivieren kann, bevor eine Sperreneskalation erfolgt, wird wie folgt berechnet:

- `maxlocks * locklist * 4096 / (100 * 48)` auf einem 32-Bit-System
- `maxlocks * locklist * 4096 / (100 * 80)` in einer HP-UX/PA-RISC-Umgebung (64-Bit-System)
- `maxlocks * locklist * 4096 / (100 * 64)` auf anderen 64-Bit-Systemen

Dabei ist 4096 die Anzahl der Byte auf einer Seite, 100 ist der höchste zulässige Prozentwert für **maxlocks**, 48 ist die Anzahl der Byte pro Sperre auf einem 32-Bit-System, 80 die Anzahl der Byte pro Sperre auf einem HP-UX/PA-RISC-System (64-Bit) und 64 die Anzahl der Byte pro Sperre auf anderen 64-Bit-Systemen. Wenn Sie wissen, dass eine Ihrer Anwendungen 1000 Sperren benötigt, und Sie keine Sperreneskalation wünschen, sollten Sie die Werte für **maxlocks** und **locklist** in dieser Formel so wählen, dass das Ergebnis größer als 1000 ist. (Wenn Sie für **maxlocks** 10 und für **locklist** 100 verwenden, ist das Ergebnis der Formel höher als die erforderlichen 1000 Sperren.)

Wenn Sie für **maxlocks** einen zu niedrigen Wert wählen, tritt eine Sperreneskalation auf, obwohl für andere gleichzeitig ausgeführte Anwendungen noch genügend Speicher für Sperren verfügbar ist. Wenn ein zu hoher Wert für **maxlocks** gewählt wird, belegen einige wenige Anwendungen u. U. fast den gesamten Speicher für Sperren, während andere Anwendungen eine Sperreneskalation durchführen müssen. Die Notwendigkeit einer Sperreneskalation geht in diesem Fall zulasten des gleichzeitigen Zugriffs.

Mithilfe des Datenbanksystemmonitors können Sie diesen Konfigurationsparameter verfolgen und seinen Wert optimieren.

min_dec_div_3 - 3 Kommastellen bei Dezimaldivision

Dieser Parameter bietet die Möglichkeit, die Berechnung der Kommastellen bei der Dezimaldivision in SQL rasch zu ändern.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes, No]

Der Datenbankkonfigurationsparameter *min_dec_div_3* ändert die Anzahl der Kommastellen im Ergebnis einer Dezimalrechenoperation, die eine Division umfasst. Der Wert kann auf ″Yes″ oder ″No″ gesetzt werden. Der Standardwert für *min_dec_div_3* ist ″No″.Wenn der Wert des Parameters ″No″ ist, wird die Anzahl der Kommastellen als 31-p+s-s′ berechnet. Wenn der Wert ″Yes″ ist, wird die Anzahl der Kommastellen als MAX(3, 31-p+s-s′) berechnet. Dies führt dazu, dass das Ergebnis einer Dezimaldivision immer mindestens 3 Kommastellen hat. Die Genauigkeit ist immer 31.

Eine Änderung dieses Datenbankkonfigurationsparameters kann Änderungen bei Anwendungen für vorhandene Datenbanken bedingen. Dazu kann es kommen, wenn die Anzahl der Kommastellen bei Ergebnissen von Dezimaldivisionen von der Änderung dieses Datenbankkonfigurationsparameters betroffen wäre. Nachfolgend werden einige mögliche Szenarios beschrieben, die Auswirkungen auf Anwendungen haben können. Betrachten Sie diese Szenarios, bevor Sie den Parameter *min_dec_div_3* auf einem Datenbankserver mit vorhandenen Datenbanken ändern.

- Wenn die Anzahl der Kommastellen im Ergebnis für eine der Sichtspalten geändert wird, könnte eine Sicht, die in einer Umgebung mit einer Einstellung definiert ist, mit SQLCODE -344 fehlschlagen, wenn nach Änderung des Datenbankkonfigurationsparameters auf diese Sicht verwiesen wird. Die Nachricht SQL0344N verweist auf rekursive allgemeine Tabellenausdrücke, wenn der Objektname (erstes Token) jedoch eine Sicht ist, müssen Sie die Sicht löschen und sie erneut erstellen, um diesen Fehler zu vermeiden.
- Das Verhalten eines statischen Pakets ändert sich erst, wenn das Paket implizit oder explizit erneut gebunden wird. Zum Beispiel werden nach einer Änderung des Werts von NO in YES die zusätzlichen Kommastellen in den Ergebnissen u. U. erst berücksichtigt, nachdem das Paket erneut gebunden wurde. Für jedes geänderte statische Paket kann mit einem expliziten REBIND-Befehl eine erneute Bindeoperation erzwungen werden.
- Eine Prüfung auf Integritätsbedingung schließt u. U. einige Werte aus, die zuvor akzeptiert wurden. Solche Zeilen verletzen jetzt die Integritätsbedingung, werden jedoch erst entdeckt, wenn eine der Spalten aktualisiert wird, die in der auf Integritätsbedingung geprüften Zeile enthalten ist, oder die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED verarbeitet wird. Sie können die Prüfung auf eine solche Integritätsbedingung erzwingen, indem Sie eine ALTER TABLE-Anweisung ausführen, um die Integritätsbedingung zu löschen und die Integritätsbedingung dann mit einer ALTER TABLE-Anweisung wieder hinzuzufügen.

Anmerkung: Für *min_dec_div_3* gelten außerdem die folgenden Einschränkungen:

1. Der Befehl `GET DB CFG FOR DBNAME` zeigt die Einstellung *min_dec_div_3* nicht an. Die aktuelle Einstellung lässt sich am besten dadurch ermitteln, dass Sie beobachten, welchen Nebeneffekt das Ergebnis einer Dezimaldivision hat. Betrachten Sie z. B. die folgende Anweisung:

   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```

   Wenn diese Anweisung den SQLCODE-Wert SQL0419N zurückgibt, unterstützt die Datenbank *min_dec_div_3* nicht oder der Parameter ist auf ″No″ gesetzt. Wenn die Anweisung `1,000` zurückgibt, ist *min_dec_div_3* auf ″Yes″ gesetzt.
2. *min_dec_div_3* erscheint nicht in der Liste der Konfigurationsschlüsselwörter, wenn Sie den folgenden Befehl ausführen: `? UPDATE DB CFG`

mincommit - Anzahl der Gruppencommits

Mit diesem Parameter können Sie das Schreiben von Protokollsätzen auf die Platte verzögern, bis eine Mindestanzahl von Commitoperationen durchgeführt wurde; dadurch wird der Systemaufwand für den Datenbankmanager im Zusammenhang mit dem Schreiben von Protokollsätzen verringert.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
1 [1 – 25]

Maßeinheit
Zähler

Diese Verzögerung führt zu einer Leistungssteigerung, wenn mehrere Anwendungen für eine Datenbank ausgeführt und von den Anwendungen viele Commitoperationen innerhalb sehr kurzer Zeit angefordert werden.

Diese Gruppierung von Commits wird nur dann ausgeführt, wenn der Wert dieses Parameters größer als eins und die Anzahl der Anwendungen, die mit der Datenbank verbunden sind, größer oder gleich dem Wert dieses Parameters ist. Wenn die Gruppierung von Commits ausgeführt wird, können Commitanforderungen von Anwendungen so lange zurückgehalten werden, bis entweder eine Sekunde vergangen oder die Anzahl der Commitanforderungen gleich dem Wert dieses Parameters ist.

Dieser Parameter sollte nur in kleinen Schritten, zum Beispiel um den Wert 1, erhöht werden. Darüber hinaus sollten Sie mithilfe von Tests mit mehreren Benutzern sicherstellen, dass die Erhöhung des Werts dieses Parameters die erwarteten Ergebnisse liefert.

Am Wert dieses Parameters vorgenommene Änderungen werden sofort wirksam. Sie brauchen nicht abzuwarten, bis alle Anwendungen von der Datenbank getrennt wurden.

Empfehlung: Erhöhen Sie den Wert dieses Parameters über seinen Standardwert hinaus, wenn mehrere Schreib-/Leseanwendungen regelmäßig gleichzeitige Datenbankcommits anfordern. Dadurch wird eine höhere Effizienz bei E/A-Operationen für Protokolldateien erzielt, da diese Operationen weniger häufig auftreten und bei jeder Operation mehr Protokollsätze geschrieben werden.

Sie könnten auch die Anzahl der Transaktionen pro Sekunde ermitteln und diesen Parameter so anpassen, dass die Höchstanzahl von Transaktionen pro Sekunde (oder ein großer Prozentsatz davon) von diesem Parameter berücksichtigt wird. Durch die Berücksichtigung der Spitzenaktivitäten würde der Systemaufwand für das Schreiben von Protokollsätzen in Phasen mit hohem Transaktionsaufkommen minimiert.

Wenn der Wert des Parameters *mincommit* erhöht wird, kann es zudem notwendig werden, den Wert des Parameters *logbufsz* zu erhöhen, um zu vermeiden, dass ein voller Protokollpuffer eine Schreiboperation während dieser Phasen mit hohem

Transaktionsaufkommen erzwingt. In diesem Fall sollte der Wert für den Parameter *logbufsz* nach folgender Formel festgelegt werden:

```
mincommit * (durchschnittlicher Protokollbereich für eine Transaktion)
```

Sie können den Datenbanksystemmonitor folgendermaßen zur Optimierung des Werts dieses Parameters verwenden:

- Berechnen der Höchstanzahl von Transaktionen pro Sekunde:

  Durch das Erstellen von Monitorstichproben über einen typischen Arbeitstag hinweg können Sie die Phasen mit hohem Transaktionsaufkommen ermitteln. Sie können die Gesamtanzahl der Transaktionen durch Addieren der Werte für folgende Monitorelemente berechnen:

  – *commit_sql_stmts (versuchte COMMIT-Anweisungen)*
  – *rollback_sql_stmts (versuchte ROLLBACK-Anweisungen)*

  Anhand dieser Stichproben und der verfügbaren Zeitmarken können Sie die Anzahl der Transaktionen pro Sekunde berechnen.
- Berechnen des pro Transaktion verwendeten Protokollspeicherbereichs:

  Mithilfe von Stichproben über einen gewissen Zeitraum hinweg und für eine Anzahl von Transaktionen können Sie den durchschnittlich verwendeten Protokollspeicherbereich mit dem folgenden Monitorelement berechnen:

  – *log_space_used (Protokollbereich für UOW)*

mirrorlogpath - Pfad für Protokollspiegelung

Mit diesem Parameter können Sie eine Zeichenfolge von bis zu 242 Byte für den Pfad für die Protokollspiegelung angeben. Diese Zeichenfolge muss auf einen Pfadnamen verweisen, der ein vollständig qualifizierter Pfadname und kein relativer Pfadname ist.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [gültiger Pfad oder gültige Einheit]

Anmerkung: In einer DB2 ESE-Umgebung mit einer einzelnen oder mehreren Partitionen wird die Knotennummer automatisch an den Pfad angefügt. Dadurch wird die Eindeutigkeit des Pfads in Konfigurationen mit mehreren logischen Knoten sichergestellt.

Wenn *mirrorlogpath* konfiguriert ist, erstellt DB2 aktive Protokolldateien sowohl im Protokollpfad als auch im Pfad für die Protokollspiegelung. Alle Protokolldaten werden in beide Pfade geschrieben. Der Pfad für die Protokollspiegelung enthält Kopien der aktiven Protokolldateien, sodass die Datenbank im Fall eines Plattenfehlers oder eines Benutzerfehlers, durch den aktive Protokolldateien in einem der Pfade zerstört werden, weiter funktioniert.

Wenn der Pfad für die Protokollspiegelung geändert wird, können sich im alten Pfad für die Protokollspiegelung immer noch Protokolldateien befinden. Diese Protokolldateien wurden eventuell nicht archiviert, sodass Sie sie möglicherweise manuell archivieren müssen. Wenn Sie zudem für diese Datenbank eine Replikation ausführen, benötigt die Replikation eventuell weiterhin die vor der Protokollpfadänderung vorhandenen Protokolldateien. Wenn der Datenbankkonfigurationsparameter für Benutzerexit (*userexit*) für die Datenbank auf "Yes" gesetzt ist und

wenn alle Protokolldateien entweder automatisch durch DB2 oder von Ihnen selbst manuell archiviert wurden, kann DB2 die Protokolldateien zum Beenden des Replikationsprozesses abrufen. Andernfalls können Sie die Dateien aus dem alten Pfad für die Protokollspiegelung in den neuen Pfad für die Protokollspiegelung kopieren.

Wenn *logpath* oder *newlogpath* eine Roheinheit als Position zum Speichern der Protokolldateien angibt, ist die Protokollspiegelung, die durch *mirrorlogpath* angegeben wird, nicht zulässig. Wenn *logpath* oder *newlogpath* einen Dateipfad als Position zum Speichern von Protokolldateien angibt, ist die Protokollspiegelung zulässig, und *mirrorlogpath* muss ebenfalls einen Dateipfad angeben.

Empfehlung: Wie die Protokolldateien sollten sich auch die Spiegelprotokolldateien auf einem physischen Laufwerk mit nur geringem E/A befinden.

Es wird dringend empfohlen, dass dieser Pfad auf einer anderen Einheit eingerichtet wird, als der primäre Protokollpfad.

Mit dem Datenbanksystemmonitor können Sie die Anzahl der E/A-Operationen für die Datenbankprotokollierung verfolgen.

Die folgenden Datenelemente geben das Volumen der E/A-Aktivitäten für die Datenbankprotokollierung zurück. Sie können ein Tool zur Betriebssystemüberwachung verwenden, um Daten zu anderen Platten-E/A-Aktivitäten zu sammeln. Anschließend können Sie beide Arten von E/A-Aktivitäten vergleichen.

- *log_reads* (Anzahl gelesener Protokollseiten)
- *log_writes* (Anzahl geschriebener Protokollseiten)

multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv

Die Zuordnung aus mehreren Seiten bestehender Dateien erhöht die Leistung beim Einfügen. Sie gilt nur für SMS-Tabellenbereiche. Wenn dieser Parameter aktiviert ist, sind alle SMS-Tabellenbereiche davon betroffen. Es ist keine Auswahl für einzelne SMS-Tabellenbereiche möglich.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Der Standardwert für den Parameter ist ″Yes″, d. h. die Zuordnung aus mehreren Seiten bestehender Dateien ist aktiviert.

Im Anschluss an eine Datenbankerstellung kann dieser Parameter nicht auf ″No″ gesetzt werden. Die Zuordnung mehrseitiger Dateien kann nach der Aktivierung nicht mehr inaktiviert werden. Das Tool db2empfa kann zur Aktivierung der Zuordnung mehrseitiger Dateien für eine Datenbank verwendet werden, für die dieses Merkmal inaktiviert ist.

newlogpath - Datenbankprotokollpfad ändern

Mit diesem Parameter können Sie eine Zeichenfolge mit bis zu 242 Byte angeben, um die Speicherposition der Protokolldateien zu ändern.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [gültiger Pfad oder gültige Einheit]

Die Zeichenfolge kann auf einen Pfadnamen oder auf eine Roheinheit verweisen. Beachten Sie, dass ab DB2 Version 9 die Verwendung von Roheinheiten für die Datenbankprotokollierung veraltet ist. Alternativ zur Protokollierung auf Roheinheiten können Sie die Funktionalität zur direkten Ein-/Ausgabe (Direct IO, DIO) oder zur gleichzeitigen Ein-/Ausgabe (Concurrent IO, CIO) verwenden.

Verweist die Zeichenfolge auf einen Pfadnamen, muss es sich um einen vollständig qualifizierten Pfad handeln und nicht um einen relativen Pfad.

In einer DB2 ESE-Umgebung mit einer einzelnen oder mehreren Partitionen wird die Knotennummer automatisch an den Pfad angefügt. Dadurch wird die Eindeutigkeit des Pfads in Konfigurationen mit mehreren logischen Knoten sichergestellt.

Wenn Sie mit Replikation arbeiten wollen und Ihr Protokollpfad eine Roheinheit ist, muss der Konfigurationsparameter *overflowlogpath* konfiguriert werden.

Geben Sie zum Angeben einer Einheit eine Zeichenfolge an, die vom Betriebssystem als eine Einheit erkannt wird. Beispiel:

- Unter Windows: `\\.\d:` oder `\\.\PhysicalDisk5`

  Anmerkung: Damit Protokolle auf eine Einheit geschrieben werden können, muss Windows Version 4.0 mit Service Pack 3 oder höher installiert sein.
- Auf Linux- und UNIX-Plattformen: `/dev/rdblog8`

Anmerkung: Eine Einheit können Sie nur auf AIX-, Windows 2000-, Windows-, Solaris, HP-UX- und Linux-Plattformen angeben.

Diese Einstellung wird nur dann zum Wert des Parameters *logpath* , wenn die beiden folgenden Bedingungen zutreffen:

- Die Datenbank ist in einem konsistenten Zustand, wie durch den Parameter *database_consistent* angegeben.
- Alle Anwendungen sind von der Datenbank getrennt.

Wenn die erste neue Verbindung zur Datenbank hergestellt wird, versetzt der Datenbankmanager die Protokolle an die neue, von *logpath* angegebene Speicherposition.

Im alten Protokollpfad befinden sich eventuell noch Protokolldateien. Diese Protokolldateien wurden eventuell nicht archiviert. Sie müssen sie möglicherweise manuell archivieren. Wenn Sie zudem für diese Datenbank eine Replikation ausführen, benötigt die Replikation eventuell weiterhin die vor der Protokollpfadänderung vorhandenen Protokolldateien. Wenn der Datenbankkonfigurationsparameter für Benutzerexit (*userexit*) für die Datenbank auf "Yes" gesetzt ist und wenn alle Protokolldateien entweder automatisch durch DB2 oder von Ihnen selbst manuell archiviert wurden, kann DB2 die Protokolldateien zum Beenden des Replikationsprozesses abrufen. Andernfalls können Sie die Dateien aus dem alten Protokollpfad in den neuen Protokollpfad kopieren.

Wenn *logpath* oder *newlogpath* eine Roheinheit als Position zum Speichern der Protokolldateien angibt, ist die Protokollspiegelung, die durch *mirrorlogpath* angegeben wird, nicht zulässig. Wenn *logpath* oder *newlogpath* einen Dateipfad als Position zum Speichern von Protokolldateien angibt, ist die Protokollspiegelung zulässig, und *mirrorlogpath* muss ebenfalls einen Dateipfad angeben.

Empfehlung: Es empfiehlt sich, die Protokolldateien auf einer physischen Platte zu speichern, auf der **kein** großes Volumen an E/A-Operationen auftritt. Sie sollten beispielsweise die Protokolldateien nicht auf derselben Platte wie das Betriebssystem oder umfangreiche Datenbanken speichern. Dadurch werden die Protokolliervorgänge effizient und verursachen nur ein Minimum an Systemaufwand, wie z. B. Warten auf E/A-Operationen.

Mit dem Datenbanksystemmonitor können Sie die Anzahl der E/A-Operationen für die Datenbankprotokollierung verfolgen.

Die Monitorelemente *log_reads* (Anzahl gelesener Protokollseiten) und *log_writes* (Anzahl geschriebener Protokollseiten) geben den auf die Datenbankprotokollierung bezogenen Umfang an E/A-Aktivität zurück. Sie können ein Tool zur Betriebssystemüberwachung verwenden, um Daten zu anderen Platten-E/A-Aktivitäten zu sammeln. Anschließend können Sie beide Arten von E/A-Aktivitäten vergleichen.

Verwenden Sie kein gemeinsam genutztes lokales Dateisystem oder Netzdateisystem als Protokollpfad für die Primärdatenbank und die Bereitschaftsdatenbank in einem DB2 High Availability Disaster Recovery-Datenbankpaar (HADR-Datenbankpaar). Die Primärdatenbank und die Bereitschaftsdatenbank haben jeweils Kopien der Transaktionsprotokolle, wobei die Primärdatenbank Protokolle an die Bereitschaftsdatenbank sendet. Wenn der Protokollpfad sowohl für die Primärdatenbank als auch für die Bereitschaftsdatenbank auf dieselbe physische Position zeigen würde, würden die Primärdatenbank und Bereitschaftsdatenbank dieselben physischen Dateien für ihre jeweiligen Kopien der Protokolle verwenden. Der Datenbankmanager gibt einen Fehler zurück, wenn er einen gemeinsam genutzten Protokollpfad erkennt.

num_db_backups - Anzahl der Datenbank-Backups

Dieser Parameter gibt die Anzahl der Datenbank-Backups an, die für eine Datenbank beibehalten werden sollen.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Transaktionsgrenzwert

Standardwert [Bereich]
12 [1 - 32 767]

Wird die angegebene Anzahl von Backups erreicht, werden alte Backups in der Datei des Recoveryprotokolls als abgelaufen markiert. Die Einträge in der Datei des Recoveryprotokolls für Tabellenbereichs-Backups und Ladekopie-Backups, die mit dem abgelaufenen Datenbank-Backup verbunden sind, werden ebenfalls als abgelaufen markiert. Wird ein Backup als abgelaufen markiert, können die physi-

schen Backups von ihrem Speicherort (z. B. Platte, Band, TSM) entfernt werden. Das nächste Datenbank-Backup entfernt die abgelaufenen Einträge aus der Datei des Recoveryprotokolls.

Der Konfigurationsparameter *rec_his_retentn* sollte auf einen Wert gesetzt werden, der mit dem Wert von *num_db_backups* kompatibel ist. Wenn *num_db_backup* z. B. auf einen hohen Wert gesetzt ist, muss der Wert für *rec_his_retentn* hoch genug sein, um diese Anzahl der Backups unterstützen zu können.

num_freqvalues - Anzahl der häufigsten Werte

Mit diesem Parameter können Sie die Anzahl der „häufigsten Werte" angeben, die gesammelt werden, wenn die Option WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [0 - 32 767]

Maßeinheit
Zähler

Wenn der Wert dieses Parameters erhöht wird, erhöht sich auch die Menge an Statistikzwischenspeicher (*stat_heap_sz*), die zum Sammeln statistischer Daten benötigt wird.

Die Statistik der „häufigsten Werte" unterstützt das Optimierungsprogramm beim Feststellen der Verteilung der Datenwerte innerhalb einer Spalte. Wenn der Wert erhöht wird, stehen dem Abfrageoptimierungsprogramm mehr Daten zur Verfügung. Allerdings ist zusätzlicher Katalogspeicher erforderlich. Wenn 0 angegeben wird, wird keine Statistik über häufige Werte erhoben, auch wenn Sie anfordern, dass statistische Informationen zur Datenverteilung gesammelt werden.

Sie können die Anzahl der häufigsten Werte, die auf der Tabellen- oder Spaltenebene durch den Befehl RUNSTATS erfasst werden, auch mithilfe der Option NUM_FREQVALUES angeben. Wenn kein Wert angegeben wird, wird der Wert des Konfigurationsparameters *num_freqvalues* verwendet. Es ist einfacher, die Anzahl der häufigsten Werte über den Befehl RUNSTATS zu ändern, als über den Datenbankkonfigurationsparameter *num_freqvalues*.

Durch Aktualisieren dieses Parameters kann das Optimierungsprogramm bessere Schätzwerte für die Auswahl von nicht gleichmäßig verteilten Daten mithilfe einiger Vergleichselemente (=, <, >) erzielen. Exaktere Berechnungen der Selektivität können zur Auswahl effizienterer Zugriffspläne führen.

Nach der Änderung dieses Parameters müssen Sie wie folgt vorgehen:

- Führen Sie den Befehl RUNSTATS erneut aus, um Statistikdaten mit der geänderten Anzahl der häufigsten Werte zu sammeln.
- Führen Sie für alle Pakete mit statischen SQL- oder XQuery-Anweisungen einen Rebind durch.

Beim Verwenden des Befehls RUNSTATS haben Sie die Möglichkeit, die Anzahl der häufigsten Werte zu begrenzen, die auf Tabellenebene oder auf Spaltenebene gesammelt werden. Dadurch können Sie den in den Katalogen belegten Speicherbereich optimieren, indem Sie die Verteilungsstatistik für Spalten reduzieren, für die sie nicht genutzt werden kann, die Informationen für kritische Spalten jedoch weiterhin verwenden.

Empfehlung: Zur Aktualisierung dieses Parameters sollten Sie den Grad der Ungleichmäßigkeit der Daten in den wichtigsten Spalten (in den wichtigsten Tabellen) feststellen, für die in der Regel Auswahlvergleichselemente angegeben werden. Dies kann mithilfe einer SQL-Anweisung SELECT geschehen, die eine Rangfolge des Vorkommens jedes Werts in einer Spalte liefert. Dabei dürfen Sie gleichmäßig verteilte, eindeutige, LONG- oder LOB-Spalten nicht berücksichtigen. Ein geeigneter praktischer Wert für diesen Parameter liegt im Bereich zwischen 10 und 100.

Beachten Sie, dass das Sammeln statistischer Daten über die häufigsten Werte erhebliche CPU- und Speicherressourcen (*stat_heap_sz*) erfordert.

num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen

Mit diesem Parameter können Sie die Anzahl asynchroner Seitenlöschfunktionen für eine Datenbank angeben.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Automatic [0 – 255]

Maßeinheit
Zähler

Diese Seitenlöschfunktionen schreiben geänderte Seiten aus dem Pufferpool auf Platte, bevor der Bereich im Pufferpool von einem Datenbankagenten angefordert wird. Daher müssen Datenbankagenten in der Regel nicht die Auslagerung geänderter Seiten abwarten, bevor sie den Speicherbereich im Pufferpool nutzen können. Dadurch wird die Gesamtleistung der Datenbankanwendungen verbessert.

Wenn der Parameter auf den Wert 0 gesetzt wird, werden keine Seitenlöschfunktionen gestartet, und infolgedessen schreiben die Agenten alle geänderten Seiten aus dem Pufferpool auf Platte. Dieser Parameter kann erhebliche Auswirkungen auf die Leistung einer Datenbank haben, die auf mehrere physische Speichereinheiten verteilt ist, weil in diesem Fall die Wahrscheinlichkeit größer ist, dass auf einer dieser Einheiten momentan keine Operationen ausgeführt werden. Wenn keine Seitenlöschfunktionen konfiguriert sind, könnten Ihre Anwendungen in regelmäßigen Abständen auf volle Protokolle stoßen.

Wenn dieser Parameter auf AUTOMATIC gesetzt wird, richtet sich die Anzahl der gestarteten Seitenlöschfunktionen nach der Anzahl der auf dem aktuellen System konfigurierten CPUs sowie nach der Anzahl lokaler logischer Datenbankpartitionen in einer Umgebung mit partitionierten Datenbanken. Es wird in jedem Fall mindestens eine Seitenlöschfunktion gestartet, wenn dieser Parameter auf AUTOMATIC gesetzt ist.

Die Anzahl der zu startenden Seitenlöschfunktionen, wenn dieser Parameter auf AUTOMATIC gesetzt ist, wird anhand der folgenden Formel berechnet:

```
Anzahl Seitenlöschfunktionen = max( ceil( Anzahl CPUs / Anzahl lokaler logischer DBpartitionen ) - 1
```

Diese Formel stellt sicher, dass die Anzahl der Seitenlöschfunktionen nahezu gleichmäßig auf Ihre logischen Datenbankpartitionen verteilt wird und dass nicht mehr Seitenlöschfunktionen als CPUs vorhanden sind.

Wenn die Anwendungen für eine Datenbank im Wesentlichen aus Transaktionen bestehen, mit denen die Daten aktualisiert werden, führt eine Erhöhung der Anzahl der Seitenlöschfunktionen zu einer Leistungsverbesserung. Durch die Erhöhung der Anzahl der Seitenlöschfunktionen wird außerdem die Zeit für Recoverys nach Systemausfällen, zum Beispiel aufgrund von Netzausfall, verringert, weil der Inhalt der Datenbank auf der Platte zu einem gegebenen Zeitpunkt aktueller ist.

Empfehlung: Bei der Einstellung dieses Parameters müssen folgende Faktoren beachtet werden:

- Anwendungsart
  - Wenn es sich um eine reine Abfragedatenbank handelt, die nicht aktualisiert wird, setzen Sie diesen Parameter auf den Wert Null (0). Eine Ausnahme hiervon wäre, wenn die Abfrageauslastung dazu führt, dass viele TEMP-Tabellen erstellt werden. (Verwenden Sie das Dienstprogramm EXPLAIN, um dies festzustellen.)
  - Wenn Transaktionen für die Datenbank ausgeführt werden, setzen Sie diesen Parameter auf einen Wert zwischen 1 und der Anzahl der physischen Speichereinheiten, die für diese Datenbank verwendet werden.
- Auslastung

  Umgebungen mit hohem Aufkommen an aktualisierenden Transaktionen können eventuell die Konfiguration weiterer Seitenlöschfunktionen erforderlich machen.
- Pufferpoolgrößen

  Umgebungen mit großen Pufferpools können eventuell auch die Konfiguration weiterer Seitenlöschfunktionen erforderlich machen.

Mithilfe des Datenbanksystemmonitors können Sie diesen Konfigurationsparameter optimieren, indem Sie die Informationen des Ereignismonitors zu Schreibaktivitäten aus einem Pufferpool heranziehen:

- Der Wert des Parameters kann verringert werden, wenn die beiden folgenden Bedingungen erfüllt sind:
  - *pool_data_writes* ist ungefähr gleich *pool_async_data_writes*.
  - *pool_index_writes* ist ungefähr gleich *pool_async_index_writes*.
- Der Wert des Parameters sollte erhöht werden, wenn eine der folgenden Bedingungen erfüllt ist:
  - *pool_data_writes* ist viel größer als *pool_async_data_writes*.
  - *pool_index_writes* ist viel größer als *pool_async_index_writes*.

num_ioservers - Anzahl von E/A-Servern

Mit diesem Parameter wird die Anzahl der E/A-Server für eine Datenbank definiert. Zu jedem beliebigen Zeitpunkt kann nur diese Anzahl von E/A-Servern zum Vorablesen und für Dienstprogramme für eine Datenbank aktiv sein.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Automatic [1 – 255]

Maßeinheit
Zähler

Zuordnung
Wenn eine Anwendung die Verbindung zu einer Datenbank herstellt

Freigabe
Wenn eine Anwendung die Verbindung zu einer Datenbank trennt

E/A-Server, die auch als Vorablesefunktionen bezeichnet werden, werden für Datenbankagenten verwendet, um Vorablese-E/A-Operationen und asynchrone E/A-Operationen durch Dienstprogramme wie BACKUP und RESTORE auszuführen. Ein E/A-Server wartet, während eine vom ihm eingeleitete E/A-Operation ausgeführt wird. Nicht vorabgelesene Ein-/Ausgaben werden direkt von den Datenbankagenten terminiert, sodass diese Ein-/Ausgaben nicht der Begrenzung durch den Parameter *num_ioservers* unterliegen.

Wenn dieser Parameter auf AUTOMATIC gesetzt wird, richtet sich die Anzahl der gestarteten Vorablesefunktionen nach den Einstellungen für die Parallelität der Tabellenbereiche in der aktuellen Datenbankpartition. (Die Einstellungen für die Parallelität werden mithilfe der Umgebungsvariablen DB2_PARALLEL_IO gesteuert.) Für jeden DMS-Tabellenbereich wird der Wert dieser Parallelitätseinstellung mit der maximalen Anzahl von Containern im Stripe-Set des Tabellenbereichs multipliziert. Für jeden SMS-Tabellenbereich wird der Wert dieser Parallelitätseinstellung mit der Anzahl von Containern im Tabellenbereich multipliziert. Das höchste Ergebnis aus allen Tabellenbereichen in der aktuellen Datenbankpartition wird als Anzahl der zu startenden Vorablesefunktionen verwendet. Es werden in jedem Fall mindestens drei Vorablesefunktionen gestartet, wenn dieser Parameter auf AUTOMATIC gesetzt ist.

Wenn dieser Parameter auf AUTOMATIC gesetzt ist, wird die Anzahl der bei der Aktivierung der Datenbank zu startenden Vorablesefunktionen anhand der folgenden Formel berechnet:

```
Anzahl Vorablesefunktionen = max( max über alle Tabellenbereiche
( Parallelitätseinstellung * [SMS: Anz. Container; DMS: max. Anz. Container
im Stripe-Set] ), 3 )
```

Empfehlung: Um alle E/A-Einheiten des Systems voll auszunutzen, empfiehlt sich im Allgemeinen ein Wert, der um 1 oder 2 höher ist als die Anzahl der physischen Einheiten, auf denen sich die Datenbank befindet. Es ist besser, zusätzliche E/A-Server zu konfigurieren, da mit jedem E/A-Server nur geringfügiger Systemaufwand verbunden ist und alle nicht benötigten E/A-Server inaktiv bleiben.

num_log_span - Anzahl verwendeter Protokolldateien

Dieser Parameter gibt an, ob für die Menge an Protokolldateien, die eine Transaktion umfassen kann, ein Grenzwert definiert wurde und wie dieser lautet.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
0 [0 - 65 535]

Maßeinheit
Zähler

Wenn der Wert nicht 0 ist, zeigt dieser Parameter die Anzahl aktiver Protokolldateien an, über die sich eine aktive Transaktion erstrecken kann.

Wird der Wert auf 0 gesetzt, besteht keine Begrenzung für die Anzahl der Protokolldateien, die eine einzelne Transaktion umfassen kann. Dies war die Funktionsweise von Transaktionen vor Version 8.

num_quantiles - Anzahl der Quantile für Spalten

Mit diesem Parameter wird die Anzahl der Quantile gesteuert, die gesammelt werden, wenn die Option WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
20 [0 - 32 767]

Maßeinheit
Zähler

Wenn der Wert dieses Parameters erhöht wird, erhöht sich auch die Menge an Statistikzwischenspeicher (*stat_heap_sz*), die zum Sammeln statistischer Daten benötigt wird.

Die Statistik der „Quantile" unterstützt das Optimierungsprogramm beim Feststellen der Verteilung der Datenwerte innerhalb einer Spalte. Wenn der Wert erhöht wird, stehen dem Abfrageoptimierungsprogramm mehr Daten zur Verfügung; allerdings ist zusätzlicher Katalogspeicher erforderlich. Wenn die Werte 0 oder 1 angegeben werden, werden keine Quantil-Statistikdaten erhoben, auch wenn Sie anfordern, dass Verteilungsstatistikdaten gesammelt werden.

Sie können die Anzahl der Quantile, die auf der Tabellen- oder Spaltenebene durch den Befehl RUNSTATS erfasst werden, auch mit der Option NUM_QUANTILES angeben. Wenn kein Wert angegeben wird, wird der Wert des Konfigurationsparameters *num_quantiles* verwendet. Es ist einfacher, die Anzahl der zu erfassenden Quantile über den Befehl RUNSTATS zu ändern, als über den Datenbankkonfigurationsparameter *num_quantiles*.

Durch Aktualisieren dieses Parameters können bessere Schätzwerte für die Auswahl von nicht gleichmäßig verteilten Daten mithilfe von Bereichsvergleichselementen erzielt werden. Unter anderem entscheidet das Optimierungsprogramm mithilfe dieser Informationen, ob eine Indexsuche oder eine Tabellensuche gewählt wird. (Beim Zugriff auf einen Bereich von Werten, die häufig vorkommen, ist eine Tabellensuche effizienter, während bei einem Bereich von Werten, die nicht häufig vorkommen, eine Indexsuche effizienter ist.)

Nach der Änderung dieses Parameters müssen Sie wie folgt vorgehen:

- Führen Sie den Befehl RUNSTATS erneut aus, um Statistikdaten mit der geänderten Anzahl der Quantile zu sammeln.
- Führen Sie für alle Pakete mit statischen SQL- oder XQuery-Anweisungen einen Rebind durch.

Beim Verwenden des Befehls RUNSTATS haben Sie die Möglichkeit, die Anzahl der erfassten Quantile sowohl auf Tabellenebene als auch auf Spaltenebene zu begrenzen. Dadurch können Sie den in den Katalogen belegten Speicherbereich optimieren, indem Sie die Verteilungsstatistik für Spalten reduzieren, in denen Sie nicht genutzt werden kann, und die Informationen jedoch weiter für kritische Spalten verwenden.

Empfehlung: Der Standardwert für diesen Parameter garantiert einen maximalen Schätzfehler von ungefähr 2,5 % für alle einseitigen Bereichsvergleichselemente (>, >=, < oder <=) und einen maximalen Fehler von 5 % für jedes BETWEEN-Vergleichselement. Nachfolgend eine einfache Möglichkeit, um die Anzahl der Quantile einzugrenzen:

- Bestimmen Sie den maximalen Fehler, der bei der Schätzung der Anzahl von Zeilen jeder Bereichsabfrage tolerierbar ist, als Prozentwert P.
- Die Anzahl der Quantile sollte ca. 100/P sein, wenn die meisten Ihrer Vergleichselemente BETWEEN-Vergleichselemente sind, und 50/P, wenn die meisten Ihrer Vergleichselemente andere Arten von Bereichsvergleichselementen (<, <=, > oder >=) sind.

Zum Beispiel ergeben 25 Quantile einen maximalen Schätzfehler von 4 % bei BETWEEN-Vergleichselementen und 2 % bei Vergleichselementen mit ″>″. Ein geeigneter praktischer Wert für diesen Parameter liegt im Bereich zwischen 10 und 50.

numarchretry - Anzahl der Wiederholungen bei Fehler

Mit diesem Parameter wird die Anzahl der Versuche angegeben, die DB2 zur Archivierung einer Protokolldatei im primären oder sekundären Archivverzeichnis unternehmen soll, bevor es versucht, Protokolldateien im Verzeichnis für die Funktionsübernahme zu speichern.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
5 [0 - 65 535]

Dieser Parameter wird nur verwendet, wenn der Datenbankkonfigurationsparameter *failarchpath* definiert ist. Wenn *numarchretry* nicht definiert wird, setzt DB2 die Versuche zur Archivierung im primären oder sekundären Protokollpfad kontinuierlich fort.

numsegs - Standardanzahl von SMS-Containern

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Maßeinheit
Zähler

Dieser Parameter gibt die Anzahl der Container an, die innerhalb der Standardtabellenbereiche erstellt werden. Er zeigt auch die Informationen an, die bei der Erstellung der Datenbank verwendet wurden, unabhängig davon, ob die Angabe im Befehl CREATE DATABASE explizit oder implizit war.

Dieser Parameter gilt nur für SMS-Tabellenbereiche. Er wird von der Anweisung CREATE TABLESPACE **nicht** verwendet.

overflowlogpath - Überlaufprotokollpfad

Dieser Parameter gibt eine Speicherposition an, an der DB2 nach Protokolldateien suchen soll, die für eine ROLLFORWARD-Operation benötigt werden; außerdem gibt er eine Speicherposition für die aktiven Protokolldateien an, die aus dem Archiv abgerufen werden. Er gibt auch eine Speicherposition zum Suchen nach und Speichern von Protokolldateien an, die für die Verwendung der API db2ReadLog erforderlich sind.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiger gültiger Pfad]

Je nach Ihren Protokollierungsanforderungen kann dieser Parameter für verschiedene Funktionen verwendet werden.

- Mit diesem Parameter können Sie eine Speicherposition angeben, an der DB2 nach Protokolldateien suchen soll, die für eine ROLLFORWARD-Operation benötigt werden. Dies bietet eine Alternative zur Verwendung der Option OVERFLOW LOG PATH mit dem ROLLFORWARD-Befehl. Anstatt jedes Mal die Option OVERFLOW LOG PATH mit dem ROLLFORWARD-Befehl anzugeben, können Sie einmal diesen Konfigurationsparameter festlegen. Wenn allerdings beides verwendet wird, überschreibt die Option OVERFLOW LOG PATH den Konfigurationsparameter *overflowlogpath* für diese bestimmte ROLLFORWARD-Operation.

- Wenn *logsecond* auf -1 gesetzt ist, können Sie über den Parameter *overflowlogpath* ein Verzeichnis angeben, in dem DB2 aus dem Archiv abgerufene aktive Protokolldateien speichert. (Aktive Protokolldateien müssen für Rollbacks abgerufen werden, wenn sie sich nicht mehr im Pfad für aktive Protokolldateien befinden). Wird für 'overflowlogpath' kein Wert angegeben, ruft DB2 die Protokolldateien in den Pfad für aktive Protokolldateien ab. Mit dem Parameter *overflowlogpath* können Sie DB2 zusätzliche Ressourcen zum Speichern von abgerufenen Protokolldateien bereitstellen. Dies bietet den Vorteil, den Ein-/Ausgabeaufwand auf verschiedene Datenträger zu verteilen und mehr Protokolldateien im Pfad für aktive Protokolldateien speichern zu können.
- Wenn Sie beispielsweise zur Replikation die API db2ReadLog (vor DB2 Version 8 hatte db2ReadLog den Namen sqlurlog) benutzen müssen, können Sie beispielsweise mit dem Parameter *overflowlogpath* eine Position angeben, an der DB2 nach Protokolldateien sucht, die für diese API benötigt werden. Wenn die Protokolldatei nicht gefunden wird (weder im Pfad für aktive Protokolldateien noch im Überlaufprotokollpfad) und die Datenbank mit dem aktivierten Parameter *userexit* konfiguriert ist, ruft DB2 die Protokolldatei ab. Über den Parameter *overflowlogpath* können Sie auch ein Verzeichnis angeben, in dem DB2 die abgerufenen Protokolldateien speichert. Dadurch wird der Ein-/Ausgabeaufwand für den Pfad für aktive Protokolldateien reduziert, und es können mehr Protokolldateien im Pfad für aktive Protokolldateien gespeichert werden.
- Wenn Sie als Pfad für aktive Protokolldateien eine unformatierte Einheit (Roheinheit) konfiguriert haben, muss *overflowlogpath* konfiguriert werden, wenn Sie *logsecond* auf -1 setzen oder die API db2ReadLog verwenden wollen.

Zum Definieren von *overflowlogpath* müssen Sie eine Zeichenfolge mit maximal 242 Byte angeben. Diese Zeichenfolge muss auf einen Pfadnamen verweisen, der ein vollständig qualifizierter Pfadname und kein relativer Pfadname ist. Der Pfadname muss ein Verzeichnis sein, keine Roheinheit.

Anmerkung: In einer DB2 ESE-Umgebung mit einer einzelnen oder mehreren Partitionen wird die Knotennummer automatisch an den Pfad angefügt. Dadurch wird die Eindeutigkeit des Pfads in Konfigurationen mit mehreren logischen Knoten sichergestellt.

pagesize - Standardseitengröße für die Datenbank

Dieser Parameter enthält den Wert, der als Standardseitengröße bei der Erstellung der Datenbank verwendet wurde. Gültige Werte: 4 096, 8 192, 16 384 und 32 768. Wenn in dieser Datenbank ein Pufferpool oder ein Tabellenbereich erstellt wird, wird dieselbe Standardseitengröße verwendet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

pckcachesz - Größe des Paketcache

Dieser Parameter wird aus dem gemeinsam benutzten Datenbankspeicher zugeordnet und für das Caching (Zwischenspeichern) von Abschnitten statischer und dynamischer SQL- und XQuery-Anweisungen in einer Datenbank verwendet.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
Automatic [-1, 32 - 128 000]

64-Bit-Plattformen
Automatic [-1, 32 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank initialisiert wird

Freigabe
Wenn die Datenbank heruntergefahren wird

In einem partitionierten Datenbanksystem gibt es für jede Datenbankpartition einen Paketcache.

Das Caching von Paketen ermöglicht dem Datenbankmanager, den internen Systemaufwand zu verringern, da beim erneuten Laden eines Pakets kein Zugriff auf die Systemkataloge bzw. bei dynamischen SQL- oder XQuery-Anweisungen keine Kompilierung mehr erforderlich ist. Die Abschnitte werden im Paketcache behalten, bis einer der folgenden Umstände eintritt:

- Die Datenbank wird heruntergefahren.
- Das Paket oder die dynamische SQL- oder XQuery-Anweisung wird ungültig gemacht.
- Im Cache ist nicht mehr genügend Platz.

Dieses Caching des Abschnitts für eine statische oder dynamische SQL- oder XQuery-Anweisung kann die Leistung besonders dann verbessern, wenn dieselbe Anweisung mehrere Male von Anwendungen verwendet wird, die mit einer Datenbank verbunden sind. Dies ist insbesondere für eine Umgebung wichtig, die Transaktionen verarbeitet.

Der Wert AUTOMATIC für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Wenn der Parameter *self_tuning_mem* auf ON gesetzt ist, passt die Speicheroptimierungsfunktion die durch den Parameter *pckcachesz* gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch an. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherverbraucher verteilt, müssen mindestens zwei Speicherverbraucher für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter *self_tuning_mem* muss auf "ON" gesetzt sein).

Wenn dieser Parameter auf den Wert -1 gesetzt ist, wird als Wert für die Berechnung der Seitenzuordnung das Achtfache des für den Konfigurationsparameter

maxappls angegebenen Werts verwendet. Dies gilt jedoch nicht, wenn das Achtfache von *maxappls* kleiner als 32 ist. In diesem Fall entspricht der Standardwert -1 für *pckcachesz* dem Wert 32.

Empfehlung: Beim Optimieren dieses Parameters sollten Sie überlegen, ob der zusätzliche Speicher, der für den Paketcache reserviert wird, nicht günstiger für einen anderen Zweck zugeordnet werden könnte, z. B. für den Pufferpool oder einen Katalogcache. Aus diesem Grund sollten Sie zur Optimierung dieses Parameters Vergleichstests (Benchmark-Tests) durchführen.

Die optimale Einstellung dieses Parameters ist von besonderer Bedeutung, wenn zu Beginn mehrere Abschnitte verwendet werden und nur wenige Abschnitte wiederholt ausgeführt werden. Wenn der Cache zu groß ist, wird Speicher zum Behalten von Kopien der Anfangsabschnitte verschwendet.

Mithilfe der folgenden Monitorelemente können Sie ermitteln, ob Sie den Wert dieses Konfigurationsparameters anpassen sollten:

- *pkg_cache_lookups* (Zugriffe auf Paketcache)
- *pkg_cache_inserts* (Einfügungen in Paketcache)
- *pkg_cache_size_top* (obere Paketcachegrenze)
- *pkg_cache_num_overflows* (Überläufe des Paketcache)

Anmerkung: Der Paketcache ist ein Arbeitscache. Deshalb kann dieser Parameter nicht auf den Wert 0 gesetzt werden. Diesem Cache muss ausreichend Speicherplatz für alle Abschnitte der SQL- oder XQuery-Anweisungen zugeordnet sein, die momentan ausgeführt werden. Wenn mehr als der momentan benötigte Speicherplatz zugeordnet ist, werden Abschnitte zwischengespeichert. Diese Abschnitte können einfach ausgeführt werden, wenn sie das nächste Mal benötigt werden, und müssen nicht erneut geladen oder kompiliert werden.

Der durch den Parameter *pckcachesz* angegebene Grenzwert ist ein veränderlicher Grenzwert. Dieser Grenzwert kann, falls erforderlich, überschritten werden, wenn in dem von den Datenbanken gemeinsam benutzten Speicher noch Speicherplatz verfügbar ist. Sie können mit dem Monitorelement *pkg_cache_size_top* den Höchstwert ermitteln, bis zu dem der Paketcache angewachsen ist. Mit dem Monitorelement *pkg_cache_num_overflows* können Sie ermitteln, wie häufig der durch den Parameter *pckcachesz* angegebene Grenzwert überschritten wurde.

priv_mem_thresh - Schwellenwert für privaten Speicher

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
20 000 [-1; 32 - 112 000]

Maßeinheit
Seiten (4 KB)

Dieser Parameter wird zur Festlegung des nicht benutzten privaten Agentenspeichers verwendet, der zugeordnet bleibt und zur Verwendung durch neue Agenten, die gestartet werden, zur Verfügung steht. Er gilt nicht für Linux- und UNIX-Plattformen.

Der Wert -1 bewirkt, dass dieser Parameter den Wert des Parameters *min_priv_mem* verwendet.

Empfehlung: Bei der Einstellung dieses Parameters sollten die Abläufe, wann und wie Clients die Verbindung herstellen bzw. trennen, sowie der Speicherbedarf anderer Prozesse auf derselben Maschine berücksichtigt werden.

Wenn es nur eine kurze Phase gibt, während der viele Clients gleichzeitig mit der Datenbank verbunden sind, verhindert ein hoher Schwellenwert, dass ungenutzter Speicher freigegeben und für andere Prozesse verfügbar gemacht wird. Dies führt zu einer schlechten Speicherverwaltung, die andere Prozesse, für die Speicher erforderlich ist, beeinträchtigen kann.

Wenn die Anzahl gleichzeitig zugreifender Clients eher gleichmäßig hoch ist, aber zahlreiche Änderungen an dieser Zahl auftreten, stellt ein hoher Schwellenwert sicher, dass Speicher für die Clientprozesse verfügbar ist, und verringert so den Systemaufwand, der durch das Zuordnen und Freigeben von Speicher entsteht.

rec_his_retentn - Aufbewahrungszeitraum für Recoveryprotokoll

Mit diesem Parameter wird angegeben, wie viele Tage die Protokolldaten zu Backups aufbewahrt werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
366 [-1; 0 - 30 000]

Maßeinheit
Tage

Wenn die Datei des Recoveryprotokolls nicht benötigt wird, um Backups, Restores und Ladevorgänge festzuhalten, kann dieser Parameter auf einen kleineren Wert gesetzt werden.

Wenn der Wert dieses Parameters -1 lautet, entspricht die Anzahl der Einträge, die Datenbankgesamtbackups anzeigen (sowie alle Tabellenbereichs-Backups, die dem Datenbank-Backup zugeordnet sind) dem Wert, der durch den Parameter *num_db_backups* angegeben ist. Weitere Einträge in der Datei des Recoveryprotokolls können nur explizit mithilfe der verfügbaren Befehle oder APIs entfernt werden.

Unabhängig davon, wie kurz der Aufbewahrungszeitraum ist, werden das aktuelle Datenbank-Backup und die zugehörige Restoregruppe immer zurückbehalten, sofern Sie nicht das Dienstprogramm PRUNE mit der Angabe WITH FORCE OPTION verwenden.

restore_pending - Restore anstehend

Dieser Parameter gibt an, ob sich die Datenbank im Status Restore anstehend befindet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

restrict_access - Eingeschränkter Datenbankzugriff (Konfigurationsparameter)

Dieser Parameter gibt an, ob die Datenbank mit dem eingeschränkten Satz an Standardaktionen erstellt wurde. Das heißt, ob sie mit der Klausel RESTRICTIVE im Befehl CREATE DATABASE erstellt wurde.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

YES: Die Klausel RESTRICTIVE wurde bei der Erstellung dieser Datenbank im Befehl CREATE DATABASE verwendet.

NO: Die Klausel RESTRICTIVE wurde bei der Erstellung dieser Datenbank im Befehl CREATE DATABASE nicht verwendet.

rollfwd_pending - Aktualisierende Recovery anstehend

Über diesen Parameter werden Sie darüber informiert, ob eine aktualisierende Recovery erforderlich ist und wo sie erforderlich ist.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter kann einen der folgenden Status anzeigen:

- **DATABASE**, d. h. eine aktualisierende Recovery ist für diese Datenbank erforderlich
- **TABLESPACE**, d. h. für mindestens einen Tabellenbereich muss eine aktualisierende Recovery durchgeführt werden
- **NO**, d. h. die Datenbank ist verwendbar und keine aktualisierende Recovery erforderlich

Die Recovery (mit ROLLFORWARD DATABASE) muss erfolgreich beendet sein, bevor auf die Datenbank bzw. den Tabellenbereich zugegriffen werden kann.

self_tuning_mem - Speicher mit automatischer Leistungsoptimierung

Dieser Parameter legt fest, ob die Speicheroptimierungsfunktion verfügbare Speicherressourcen bei Bedarf unter den Speicherkonsumenten, für die die automatische Optimierung aktiviert ist, dynamisch verteilt.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

Umgebungen mit einer Einzeldatenbankpartition
ON [ON; OFF]

Umgebungen mit mehreren Datenbankpartitionen
OFF [ON; OFF]

In einer Datenbank, die von einer früheren Version migriert wurde, ist der Parameter *self_tuning_mem* auf den Wert OFF gesetzt.

Da der Speicher zwischen Speicherkonsumenten aufgeteilt wird, müssen mindestens zwei Speicherkonsumenten für die automatische Optimierung aktiviert sein, damit die Speicheroptimierungsfunktion aktiviert werden kann. Wenn der Parameter *self_tuning_mem* auf den Wert ON gesetzt ist, jedoch weniger als zwei Speicherkonsumenten für die automatische Optimierung aktiviert sind, bleibt die Speicheroptimierungsfunktion inaktiv. (Eine Ausnahme bildet der Speicherbereich des Sortierzwischenspeichers, der unabhängig davon optimiert werden kann, ob andere Speicherkonsumenten für die automatische Optimierung aktiviert sind oder nicht.) Wenn der Parameter *database_memory* auf einen numerischen Wert gesetzt ist, wird er als für die automatische Optimierung aktiviert betrachtet.

Dieser Parameter hat in Umgebungen mit einer Einzeldatenbankpartition standardmäßig den Wert ON. In Umgebungen mit mehreren Datenbankpartitionen ist er standardmäßig auf OFF gesetzt.

Die folgenden Speicherkonsumenten können für die automatische Optimierung aktiviert werden:

- Pufferpools (Steuerung durch den Parameter SIZE der Anweisungen ALTER BUFFERPOOL und CREATE BUFFERPOOL)
- Paketcache (Steuerung durch den Konfigurationsparameter *pckcachesz*)
- Sperrenliste (Steuerung durch die Konfigurationsparameter *locklist* und *maxlocks*)
- Sortierspeicher (Steuerung durch die Konfigurationsparameter *sheapthres_shr* und *sortheap*)
- Gemeinsam genutzter Datenbankspeicher (Steuerung durch den Konfigurationsparameter *database_memory*)

Zum Anzeigen der aktuellen Einstellung dieses Parameters verwenden Sie den Befehl GET DATABASE CONFIGURATION mit dem Parameter SHOW DETAIL. Die folgenden Einstellungen, die für diesen Parameter zurückgegeben werden, sind möglich:

```
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = OFF
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = ON (Aktiv)
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = ON (Inaktiv)
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = ON
```

Diese Werte geben Folgendes an:

- `ON (Aktiv)`: Mit der Speicheroptimierung wird der Speicher im System aktiv optimiert.
- `ON (Inaktiv)`: Die Speicheroptimierung ist zwar aktiviert, sie wird jedoch nicht aktiv ausgeführt, da weniger als zwei Speicherkonsumenten für die automatische Optimierung aktiviert sind.
- `ON` (weder (Aktiv) noch (Inaktiv)): Aus einer Abfrage ohne die Option SHOW DETAIL bzw. ohne Datenbankverbindung.

In partitionierten Umgebungen zeigt der Konfigurationsparameter *self_tuning_mem* den Wert ON (Aktiv) nur für die Datenbankpartition, in der die Optimierungsfunktion ausgeführt wird. Auf allen anderen Knoten hat *self_tuning_mem* den Wert 'ON (Inaktiv)'. Wenn Sie feststellen möchten, ob die Speicheroptimierung in einer partitionierten Datenbank aktiv ist, müssen daher Sie den Parameter *self_tuning_mem* in allen Datenbankpartitionen überprüfen.

Wenn Sie eine Migration von einer früheren Version von DB2 auf DB2 Version 9 durchgeführt haben und planen, die Funktion für die automatische Speicheroptimierung zu nutzen, sollten Sie die folgenden Diagnoseanzeiger konfigurieren, um die Überprüfung von Schwellenwerten oder Statusangaben zu inaktivieren:

- Auslastung des gemeinsamen Sortierspeichers - db.sort_shrmem_util
- Prozentsatz der Sortierüberläufe - db.spilled_sorts
- Langfristige Auslastung des gemeinsamen Sortierspeichers - db.max_sort_shrmem_util
- Auslastung der Sperrenliste - db.locklist_util
- Rate für Sperreneskalation - db.lock_escal_rate
- Trefferquote im Paketcache - db.pkgcache_hitratio

Eines der Ziele der automatischen Speicheroptimierungsfunktion besteht darin, die Zuordnung von Speicher für einen Speicherkonsumenten zu vermeiden, wenn dies nicht unmittelbar erforderlich ist. Daher kann die Auslastung des Speichers, der einem Speicherkonsumenten zugeordnet ist, dem Wert 100 % sehr nahe kommen, bevor weiterer Speicher zugeordnet wird. Durch die Inaktivierung dieser Diagnoseanzeiger vermeiden Sie unnötige Alerts, die durch die hohe Speicherauslastung für einen Speicherkonsumenten ausgelöst werden.

Bei Instanzen, die in DB2 Version 9 erstellt werden, sind diese Diagnoseanzeiger standardmäßig inaktiviert.

seqdetect - Markierung für Sequenzerkennung

Mit diesem Parameter wird gesteuert, ob der Datenbankmanager bei der E/A-Aktivität das sequenzielle Lesen von Seiten aufspüren darf.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Yes [Yes; No]

Der Datenbankmanager kann die E/A-Operationen überwachen und, wenn Seiten sequenziell gelesen werden, den E/A-Vorablesezugriff aktivieren. Diese Art des sequenziellen Vorablesens ist die *Sequenzerkennung*.

Wenn für diesen Parameter der Wert "No" angegeben wird, findet das Vorablesen nur dann statt, wenn der Datenbankmanager erkennt, dass dies nützlich ist, z. B. bei Tabellensortierungen, Tabellensuchen oder einem Vorablesezugriff über Listen.

Empfehlung: In den meisten Fällen sollte der Standardwert für diesen Parameter verwendet werden. Inaktivieren Sie die Sequenzerkennung nur dann, wenn andere Optimierungsversuche bisher nicht zur Behebung schwerwiegender Leistungsprobleme bei Abfragen geführt haben.

sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge

Dieser Parameter gibt einen veränderlichen Grenzwert für die Gesamtmenge des gemeinsam genutzten Datenbankspeichers an, die gleichzeitig von Sortierspeicherkonsumenten verwendet werden kann.

Konfigurationstyp
Datenbank

Gilt für
OLAP-Funktionen

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
Automatic [250 - 524 288]

64-Bit-Plattformen
Automatic [250 - 2 147 483 647]

Maßeinheit
Seiten (4 KB)

Neben Sortiervorgängen gibt es noch weitere Konsumenten von Sortierspeicher, wie zum Beispiel Hash-Joins, logische Verknüpfungen von Indizes über AND (Index ANDing), logische Verknüpfungen von Blockindizes über AND, Mischjoins und speicherinterne Tabellen. Wenn sich die Gesamtgröße des gemeinsam genutzten Speichers, der von Konsumenten des gemeinsam genutzten Sortierspeichers beansprucht wird, dem Grenzwert *sheapthres_shr* nähert, wird ein Speicherdrosselungsmechanismus aktiviert, sodass bei nachfolgenden Anforderungen von Konsumenten des gemeinsam genutzten Sortierspeichers unter Umständen weniger Speicher als angefordert zugeordnet wird. Jedoch wird in jedem Fall mehr als das Minimum zugeordnet, das zur Ausführung der Operation erforderlich ist. Wenn der Grenzwert *sheapthres_shr* überschritten wird, erhalten alle Anforderungen von gemeinsam genutztem Sortierspeicher durch Sortierspeicherkonsumenten nur die minimale Größe an Speicher, die erforderlich ist, um die Operation zu beenden.

Wenn die Gesamtmenge des gemeinsam genutzten Speichers für aktive gemeinsame Sortiervorgänge diesen Grenzwert erreicht, können nachfolgende Sortiervorgänge fehlschlagen (SQL0955C).

Wenn der Datenbankkonfigurationsparameter *'sheapthres'* den Wert 0 besitzt, verwenden alle Sortierspeicherkonsumenten für die Datenbank den gemeinsam genutzten Datenbankspeicher mit *sheapthres_shr* anstelle des privaten Sortierspeichers.

Der Wert AUTOMATIC aktiviert *'sheapthres_shr'* für die automatische Leistungsoptimierung. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch anzupassen. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherkonsumenten verteilt, müssen mindestens zwei Speicherkonsumenten für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann. Speicherkonsumenten sind zum Beispiel SHEAPTHRES_SHR, PCKCACHESZ, BUFFER POOL (jeder Pufferpool zählt als einer), LOCKLIST und DATABASE_MEMORY.

Die automatische Optimierung von *sheapthres_shr* ist nur zulässig, wenn der Datenbankkonfigurationsparameter *sheapthres* auf den Wert 0 gesetzt ist.

Der Wert des Parameters *sortheap* wird zusammen mit dem Parameter *sheapthres_shr* optimiert, sodass eine Inaktivierung der automatischen Optimierung des Parameters *sortheap* automatisch auch eine Inaktivierung der automatischen Optimierung des Parameters *sheapthres_shr* bewirkt. Die Aktivierung der automatischen Optimierung für den Parameter *sheapthres_shr* führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters *sortheap*.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter *self_tuning_mem* muss auf "ON" gesetzt sein).

Wenn der Wert dieses Parameters online aktualisiert wird, wird der neue Wert nur von neuen Anforderungen für gemeinsam genutzten Sortierspeicher verwendet, die nach der Aktualisierung erfolgen. Es wird empfohlen, vor einer Verringerung des Werts von *sheapthres_shr* den Wert von *sortheap* herabzusetzen und vor einer Erhöhung des Werts von *sortheap* den Wert von *sheapthres_shr* heraufzusetzen.

Wenn der Wert des Datenbankkonfigurationsparameters *sheapthres* größer als 0 ist, ist der Parameter *sheapthres_shr* nur in zwei Fällen von Bedeutung:

- Wenn der Konfigurationsparameter *intra_parallel* des Datenbankmanagers auf *yes* gesetzt ist, da keine gemeinsamen Sortiervorgänge durchgeführt werden, wenn *intra_parallel* auf *no* gesetzt ist.
- Wenn der Konzentrator aktiviert ist (d. h. wenn *max_connections* größer als *max_coordagents* ist), da Sortiervorgängen, die einen mit der Option WITH HOLD deklarierten Cursor verwenden, Speicher aus dem gemeinsam genutzten Speicher zugeordnet wird.

softmax - Recoverybereich und Intervall für bedingte Prüfpunkte

Mit diesem Parameter wird die Häufigkeit von bedingten Prüfpunkten und der Recoverybereich festgelegt, die bei einer Recovery nach einem Systemabsturz für Unterstützung sorgen.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
100 [1 – 100 * *logprimary*]

Maßeinheit
Prozentsatz der Größe einer primären Protokolldatei

Dieser Parameter hat folgende Funktionen:

- Beeinflussen der Anzahl von Protokolldateien, die für die Recovery nach einem Systemabsturz (z. B. nach einem Stromausfall) erforderlich sind. Wenn beispielsweise der Standardwert verwendet wird, versucht der Datenbankmanager, die Anzahl der wiederherzustellenden Protokolldateien auf 1 zu halten. Wenn Sie 300 als Wert für diesen Parameter angeben, versucht der Datenbankmanager, die Anzahl der wiederherzustellenden Protokolldateien auf 3 zu halten.

  Beim Steuern der Anzahl der Protokolldateien, die für die Recovery nach einem Systemabsturz erforderlich sind, verwendet der Datenbankmanager diesen Parameter zum Starten der Seitenlöschfunktionen. Dadurch wird sichergestellt, dass Seiten, die älter sind als das angegebene Recoveryfenster, bereits auf Platte geschrieben sind.
- Festlegen der Frequenz der bedingten Prüfpunkte.

Zum Zeitpunkt einer Datenbankstörung, die zum Beispiel durch einen Stromausfall verursacht wird, kann für in der Datenbank ausgeführte Änderungen Folgendes gelten:

- Die Änderungen wurden nicht mit COMMIT festgeschrieben, jedoch wurden die Daten im Pufferpool aktualisiert.
- Die Änderungen wurden mit COMMIT festgeschrieben, jedoch noch nicht vom Pufferpool auf die Festplatte geschrieben.
- Die Änderungen wurden mit COMMIT festgeschrieben und vom Pufferpool auf die Festplatte geschrieben.

Wenn eine Datenbank erneut gestartet wird, werden die Protokolldateien verwendet, um eine Recovery der Datenbank nach dem Systemabsturz auszuführen, die sicherstellt, dass die Datenbank in einem konsistenten Zustand verbleibt (d. h., alle mit COMMIT festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen, und keine der nicht festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen).

Der Datenbankmanager verwendet in einer Protokollsteuerdatei gespeicherte Informationen, um festzustellen, welche Datensätze aus der Protokolldatei in der Datenbank nachvollzogen werden müssen. (Der Datenbankmanager verwaltet tatsächlich zwei Kopien der Protokollsteuerdatei, SQLOGCTL.LFH.1 und SQLOGCTL.LFH.2, sodass er bei einer Beschädigung der einen Kopie immer noch die andere Kopie verwenden kann.) Diese Protokollsteuerdateien werden in regelmäßigen Abständen auf die Festplatte geschrieben, und der Datenbankmanager kann abhängig von der

Frequenz dieses Ereignisses Protokollsätze festgeschriebener Transaktionen oder Protokollsätze zu Änderungen, die bereits aus dem Pufferpool auf Platte geschrieben wurden, nachvollziehen. Diese Protokollsätze haben keine Auswirkung auf die Datenbank, das Nachvollziehen dieser Protokollsätze führt jedoch zu einem gewissen erhöhten Systemaufwand während des Neustarts der Datenbank.

Die Protokollsteuerdateien werden immer dann auf die Festplatte geschrieben, wenn eine Protokolldatei voll ist, und außerdem bei bedingten Prüfpunkten. Sie können diesen Konfigurationsparameter dazu verwenden, zusätzliche bedingte Prüfpunkte auszulösen.

Die Ablaufsteuerung für bedingte Prüfpunkte wird mithilfe der Differenz zwischen dem „aktuellen Stand" und dem „aufgezeichneten Stand" festgelegt. Diese Differenz wird als Prozentsatz vom Wert des Parameters *logfilsiz* angegeben. Der „aufgezeichnete Stand" wird anhand des ältesten gültigen Protokollsatzes ermittelt, der in den Protokollsteuerdateien auf der Festplatte angegeben wird, während der „aktuelle Stand" anhand der Protokollsteuerinformationen im Hauptspeicher ermittelt wird. (Der älteste gültige Protokollsatz ist der erste Protokollsatz, der bei einem Recoveryprozess gelesen würde.) Der bedingte Prüfpunkt wird ausgelöst, wenn der nach der folgenden Formel berechnete Wert größer oder gleich dem Wert dieses Parameters ist:

```
( (Bereich zw. aufgezeichnetem u. aktuellem Stand) / logfilsiz ) * 100
```

Empfehlung: Sie können den Wert dieses Parameters erhöhen oder verringern, je nachdem, ob Ihr akzeptables Recoveryfenster größer oder kleiner als eine Protokolldatei ist. Wenn Sie den Wert dieses Parameters herabsetzen, wird der Datenbankmanager veranlasst, die Seitenlöschfunktionen häufiger auszulösen und häufiger bedingte Prüfpunkte anzusetzen. Diese Maßnahmen können die Anzahl der Protokollsätze, die verarbeitet werden müssen, und die Anzahl der überschüssigen Protokollsätze, die während der Recovery verarbeitet werden, verringern.

Beachten Sie jedoch, dass mehr Trigger von Seitenlöschfunktionen und häufigere bedingte Prüfpunkte den Systemaufwand erhöhen, der mit der Protokollierung der Datenbank verbunden ist, was sich negativ auf die Leistung des Datenbankmanagers auswirken kann. Daneben können auch folgende Umstände dazu führen, dass häufigere bedingte Prüfpunkte die für den Neustart einer Datenbank benötigte Zeit nicht verkürzen:

- Es werden sehr lange Transaktionen mit wenigen Commitpunkten ausgeführt.
- Der Pufferpool ist sehr groß, und die Seiten mit den festgeschriebenen Transaktionen werden nicht sehr oft auf die Platte zurückgeschrieben. (Beachten Sie, dass durch das Verwenden asynchroner Seitenlöschfunktionen solche Situationen vermieden werden können.)

In beiden Fällen ändern sich die Protokollsteuerdaten im Hauptspeicher nur selten, und es ist nur dann sinnvoll, die Protokollsteuerdaten auf die Festplatte zu schreiben, wenn sie sich geändert haben.

sortheap - Sortierspeichergröße

Mit diesem Parameter wird die maximale Anzahl von Seiten des privaten Speichers definiert, die für private Sortiervorgänge verwendet werden soll, bzw. die maximale Anzahl von Seiten des gemeinsamen Speichers, die für gemeinsame Sortiervorgänge verwendet werden soll.

Konfigurationstyp
Datenbank

Gilt für
OLAP-Funktionen

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
Automatic [16 - 524 288]

64-Bit-Plattformen
Automatic [16 - 4 194 303]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wie zur Ausführung von Sortiervorgängen erforderlich

Freigabe
Wenn der Sortiervorgang abgeschlossen ist

Wenn es sich um einen privaten Sortiervorgang handelt, bezieht sich dieser Parameter auf den privaten Agentenspeicher. Handelt es sich um einen gemeinsamen Sortiervorgang, bezieht sich dieser Parameter auf den gemeinsamen Datenbankspeicher. Jeder Sortiervorgang verwendet einen getrennten Sortierspeicher, der bei Bedarf vom Datenbankmanager zugeordnet wird. Dieser Sortierspeicher ist der Bereich, in dem Daten sortiert werden. Bei Steuerung durch das Optimierungsprogramm wird anhand der vom Optimierungsprogramm bereitgestellten Informationen ein kleinerer als der durch diesen Parameter angegebene Sortierspeicher zugeordnet.

Der Wert AUTOMATIC für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch anzupassen.

Der Wert des Parameters *sortheap* wird zusammen mit dem Parameter *sheapthres_shr* optimiert, sodass eine Inaktivierung der automatischen Optimierung des Parameters *sortheap* automatisch auch eine Inaktivierung der automatischen Optimierung des Parameters *sheapthres_shr* bewirkt. Die Aktivierung der automatischen Optimierung für den Parameter *sheapthres_shr* führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters *sortheap*. Der Parameter *sortheap* kann jedoch für die automatische Optimierung aktiviert werden, ohne dass der Parameter *sheapthres_shr* auf AUTOMATIC gesetzt ist.

Die automatische Optimierung von *sortheap* ist nur zulässig, wenn der Datenbankkonfigurationsparameter *sheapthres* auf den Wert 0 gesetzt ist.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter *self_tuning_mem* muss auf ″ON″ gesetzt sein).

Empfehlung: Bei der Arbeit mit dem Sortierspeicher ist Folgendes zu beachten:

- Geeignete Indizes können die Verwendung des Sortierspeichers minimieren.

- Hash-Join-Puffer, logisches Verknüpfen von Blockindizes über AND (Index ANDing), Mischjoins, Tabellen im Speicher und dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND (Index ANDing) und Sternjoins verwendet werden) nutzen Sortierspeicher. Erhöhen Sie den Wert dieses Parameters, wenn diese Verfahren verwendet werden.
- Erhöhen Sie den Wert dieses Parameters, wenn häufig umfangreiche Sortiervorgänge ausgeführt werden müssen.
- Wenn Sie den Wert dieses Parameters erhöhen, sollten Sie überprüfen, ob auch die Werte der Parameter *sheapthres* und *sheapthres_shr* in der Konfigurationsdatei des Datenbankmanagers angepasst werden müssen.
- Die Größe des Zwischenspeichers für Sortierlisten wird vom Optimierungsprogramm zur Bestimmung der Zugriffspfade verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl REBIND).

Bei einer Aktualisierung des Werts für 'sortheap' beginnt der Datenbankmanager für sämtliche aktuellen und neuen Sortiervorgänge unmittelbar mit der Verwendung dieses neuen Werts.

stat_heap_sz - Größe des Statistikzwischenspeichers

Mit diesem Parameter wird die *maximale* Größe des Zwischenspeichers angegeben, der bei Erfassung statistischer Daten mit dem Befehl RUNSTATS verwendet wird.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass er nach Bedarf erhöht wird, bis entweder der Grenzwert des Parameters *appl_memory* oder des Parameters *instance_memory* erreicht ist.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Automatic [1 096 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn das Dienstprogramm RUNSTATS gestartet wird

Freigabe
Wenn das Dienstprogramm RUNSTATS beendet ist

Empfehlung: Es wird empfohlen, den Standardwert AUTOMATIC zu verwenden.

stmtheap - Größe des Anweisungszwischenspeichers

Dieser Parameter legt die Größe des Anweisungszwischenspeichers fest, der als Arbeitsbereich für den SQL- oder XQuery-Compiler während der Kompilierung einer SQL- oder XQuery-Anweisung verwendet wird.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass er nach Bedarf erhöht wird, bis entweder der Grenzwert des Parameters *appl_memory* oder des Parameters *instance_memory* erreicht ist.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]

Für 32-Bit- und 64-Bit-Plattformen
Automatic [128 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Für jede Anweisung während des Vorkompilierens oder des Bindens

Freigabe
Wenn das Vorkompilieren oder Binden der betreffenden Anweisung abgeschlossen ist

Dieser Bereich bleibt nicht permanent zugeordnet, sondern wird für die Verarbeitung jeder SQL- oder XQuery-Anweisung einzeln zugeordnet und anschließend wieder freigegeben. Beachten Sie, dass dieser Arbeitsbereich bei dynamischen SQL- oder XQuery-Anweisungen während der Ausführung Ihres Programms, bei statischen SQL- oder XQuery-Anweisungen hingegen während des Bindens, und nicht während der Ausführung des Programms, verwendet wird.

Empfehlung: In den meisten Fällen kann der Standardwert AUTOMATIC für diesen Parameter übernommen werden. Wenn der Wert AUTOMATIC definiert ist, besteht während der Kompilierungsphase der dynamisch programmierten Joinaufzählung (Dynamic Programming Join Enumeration) ein interner Grenzwert für die gesamte zugeordnete Speichergröße. Wenn dieser Wert überschritten wird, wird die Anweisung über die Methode der schnellen Joinaufzählung (Greedy Join Enumeration) kompiliert und lediglich durch die begrenzte verbliebene Kapazität des Anwendungsspeichers (Parameter *appl_memory*) oder des Instanzspeichers (Parameter *instance_memory*) oder beide eingeschränkt. Wenn Ihre Anwendung SQL-Warnungen mit dem Code SQL0437W empfängt und die Laufzeitleistung für Ihre Abfrage nicht akzeptabel ist, kann es sinnvoll sein, manuell einen ausreichenden großen Wert für *stmtheap* festzulegen, um sicherzustellen, dass immer die dynamische Joinaufzählung (Dynamic Join Enumeration) verwendet wird.

Anmerkung: Dynamische Joinaufzählungen treten nur bei der Optimierungsklasse 3 und höher auf (5 ist der Standardwert).

territory - Datenbankgebiet

Dieser Parameter zeigt das Gebiet an, mit dem die Datenbank erstellt wurde. Der Parameter *territory* wird vom Datenbankmanager bei der Verarbeitung von Daten verwendet, für die das Gebiet von Bedeutung ist.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

trackmod - Geänderte Seiten protokollieren

Mit diesem Parameter wird angegeben, ob der Datenbankmanager Datenbankänderungen verfolgt, damit das Backup-Dienstprogramm feststellen kann, welche Untergruppen der Datenbankseiten bei einem inkrementellen Backup zu berücksichtigen und eventuell in das Backup-Image aufzunehmen sind.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes, No]

Wenn Sie diesen Parameter auf ″Yes″ setzen, müssen Sie zunächst ein vollständiges Backup der Datenbank durchführen, damit Sie über eine Ausgangsbasis für inkrementelle Backups verfügen. Wenn dieser Parameter aktiviert ist und ein Tabellenbereich erstellt wird, müssen Sie außerdem ein Backup erstellen, das den betreffenden Tabellenbereich umfasst. Dabei kann es sich entweder um ein Datenbank-Backup oder ein Tabellenbereichs-Backup handeln. Nach erfolgtem Backup dürfen inkrementelle Backups diesen Tabellenbereich umfassen.

tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse

Die Tivoli Storage Manager-Verwaltungsklasse (TSM) legt fest, wie der TSM-Server die Backup-Versionen der Objekte verwalten soll, die gesichert werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Standardmäßig ist für DB2 keine Verwaltungsklasse angegeben.

Beim Ausführen eines TSM-Backups versucht TSM vor Verwendung der im Datenbankkonfigurationsparameter angegebenen Verwaltungsklasse zunächst, das Backup-Objekt an die Verwaltungsklasse zu binden, die in der Liste INCLUDE-EXCLUDE angegeben ist, die sich in der Datei mit den TSM-Clientoptionen befindet. Wird keine Übereinstimmung gefunden, wird die standardmäßige TSM-Verwaltungsklasse verwendet, die auf dem TSM-Server angegeben ist. Anschließend bindet TSM das Backup-Objekt erneut an die mit dem Datenbankkonfigurationsparameter angegebene Verwaltungsklasse.

Daher muss sowohl die Standardverwaltungsklasse als auch die mit dem Datenbankkonfigurationsparameter angegebene Verwaltungsklasse eine Backup-Kopiengruppe enthalten, da die Backup-Operation sonst fehlschlägt.

tsm_nodename - TSM-Knotenname

Mit diesem Parameter kann die Standardeinstellung für den Knotennamen, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Der Knotenname ist erforderlich, um eine Datenbank wiederherstellen zu können, die von einem anderen Knoten aus in TSM gesichert wurde.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus das Backup ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter *tsm_nodename* bei einem Backup mit DB2 (z. B. mit dem Befehl BACKUP DATABASE) überschrieben wird.

tsm_owner - TSM-Eignername

Mit diesem Parameter kann die Standardeinstellung für den Eigner, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Der Eignername ist erforderlich, um eine Datenbank wiederherstellen zu können, die von einem anderen Knoten aus in TSM gesichert wurde. Es ist möglich, dass der Wert für den Parameter *tsm_owner* bei einem Backup mit DB2 (z. B. mit dem Befehl BACKUP DATABASE) überschrieben wird.

Anmerkung: Beim Eignernamen muss die Groß-/Kleinschreibung beachtet werden.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus das Backup ausgeführt wurde.

tsm_password - TSM-Kennwort

Mit diesem Parameter kann die Standardeinstellung für das Kennwort, das dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Dieses Kennwort ist erforderlich, um eine Datenbank wiederherstellen zu können, die in TSM von einem anderen Knoten aus gesichert wurde.

Anmerkung: Wenn der Parameter *tsm_nodename* bei einem Backup mit DB2 (z. B. mit dem Befehl BACKUP DATABASE) überschrieben wird, muss möglicherweise auch der Parameter *tsm_password* festgelegt werden.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus das Backup ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter *tsm_nodename* bei einem Backup mit DB2 überschrieben wird.

user_exit_status - Statusanzeiger für Benutzerexit

Wenn der Wert dieses Parameters "On" lautet, bedeutet dies, dass der Datenbankmanager für die aktualisierende Recovery aktiviert ist und dass das Benutzerexitprogramm verwendet wird, um Protokolldateien zu archivieren und wieder abzurufen, wenn es vom Datenbankmanager aufgerufen wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

userexit - Benutzerexit aktivieren

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Wenn dieser Parameter aktiviert ist, wird unabhängig von der Einstellung des Parameters *logretain* eine Protokollierung mit Protokollspeicherung ausgeführt. Mit diesem Parameter wird außerdem angegeben, dass ein Benutzerexitprogramm verwendet werden soll, um Protokolldateien zu archivieren und wieder abzurufen.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Off [On; Off]

Protokolldateien werden archiviert, wenn die Protokolldatei voll ist. Die Protokolldateien werden wieder abgerufen, wenn das Dienstprogramm ROLLFORWARD sie für den Restore einer Datenbank benötigt.

Nach dem Aktivieren des Parameters *logretain* und/oder *userexit* muss ein Gesamtbackup der Datenbank ausgeführt werden. Dieser Status wird durch den Markierungsparameter *backup_pending* angezeigt.

Wenn beide Parameter inaktiviert werden, ist die aktualisierende Recovery der Datenbank nicht mehr möglich, da keine Protokolldateien mehr gespeichert wer-

den. In diesem Fall löscht der Datenbankmanager alle Protokolldateien im Verzeichnis *logpath* (einschließlich der Online-Archivprotokolldateien), ordnet neue aktive Protokolldateien zu und reaktiviert die Umlaufprotokollierung.

util_heap_sz - Zwischenspeichergröße für Dienstprogramme

Dieser Parameter gibt die maximale Speichergröße an, die gleichzeitig von den Dienstprogrammen BACKUP, RESTORE und LOAD (einschließlich Recovery) verwendet werden kann.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
5000 [16 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn für die Dienstprogramme des Datenbankmanagers erforderlich

Freigabe
Wenn der Speicher nicht mehr vom Dienstprogramm benötigt wird

Empfehlung: Verwenden Sie den Standardwert, bis Ihren Dienstprogrammen nicht mehr genügend Speicher zur Verfügung steht. Wenn der Speicher nicht ausreicht, müssen Sie den Wert erhöhen. Wenn der auf Ihrem System verfügbare Speicher eingeschränkt ist, können Sie den Wert für diesen Parameter herabsetzen, um den von den Dienstprogrammen der Datenbank verwendeten Speicher zu begrenzen. Wenn ein zu niedriger Parameterwert angegeben wird, können Sie Dienstprogramme eventuell nicht gleichzeitig ausführen. Sie sollten diesen Parameter nach Bedarf dynamisch aktualisieren. Für eine kleine Anzahl von Dienstprogrammen setzen Sie diesen Parameter auf einen kleinen Wert. Für eine große Anzahl von Dienstprogrammen oder für speicherintensive Dienstprogramme sollten Sie diesen Parameter auf einen größeren Wert setzen.

vendoropt - Lieferantenoptionen

Mit diesem Parameter werden zusätzliche Parameter angegeben, die DB2 möglicherweise zur Kommunikation mit Speichersystemen bei Backup-, Restore- oder Ladekopieoperationen benötigt.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
NULL [ ]

Einschränkung
Mit dem Konfigurationsparameter **vendoropt** können keine herstellerspezifischen Optionen für Backup- oder Restoremomentaufnahmeoperationen angegeben werden. Zu diesem Zweck müssen Sie den Parameter OPTIONS der Backup- bzw. Restoredienstprogramme verwenden.

wlm_collect_int - Workload-Management-Erfassungsintervall (Konfigurationsparameter)

Dieser Parameter gibt ein Erfassungs- und Zurücksetzungsintervall (in Minuten) für die Workload-Management-Statistik an.

In einem regelmäßigen zeitlichen Abstand in Minuten (x *wlm_collect_int*, wobei x der Wert des Parameters *wlm_collect_int* ist) werden alle Workload-Management-Statistikdaten gesammelt und an sämtliche aktiven Ereignismonitore für Statistiken gesendet; anschließend wird die Statistik zurückgesetzt. Wenn ein aktiver Ereignismonitor vorhanden ist, wird die Statistik in Abhängigkeit davon, wie er erstellt wurde, entweder in die Datei oder in eine Tabelle geschrieben. Wenn kein aktiver Ereignismotor vorhanden ist, wird die Statistik nur zurückgesetzt, aber nicht erfasst.

Der Erfassungs- und Zurücksetzungsprozess wird von der Katalogpartition eingeleitet. Der Parameter *wlm_collect_int* muss in der Katalogpartition angegeben werden. Er wird in keiner anderen Partition verwendet.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
0 [0 (keine Erfassung), 5 - 32 767]

Die vom Ereignismonitor für Statistiken erfasste Workload-Management-Statistik kann zum Überwachen von Kurz- und Langzeitsystemverhalten verwendet werden. Ein kleines Intervall kann zum Abrufen von Kurz- und Langzeitsystemverhalten verwendet werden, da die Ergebnisse zusammengefasst werden können und so ein Langzeitverhalten abgerufen werden kann. Ein manuelles Zusammenfassen der Ergebnisse aus unterschiedlichen Intervallen verkompliziert allerdings die Analyse. Ein kleines Intervall erhöht unnötigerweise den Systemaufwand, wenn es nicht erforderlich ist. Aus diesem Grund sollten Sie das Intervall reduzieren, um das Kurzzeitverhalten zu erfassen, und das Intervall erhöhen, um den Systemaufwand zu reduzieren, wenn lediglich die Ananlyse des Langzeitverhaltens genügt.

Das Intervall muss für jede Datenbank angepasst werden, und nicht für die einzelnen SQL-Anforderungen, Befehlsaufrufe oder Anwendungen. Andere Konfigurationsparameter müssen nicht berücksichtigt werden.

Anmerkung: Alle WLM-Statistiktabellenfunktionen geben Statistikdaten zurück, die seit der letzten Zurücksetzung der Statistik gesammelt wurden. Die Statistik wird regelmäßig auf der Basis des durch diesen Konfigurationsparameter angegebenen Intervalls zurückgesetzt.

Konfigurationsparameter des DB2-Verwaltungsservers (DAS)

authentication - DAS-Authentifizierungstyp

Mit diesem Parameter wird festgelegt, wie und wo die Authentifizierung eines Benutzers stattfindet.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
SERVER_ENCRYPT [SERVER_ENCRYPT; KERBEROS_ENCRYPT]

Hat der Parameter **authentication** den Wert SERVER_ENCRYPT, werden die Benutzer-ID und das Kennwort vom Client an den Server gesendet, damit die Authentifizierung auf dem Server ausgeführt werden kann. Über das Netzwerk gesendete Benutzer-IDs und Kennwörter werden verschlüsselt.

Der Wert KERBEROS_ENCRYPT bedeutet, dass die Authentifizierung auf einem Kerberos-Server mithilfe des Kerberos-Sicherheitsprotokolls für Authentifizierung ausgeführt wird.

Anmerkung: Der Authentifizierungstyp KERBEROS_ENCRYPT wird nur auf Servern unterstützt, auf denen Windows ausgeführt wird.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 9 aktualisiert werden.

contact_host - Speicherposition der Liste mit Ansprechpartnern

Dieser Parameter gibt die Speicherposition der Ansprechpartnerinformationen an, die der Scheduler und der Diagnosemonitor für Benachrichtigungen verwenden.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiger gültiger TCP/IP-Hostname eines DB2-Verwaltungsservers]

Die Position muss der TCP/IP-Hostname eines DB2-Verwaltungsservers sein. Dadurch, dass sich der durch den Parameter *contact_host* angegebene Kontakthost auf einem fernen DAS befinden kann, wird eine gemeinsame Verwendung der

Kontaktliste durch mehrere DB2-Verwaltungsserver unterstützt. Wenn *contact_host* nicht angegeben ist, nimmt der DAS an, dass die Kontaktinformationen lokal vorliegen.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

das_codepage - DAS-Codepage

Dieser Parameter gibt die vom DB2-Verwaltungsserver verwendete Codepage an.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebige gültige DB2-Codepage]

Wenn der Parameter auf NULL gesetzt ist, wird die Standardcodepage des Systems verwendet. Dieser Parameter sollte mit der Ländereinstellung der lokalen DB2-Instanzen kompatibel sein. Andernfalls kann der DB2-Verwaltungsserver nicht mit den DB2-Instanzen kommunizieren.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

das_territory - DAS-Gebiet

Dieser Parameter gibt das vom DB2-Verwaltungsserver verwendete Gebiet an.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiges gültiges DB2-Gebiet]

Wenn der Parameter auf NULL gesetzt ist, wird das Standardgebiet des Systems verwendet.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

dasadm_group - DASADM-Gruppenname

Dieser Parameter definiert den Gruppennamen mit DASADM-Berechtigung für den Datenbankverwaltungsserver.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Gruppenname]

Die Berechtigung DASADM ist die höchste Berechtigungsstufe innerhalb des Datenbankverwaltungsservers (DAS).

Die Berechtigung DASADM wird von den in einer bestimmten Betriebsumgebung verwendeten Sicherheitseinrichtungen ermittelt.

- Bei Windows-Betriebssystemen kann dieser Parameter auf eine beliebige lokale Gruppe gesetzt werden, die in der Sicherheitsdatenbank von Windows definiert ist. Gruppennamen werden akzeptiert, solange sie höchstens eine Länge von 30 Byte haben. Wenn für diesen Parameter der Wert „NULL" definiert wird, haben alle Mitglieder der Administratorengruppe die Berechtigung DASADM.
- Wenn bei Linux- und UNIX-Systemen der Wert „NULL" für diesen Parameter angegeben wird, gilt die Primärgruppe des Instanzeigners standardmäßig als DASADM-Gruppe.

  Ist der Wert nicht „NULL", kann jeder gültige UNIX-Gruppenname als DASADM-Gruppe definiert werden.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

db2system - Name des DB2-Serversystems

Dieser Parameter gibt den Namen an, der von Ihren Benutzern und Datenbankadministratoren verwendet wird, um das DB2-Serversystem anzugeben.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
TCP/IP-Hostname [beliebiger gültiger Systemname]

Dieser Name sollte nach Möglichkeit innerhalb Ihres Netzwerks eindeutig sein.

Dieser Name wird auf der Systemebene der Objektbaumstruktur in der Steuerzentrale angezeigt, um Administratoren bei der Identifizierung von Serversystemen zu helfen, die von der Steuerzentrale aus verwaltet werden können.

Bei der Verwendung der Funktion zum Durchsuchen des Netzwerks des Konfigurationsassistenten gibt DB2-Discovery diesen Namen zurück, und er wird auf der Systemebene der resultierenden Baumstruktur angezeigt. Dieser Name unterstützt Benutzer bei der Identifizierung des Systems, das die Datenbank enthält, auf die sie zugreifen möchten. Bei der Installation wird für *db2system* wie folgt ein Wert festgelegt:

- Unter Windows setzt das Installationsprogramm diesen Parameter auf den Wert des Computernamens, der für das Windows-System angegeben ist.
- Auf UNIX-Systemen wird für diesen Parameter der TCP/IP-Hostname des UNIX-Systems definiert.

discover - DAS-Discovery-Modus

Dieser Parameter bestimmt den Typ des Discovery-Modus, der beim Start des DB2-Verwaltungsservers gestartet wird.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
SEARCH [DISABLE; KNOWN; SEARCH]

- Wenn *discover* = SEARCH angegeben wird, bearbeitet der Verwaltungsserver Discovery-Anforderungen von Clients mit dem Parameterwert SEARCH. SEARCH stellt eine Obermenge der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität zur Verfügung. Wenn *discover* = SEARCH angegeben wird, bearbeitet der Verwaltungsserver Discovery-Anforderungen von Clients sowohl mit dem Parameterwert SEARCH als auch mit dem Parameterwert KNOWN.
- Wenn *discover* = KNOWN angegeben wird, bearbeitet der Verwaltungsserver nur Discovery-Anforderungen von Clients mit dem Parameterwert KNOWN.
- Wenn *discover* = DISABLE angegeben wird, bearbeitet der Verwaltungsserver keine Discovery-Anforderungen. Die Informationen für dieses Serversystem sind im Wesentlichen vor Clients verdeckt.

Der Standard-Discovery-Modus ist SEARCH.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

exec_exp_task - Verfallene Tasks ausführen

Dieser Parameter gibt an, ob der Scheduler Tasks ausführt, die in der Vergangenheit terminiert, bislang jedoch noch nicht ausgeführt wurden.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes; No]

Der Scheduler erkennt verfallene Tasks nur dann, wenn er gestartet wird. Wenn Sie beispielsweise die Ausführung eines Jobs für jeden Samstag terminiert haben und der Scheduler am Freitag ausgeschaltet und am Montag erneut gestartet wird, ist der für Samstag terminierte Job nun ein in der Vergangenheit terminierter Job. Wenn der Parameter *exec_exp_task* auf "Yes" gesetzt ist, wird der für Samstag terminierte Job beim Start des Schedulers ausgeführt.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

jdk_path - Installationspfad für Software Developer's Kit für Java auf DAS

Dieser Parameter gibt das Verzeichnis an, in dem das Software Developer's Kit (SDK) für Java installiert ist, das zu verwenden ist, um DB2-Verwaltungsserverfunktionen auszuführen.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Java-Standardinstallationspfad [beliebiger gültiger Pfad]

Die vom Java-Interpreter verwendeten Umgebungsvariablen werden aus dem Wert dieses Parameters berechnet.

Auf Windows-Betriebssystemen werden Java-Dateien (falls erforderlich) während der DB2-Installation im Verzeichnis sqllib (in java\jdk) gespeichert. Der Konfigurationsparameter *jdk_path* wird anschließend auf sqllib\java\jdk festgelegt. Java wird tatsächlich nicht durch DB2 auf Windows-Plattformen installiert. Die Dateien werden lediglich im Verzeichnis sqllib abgelegt, und zwar unabhängig davon, ob Java bereits installiert ist.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

sched_enable - Schedulermodus

Mit diesem Parameter wird angegeben, ob der Scheduler vom Verwaltungsserver gestartet wird oder nicht.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Off [On; Off]

Der Scheduler ermöglicht Tools wie der Taskzentrale das Terminieren und Ausführen von Tasks auf dem Verwaltungsserver.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

sched_userid - Benutzer-ID für Scheduler

Dieser Parameter gibt die Benutzer-ID an, die der Scheduler verwendet, um eine Verbindung zur Toolskatalogdatenbank herzustellen. Dieser Parameter ist nur dann relevant, wenn die Toolskatalogdatenbank eine ferne Datenbank für den DB2-Verwaltungsserver ist.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Informativ

Standardwert [Bereich]
Null [beliebige gültige Benutzer-ID]

Die vom Scheduler für eine Verbindung zur fernen Toolskatalogdatenbank verwendete Benutzer-ID und das Kennwort werden mit dem Befehl db2admin angegeben.

smtp_server - SMTP-Server

Wenn der Scheduler aktiviert ist, gibt dieser Parameter den SMTP-Server an, den der Scheduler zum Versenden von E-Mail- und Pagerbenachrichtigungen verwendet.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiger gültiger TCP/IP-Hostname für SMTP-Server]

Dieser Parameter wird vom Scheduler und dem Diagnosemonitor verwendet.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

toolscat_db - Toolskatalogdatenbank

Dieser Parameter gibt die vom Scheduler verwendete Toolskatalogdatenbank an.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Aliasname der Datenbank]

Diese Datenbank muss sich im Datenbankverzeichnis der Instanz befinden, das vom Parameter *toolscat_inst* angegeben ist.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

toolscat_inst - Instanz der Toolskatalogdatenbank

Dieser Parameter gibt den Instanznamen an, der vom Scheduler verwendet wird. Außerdem werden toolscat_db und toolscat_schema angegeben, um die Toolskatalogdatenbank anzugeben.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Instanz]

Die Toolskatalogdatenbank (Tools Catalog) enthält die Taskinformationen, die von der Taskzentrale und der Steuerzentrale erstellt werden. Diese Toolskatalogdatenbank muss sich im Datenbankverzeichnis der Instanz befinden, das von diesem Konfigurationsparameter angegeben wird. Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln. Bei Toolskatalogdatenbank muss die Instanz für TCP/IP konfiguriert sein. Bei einer fernen Datenbank muss die Datenbankpartition, die im Datenbankverzeichnis katalogisiert wird, ein TCP/IP-Knoten sein.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

toolscat_schema - Schema der Toolskatalogdatenbank

Dieser Parameter gibt das Schema der vom Scheduler verwendeten Toolskatalog-datenbank an.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiges gültiges Schema]

Das Schema wird verwendet, um eine Reihe von Toolskatalogtabellen und -sichten innerhalb der Datenbank eindeutig anzugeben.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Teil 5. Anhänge und Schlussteil

Anhang A. Übersicht über die technischen Informationen zu DB2

Die technischen Informationen zu DB2 stehen über die folgenden Tools und Methoden zur Verfügung:

- *DB2-Informationszentrale*
  - Themen (zu Tasks, Konzepten und Referenzinformationen)
  - Hilfe für DB2-Tools
  - Beispielprogramme
  - Lernprogramme
- DB2-Bücher
  - PDF-Dateien (für den Download verfügbar)
  - PDF-Dateien (auf der DB2-PDF-DVD)
  - Gedruckte Bücher
- Befehlszeilenhilfe
  - Hilfe für Befehle
  - Hilfe für Nachrichten

Anmerkung: Die Themen der *DB2-Informationszentrale* werden häufiger aktualisiert als die PDF- und Hardcopybücher. Um stets die neuesten Informationen zur Verfügung zu haben, sollten Sie die Dokumentationsaktualisierungen installieren, sobald diese verfügbar sind, oder die *DB2-Informationszentrale* unter ibm.com aufrufen.

Darüber hinaus können Sie auf zusätzliche technische Informationen zu DB2, wie beispielsweise technische Hinweise (Technotes), White Papers und IBM Redbooks, online über ibm.com zugreifen. Rufen Sie die Website 'DB2 Information Management - Software - Library' unter http://www.ibm.com/software/data/sw-library/ auf.

Feedback zur Dokumentation

Senden Sie uns Ihr Feedback zur DB2-Dokumentation! Wenn Sie Anregungen zur Verbesserung der DB2-Dokumentation haben, senden Sie eine E-Mail an db2docs@ca.ibm.com. Das DB2-Dokumentationsteam bearbeitet das gesamte Feedback, kann jedoch nicht im Einzelnen auf Ihre E-Mails antworten. Nennen Sie uns, wenn möglich, konkrete Beispiele, sodass wir die Problemstellung besser beurteilen können. Wenn Sie uns Feedback zu einem bestimmten Thema oder einer bestimmten Hilfedatei senden, geben Sie den entsprechenden Titel sowie die URL an.

Verwenden Sie diese E-Mail-Adresse nicht, wenn Sie sich an die DB2-Kundenunterstützung wenden möchten. Wenn ein technisches Problem bei DB2 vorliegt, das Sie mithilfe der Dokumentation nicht beheben können, fordern Sie beim zuständigen IBM Service-Center Unterstützung an.

Wenn Sie IBM dabei unterstützen möchten, die Benutzerfreundlichkeit der IBM Information Management-Produkte weiter zu verbessern, können Sie uns Ihr Feedback mithilfe der Umfrage zur Verbraucherfreundlichkeit von Software geben: http://www.ibm.com/software/data/info/consumability-survey/ (landessprachliche Version unter https://www-950.ibm.com/survey/oid/wsb.dll/studies/consumabilitywebform.htm?renderlang=de).

Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format

Die folgenden Tabellen enthalten eine Beschreibung der DB2-Bibliothek, die im IBM Publications Center unter www.ibm.com/shop/publications/order zur Verfügung steht. Über die folgende Adresse können Sie englische Handbücher im PDF-Format sowie übersetzte Versionen zu DB2 Version 9.5 herunterladen: www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

In den Tabellen sind die Bücher, die in gedruckter Form zur Verfügung stehen, gekennzeichnet; möglicherweise sind diese in Ihrem Land oder Ihrer Region jedoch nicht verfügbar.

Die Formnummer wird bei jeder Aktualisierung eines Handbuchs erhöht. Anhand der nachfolgenden Liste können Sie sicherstellen, dass Sie die jeweils neueste Version des Handbuchs lesen.

Anmerkung: Die *DB2-Informationszentrale* wird häufiger aktualisiert als die PDF- und Hardcopybücher.

Tabelle 73. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Administrative API Reference	SC23-5842-02	Ja	April 2009
Administrative Routines and Views	SC23-5843-02	Nein	April 2009
Call Level Interface Guide and Reference, Volume 1	SC23-5844-02	Ja	April 2009
Call Level Interface Guide and Reference, Volume 2	SC23-5845-02	Ja	April 2009
Command Reference	SC23-5846-02	Ja	April 2009
Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz	SC12-3917-02	Ja	April 2009
Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz	SC12-3919-02	Ja	April 2009
Datenserver, Datenbanken und Datenbankobjekte	SC12-3912-02	Ja	April 2009
Datenbanksicherheit	SC12-3914-02	Ja	April 2009
Developing ADO.NET and OLE DB Applications	SC23-5851-02	Ja	April 2009
Developing Embedded SQL Applications	SC23-5852-02	Ja	April 2009
Developing Java Applications	SC23-5853-02	Ja	April 2009

Tabelle 73. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Developing Perl and PHP Applications	SC23-5854-02	Nein	April 2009
Developing User-defined Routines (SQL and External)	SC23-5855-02	Ja	April 2009
Getting Started with Database Application Development	GC23-5856-02	Ja	April 2009
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GC12-3922-02	Ja	April 2009
Internationalisierung	SC12-3916-02	Ja	April 2009
Fehlernachrichten, Band 1	GI11-3098-01	Nein	April 2009
Fehlernachrichten, Band 2	GI11-3099-01	Nein	April 2009
Migration	GC12-3921-02	Ja	April 2009
Net Search Extender - Verwaltung und Benutzerhandbuch	SC12-3979-02	Ja	April 2009
Partitionierung und Clustering	SC12-3915-02	Ja	April 2009
Query Patroller - Verwaltung und Benutzerhandbuch	SC12-3977-01	Ja	April 2009
IBM Data Server-Clients - Einstieg	GC12-3924-02	Nein	April 2009
DB2-Server - Einstieg	GC12-3923-02	Ja	April 2009
Spatial Extender und Geodetic Data Management Feature - Benutzer- und Referenzhandbuch	SC12-3978-02	Ja	April 2009
SQL Reference, Volume 1	SC23-5861-02	Ja	April 2009
SQL Reference, Volume 2	SC23-5862-02	Ja	April 2009
Systemmonitor - Handbuch und Referenz	SC12-3918-02	Ja	April 2009
Text Search	SC12-3920-01	Ja	April 2009
Fehlerbehebung	GI11-3097-02	Nein	April 2009
Optimieren der Datenbankleistung	SC12-3913-02	Ja	April 2009
Lernprogramm für Visual Explain	SC12-3932-00	Nein	
Neue Funktionen	SC12-3928-02	Ja	April 2009
Workload-Manager - Handbuch und Referenz	SC12-3929-02	Ja	April 2009

Tabelle 73. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
pureXML - Handbuch	SC12-3930-02	Ja	April 2009
XQuery - Referenz	SC12-3931-02	Nein	April 2009

Tabelle 74. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
DB2 Connect Personal Edition - Einstieg	GC12-3926-02	Ja	April 2009
DB2 Connect-Server - Einstieg	GC12-3927-02	Ja	April 2009
DB2 Connect - Benutzerhandbuch	SC12-3925-02	Ja	April 2009

Tabelle 75. Technische Informationen zu Information Integration

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Information Integration: Föderierte Systeme - Verwaltung	SC12-3759-01	Ja	März 2008
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Ja	März 2008
Information Integration: Konfiguration föderierter Datenquellen	SC12-3777-01	Nein	
Information Integration: SQL Replication - Handbuch und Referenz	SC12-3782-01	Ja	März 2008
Information Integration: Replikation und Event-Publishing - Einführung	GC12-3779-01	Ja	März 2008

Bestellen gedruckter DB2-Bücher

Gedruckte DB2-Bücher können Sie in den meisten Ländern oder Regionen online bestellen. Das Bestellen gedruckter DB2-Bücher ist stets über den zuständigen IBM Ansprechpartner möglich. Beachten Sie hierbei bitte, dass einige Softcopybücher auf der DVD mit der *DB2-PDF-Dokumentation* nicht in gedruckter Form verfügbar sind. So sind beispielsweise die beiden Bände des Handbuchs *DB2 Fehlernachrichten* nicht in gedruckter Form erhältlich.

Gedruckte Versionen vieler DB2-Bücher, die auf der DVD mit der DB2-PDF-Dokumentation verfügbar sind, können gegen eine Gebühr bei IBM bestellt werden. Abhängig vom jeweiligen Land bzw. der jeweiligen Region können Sie Bücher möglicherweise online über das IBM Publications Center bestellen. Ist im jeweiligen Land bzw. der jeweiligen Region keine Onlinebestellung möglich, können Sie

gedruckte DB2-Bücher stets über den zuständigen IBM Ansprechpartner bestellen. Nicht alle Bücher, die auf der DVD mit der DB2-PDF-Dokumentation verfügbar sind, können in gedruckter Form bestellt werden.

Anmerkung: Über http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 haben Sie Zugriff auf die DB2-Informationszentrale, wo Sie die neueste und umfassendste DB2-Dokumentation finden.

Gehen Sie wie folgt vor, um gedruckte DB2-Bücher zu bestellen:

- Informationen dazu, ob in Ihrem Land oder Ihrer Region die Bestellung von gedruckten DB2-Büchern möglich ist, finden Sie auf der Website mit dem IBM Publications Center unter http://www.ibm.com/shop/publications/order. Wählen Sie ein Land, eine Region oder eine Sprache aus, um die Bestellinformationen für Veröffentlichungen aufzurufen, und führen Sie dann die entsprechenden Schritte des Bestellverfahrens für Ihr Land bzw. Ihre Region aus.
- Gehen Sie wie folgt vor, um gedruckte DB2-Bücher beim zuständigen IBM Ansprechpartner zu bestellen:
  1. Kontaktinformationen zum zuständigen Ansprechpartner finden Sie auf einer der folgenden Websites:
     - IBM Verzeichnis weltweiter Kontakte unter www.ibm.com/planetwide.
     - Website mit IBM Veröffentlichungen unter http://www.ibm.com/shop/publications/order. Wählen Sie das gewünschte Land, die gewünschte Region oder die gewünschte Sprache aus, um auf die entsprechende Homepage mit Veröffentlichungen Ihres Landes bzw. Ihrer Region zuzugreifen. Folgen Sie auf dieser Seite dem Link für Informationen zu dieser Site ("About this Site").
  2. Geben Sie bei Ihrem Anruf an, dass Sie eine DB2-Veröffentlichung bestellen möchten.
  3. Teilen Sie dem zuständigen Ansprechpartner die Titel und Formularnummern der Bücher mit, die Sie bestellen möchten. Titel und Formularnummern finden Sie unter „Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format" auf Seite 694.

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2 gibt für Bedingungen, die aufgrund einer SQL-Anweisung generiert werden können, einen SQLSTATE-Wert zurück. Die SQLSTATE-Hilfe erläutert die Bedeutung der SQL-Statuswerte und der SQL-Statusklassencodes.

Zum Aufrufen der Hilfe für SQL-Statuswerte müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? sqlstate oder ? klassencode
```

Hierbei steht *sqlstate* für einen gültigen fünfstelligen SQL-Statuswert und *klassencode* für die ersten beiden Ziffern dieses Statuswertes.
So kann beispielsweise durch die Eingabe von `? 08003` Hilfe für den SQL-Statuswert 08003 angezeigt werden, durch die Eingabe von `? 08` Hilfe für den Klassencode 08.

Zugriff auf verschiedene Versionen der DB2-Informationszentrale

Für Themen aus DB2 Version 9.5 lautet die URL der DB2-Informationszentrale http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/.

Für Themen aus DB2 Version 9 lautet die URL der DB2-Informationszentrale http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Für Themen aus DB2 Version 8 lautet die URL der Informationszentrale (Version 8, 'Information - Unterstützung') http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Anzeigen von Themen in der gewünschten Sprache in der DB2-Informationszentrale

In der DB2-Informationszentrale werden Themen, wenn möglich, in der Sprache angezeigt, die in den Vorgaben Ihres Browsers angegeben ist. Falls ein Thema nicht in die gewünschte Sprache übersetzt wurde, wird es in der DB2-Informationszentrale in Englisch angezeigt.

- Um Themen in der gewünschten Sprache im Browser 'Internet Explorer' anzuzeigen, gehen Sie wie folgt vor:
  1. Klicken Sie im Internet Explorer **Extras** —> **Internetoptionen...** —> **Sprachen...** an. Das Fenster **Spracheinstellung** wird geöffnet.
  2. Stellen Sie sicher, dass die gewünschte Sprache als erster Eintrag in der Liste angegeben ist.
     - Klicken Sie den Knopf **Hinzufügen...** an, um eine neue Sprache zur Liste hinzuzufügen.

       Anmerkung: Das Hinzufügen einer Sprache bedeutet nicht zwangsläufig, dass der Computer über die erforderlichen Schriftarten verfügt, um die Themen in der gewünschten Sprache anzuzeigen.
     - Um eine Sprache an den Anfang der Liste zu verschieben, wählen Sie zunächst die gewünschte Sprache und anschließend den Knopf **Nach oben** aus, bis die Sprache an erster Stelle in der Liste steht.
  3. Löschen Sie den Inhalt des Browser-Cache, und aktualisieren Sie anschließend die Seite, um die DB2-Informationszentrale in der gewünschten Sprache anzuzeigen.
- Um Themen in der gewünschten Sprache in einem Firefox- oder Mozilla-Browser anzuzeigen, gehen Sie wie folgt vor:
  1. Wählen Sie den Knopf im Bereich **Languages** des Dialogfensters **Tools** —> **Options** —> **Advanced** aus. Die Anzeige für die Auswahl der Sprache wird im Fenster mit den Einstellungen aufgerufen.
  2. Stellen Sie sicher, dass die gewünschte Sprache als erster Eintrag in der Liste angegeben ist.
     - Wenn Sie eine neue Sprache zur Liste hinzufügen möchten, klicken Sie den Knopf **Add...** an, um eine Sprache im entsprechenden Fenster auszuwählen.
     - Um eine Sprache an den Anfang der Liste zu verschieben, wählen Sie zunächst die gewünschte Sprache und anschließend den Knopf **Move Up** aus, bis die Sprache an erster Stelle in der Liste steht.
  3. Löschen Sie den Inhalt des Browser-Cache, und aktualisieren Sie anschließend die Seite, um die DB2-Informationszentrale in der gewünschten Sprache anzuzeigen.

Bei einigen Kombinationen aus Browser und Betriebssystem müssen Sie möglicherweise auch die Ländereinstellungen des Betriebssystems in die gewünschte Locale und Sprache ändern.

Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten DB2-Informationszentrale

Wenn Sie die DB2-Informationszentrale lokal installiert haben, können Sie Dokumentationsaktualisierungen von IBM abrufen und installieren.

Zur Aktualisierung der lokal installierten *DB2-Informationszentrale* sind die folgenden Schritte erforderlich:

1. Stoppen Sie die *DB2-Informationszentrale* auf Ihrem Computer und starten Sie die Informationszentrale im Standalone-Modus erneut. Die Ausführung der Informationszentrale im Standalone-Modus verhindert, dass andere Benutzer in Ihrem Netz auf die Informationszentrale zugreifen, und ermöglicht das Anwenden von Aktualisierungen. *DB2-Informationszentralen*, deren Installation nicht als Administrator oder Root ausgeführt wurde, werden stets im Standalone-Modus ausgeführt.
2. Verwenden Sie die Aktualisierungsfunktion, um zu prüfen, welche Aktualisierungen verfügbar sind. Falls Aktualisierungen verfügbar sind, die Sie installieren möchten, können Sie die Aktualisierungsfunktion verwenden, um diese abzurufen und zu installieren.

   Anmerkung: Wenn es in der verwendeten Umgebung erforderlich ist, die Aktualisierungen für die *DB2-Informationszentrale* auf einer Maschine zu installieren, die nicht über ein Verbindung zum Internet verfügt, müssen Sie die Aktualisierungssite auf ein lokales Dateisystem spiegeln und dabei eine Maschine verwenden, die mit dem Internet verbunden ist und auf der die *DB2-Informationszentrale* installiert ist. Wenn viele Benutzer Ihres Netzes die Dokumentationsaktualisierungen installieren sollen, können Sie die Zeit, die jeder einzelne Benutzer für die Aktualisierungen benötigt, reduzieren, indem Sie die Aktualisierungssite lokal spiegeln und ein Proxy dafür erstellen. Ist dies der Fall, verwenden Sie die Aktualisierungsfunktion, um die Pakete abzurufen. Die Aktualisierungsfunktion ist jedoch nur im Standalone-Modus verfügbar.
3. Stoppen Sie die im Standalone-Modus gestartete Informationszentrale, und starten Sie die *DB2-Informationszentrale* auf Ihrem Computer erneut.

Anmerkung: Unter Windows Vista müssen Sie zur Ausführung der nachfolgend aufgeführten Befehle über Administratorberechtigung verfügen. Zum Starten einer Eingabeaufforderung oder eines Grafiktools mit vollen Administratorberechtigungen klicken Sie mit der rechten Maustaste auf die Verknüpfung, und wählen Sie **Als Administrator ausführen** aus.

Gehen Sie wie folgt vor, um die auf Ihrem Computer bzw. Intranet-Server installierte *DB2-Informationszentrale* zu aktualisieren:

1. Stoppen Sie die *DB2-Informationszentrale*.
   - Unter Windows klicken Sie **Start** → **Einstellungen** → **Systemsteuerung** → **Verwaltung** → **Dienste** an. Klicken Sie mit der rechten Maustaste die **DB2-Informationszentrale** an, und wählen Sie **Stoppen** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv95 stop
     ```
2. Starten Sie die Informationszentrale im Standalone-Modus.
   - Unter Windows:
     a. Öffnen Sie ein Befehlsfenster.

b. Navigieren Sie zu dem Pfad, in dem die Informationszentrale installiert ist. Standardmäßig ist die *DB2-Informationszentrale* im Verzeichnis *Programme*\IBM\DB2 Information Center\Version 9.5 installiert, wobei *Programme* die Position des Programmdateiverzeichnisses ist.
c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc\bin.
d. Führen Sie die Datei help_start.bat aus:

```
help_start.bat
```

- Unter Linux:
  a. Navigieren Sie zu dem Pfad, in dem die Informationszentrale installiert ist. Standardmäßig ist die *DB2-Informationszentrale* im Verzeichnis /opt/ibm/db2ic/V9.5 installiert.
  b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc/bin.
  c. Führen Sie das Script help_start aus:

```
help_start
```

Der standardmäßig auf dem System verwendete Web-Browser wird aufgerufen und zeigt die Standalone-Informationszentrale an.

3. Klicken Sie den Aktualisierungsknopf ( ) an. Klicken Sie im rechten Fenster der Informationszentrale den Knopf für die Suche nach Aktualisierungen an. Eine Liste der Aktualisierungen für die vorhandene Dokumentation wird angezeigt.
4. Wählen Sie zum Initiieren des Installationsprozesses die gewünschten Aktualisierungen aus, und klicken Sie anschließend den Knopf für die Installation der Aktualisierungen an.
5. Klicken Sie nach Abschluss des Installationsprozesses **Fertig stellen** an.
6. Stoppen Sie die im Standalone-Modus gestartete Informationszentrale:
   - Unter Windows: Navigieren Sie in das Verzeichnis doc\bin des Installationsverzeichnisses, und führen Sie die Datei help_end.bat aus:

     ```
     help_end.bat
     ```

     Anmerkung: Die Stapeldatei help_end enthält die Befehle, die erforderlich sind, um die Prozesse, die mit der Stapeldatei help_start gestartet wurden, ordnungsgemäß zu beenden. Verwenden Sie nicht die Tastenkombination `Strg+C` oder eine andere Methode, um help_start.bat zu beenden.
   - Unter Linux: Navigieren Sie in das Verzeichnis doc/bin des Installationsverzeichnisses, und führen Sie das Script help_end aus:

     ```
     help_end
     ```

     Anmerkung: Das Script help_end enthält die Befehle, die erforderlich sind, um die Prozesse, die mit dem Script help_start gestartet wurden, ordnungsgemäß zu beenden. Verwenden Sie keine andere Methode, um das Script help_start zu beenden.
7. Starten Sie die *DB2-Informationszentrale* erneut.
   - Unter Windows klicken Sie **Start** → **Einstellungen** → **Systemsteuerung** → **Verwaltung** → **Dienste** an. Klicken Sie mit der rechten Maustaste die **DB2-Informationszentrale** an, und wählen Sie **Start** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:

     ```
     /etc/init.d/db2icdv95 start
     ```

In der aktualisierten *DB2-Informationszentrale* werden die neuen und aktualisierten Themen angezeigt.

DB2-Lernprogramme

Die DB2-Lernprogramme unterstützen Sie dabei, sich mit den unterschiedlichen Aspekten der DB2-Produkte vertraut zu machen. Die Lerneinheiten bieten eine in einzelne Schritte unterteilte Anleitung.

Vorbereitungen

Die XHTML-Version des Lernprogramms kann über die Informationszentrale unter http://publib.boulder.ibm.com/infocenter/db2help/ angezeigt werden.

In einigen der Lerneinheiten werden Beispieldaten und Codebeispiele verwendet. Informationen zu bestimmten Voraussetzungen für die Ausführung der Tasks finden Sie in der Beschreibung des Lernprogramms.

DB2-Lernprogramme

Klicken Sie zum Anzeigen des Lernprogramms den Titel an.

„pureXML" in *pureXML - Handbuch*
Einrichten einer DB2-Datenbank, um XML-Daten zu speichern und Basisoperationen mit dem nativen XML-Datenspeicher auszuführen.

„Visual Explain" in *Lernprogramm für Visual Explain*
Analysieren, Optimieren und Anpassen von SQL-Anweisungen zur Leistungsverbesserung mithilfe von Visual Explain.

Informationen zur Fehlerbehebung in DB2

Eine breite Palette verschiedener Informationen zur Fehlerbestimmung und Fehlerbehebung steht zur Verfügung, um Sie bei der Verwendung von DB2-Datenbankprodukten zu unterstützen.

DB2-Dokumentation
Informationen zur Fehlerbehebung stehen im Handbuch DB2-Fehlerbehebung oder im Abschnitt mit grundlegenden Informationen zu Datenbanken in der DB2-Informationszentrale zur Verfügung. Dort finden Sie Informationen dazu, wie Sie Probleme mithilfe der DB2-Diagnosetools und -Dienstprogramme eingrenzen und identifizieren können, Lösungen für einige der häufigsten Probleme sowie weitere Hinweise zur Behebung von Fehlern und Problemen, die bei der Verwendung der DB2-Datenbankprodukte auftreten können.

DB2-Website mit technischer Unterstützung
Auf der DB2-Website mit technischer Unterstützung finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die Website mit technischer Unterstützung enthält Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports) und Fehlerkorrekturen, Fixpacks sowie weiteren Ressourcen. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Rufen Sie die DB2-Website mit technischer Unterstützung unter http://www.ibm.com/software/data/db2/support/db2_9/ auf.

Bedingungen

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Persönliche Nutzung: Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung: Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Abgesehen von den hier gewährten Berechtigungen erhalten Sie keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum.

IBM behält sich das Recht vor, die in diesem Dokument gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Informationen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendungsfähigkeit oder die Freiheit der Rechte Dritter zur Verfügung gestellt.

Anhang B. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der Produkte, Programme oder Services können auch andere ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder andere Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Dienstleistungen in Verbindung mit Fremdprodukten und Fremddienstleistungen liegt beim Kunden, soweit nicht ausdrücklich solche Verbindungen erwähnt sind.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekannt gegeben. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Dieses Dokument enthält möglicherweise Links oder Verweise auf Websites und Ressourcen anderer Anbieter. Es bestehen keine Zusicherungen, Gewährleistungen oder Verpflichtungen von IBM hinsichtlich der Websites oder Ressourcen anderer Anbieter, auf die im vorliegenden Dokument verwiesen wird, Zugriff besteht oder Links vorhanden sind. Ein Link auf eine Website eines anderen Anbieters bedeutet nicht, dass IBM den Inhalt und die Verwendung dieser Website billigt oder deren Eigentümer anerkennt. Darüber hinaus ist IBM nicht an Transaktionen beteiligt und übernimmt keine Verantwortung für Transaktionen zwischen Ihnen und anderen Anbietern, auch wenn die Informationen (oder Links) zu diesen Anbietern auf einer IBM Website zur Verfügung stehen. IBM ist nicht für die Verfügbarkeit solcher externen Sites oder Ressourcen verantwortlich und übernimmt keine Verantwortung oder Haftung für Inhalte, Services, Produkte oder sonstiges Material, die bzw. das auf diesen oder über diese Sites oder Ressourcen verfügbar sind. Die Software anderer Anbieter unterliegt den Lizenzbedingungen der jeweiligen Software.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Dokument aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung sowie der Allgemeinen Geschäftsbedingungen von IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Garantie, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichen Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden, Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- oder Servicenamen können Marken von IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite Copyright and trademark information unter www.ibm.com/legal/copytrade.shtml.

Folgende Namen sind Marken oder eingetragene Marken anderer Unternehmen.

- Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken von Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.
- Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder ihrer Tochtergesellschaften in den USA und/oder anderen Ländern.
- Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

A

B

C

D

E

F

G

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

SC12-3912-02

Spine information:

DB2 Version 9.5 für Linux, UNIX und Windows

Datenserver, Datenbanken und Datenbankobjekte