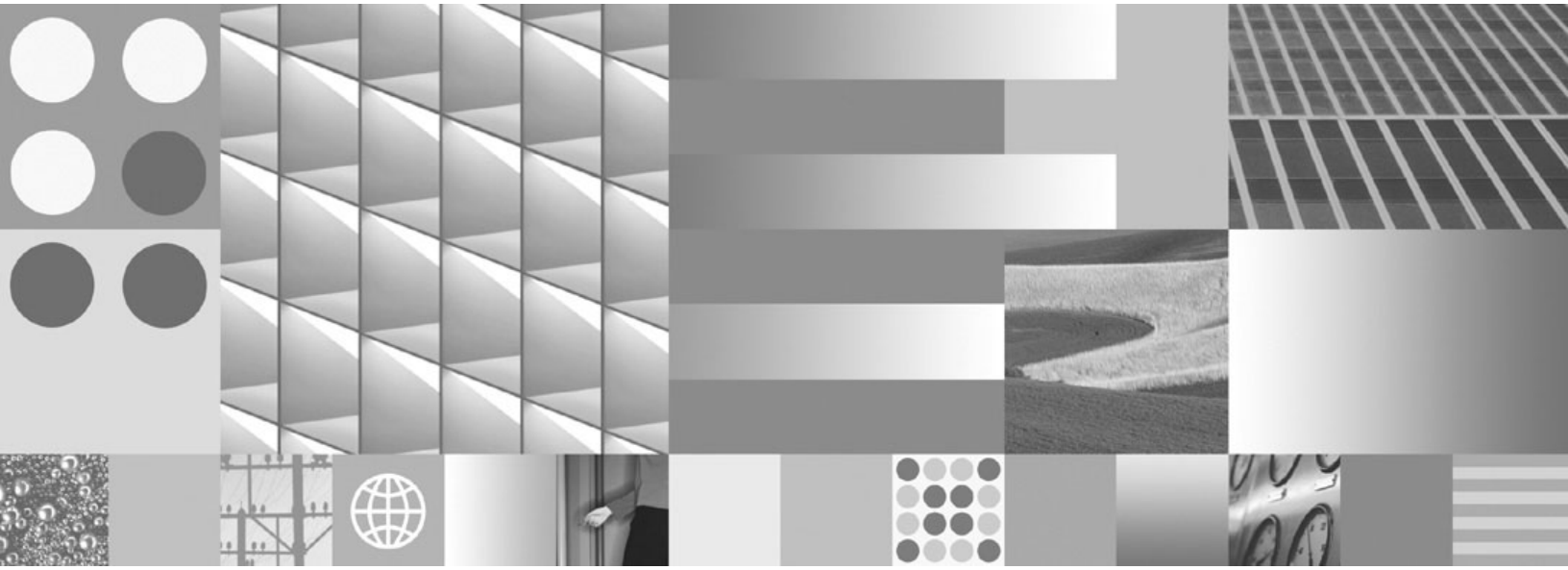
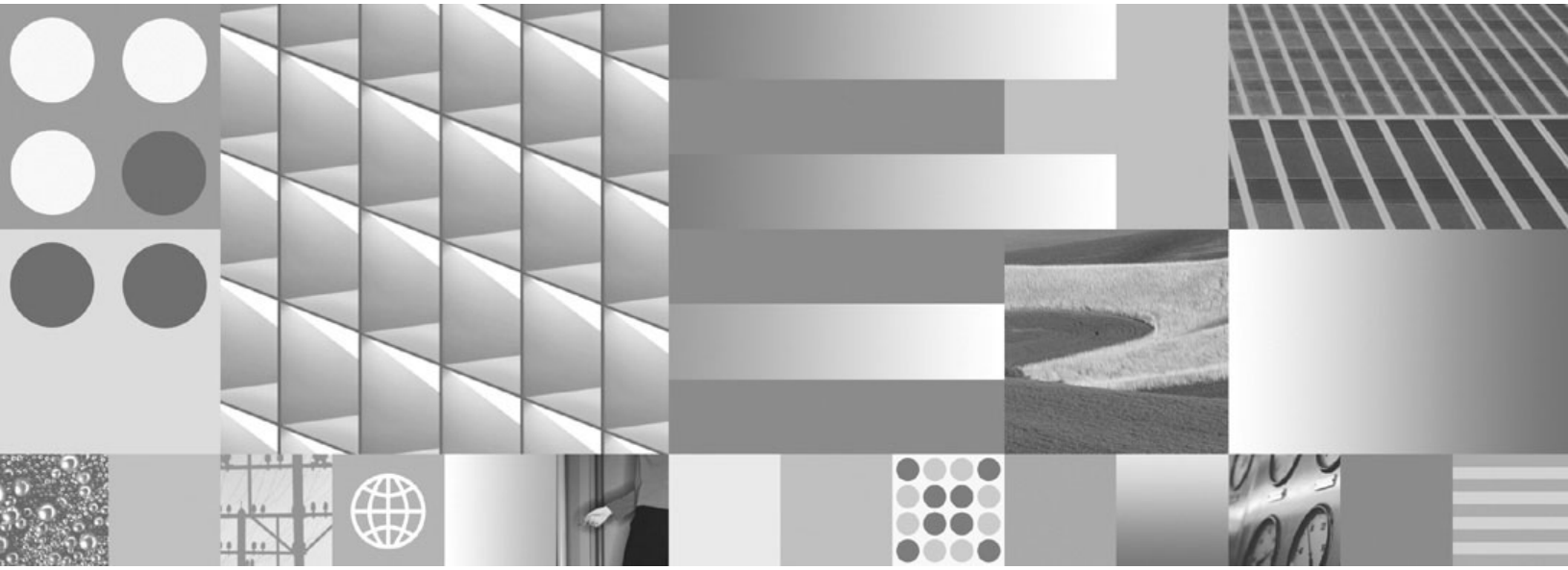
DB2 Version 9.5
für Linux, UNIX und Windows



Optimieren der Datenbankleistung

SC12-3913-01

DB2 Version 9.5
für Linux, UNIX und Windows



Optimieren der Datenbankleistung

SC12-3913-01

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter Anhang B, „Bemerkungen", auf Seite 495 gelesen werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Version 9.5 for Linux, UNIX, and Windows, Tuning Database Performance,
IBM Form SC23-5867-01,
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
März 2008

Inhaltsverzeichnis

Teil 1. Elemente der Leistung

Als *Leistung* wird die Art und Weise bezeichnet, wie ein Computersystem unter einer bestimmten Auslastung arbeitet. Die Leistung wird an der Antwortzeit, am Durchsatz und an der Verfügbarkeit gemessen. Außerdem wird die Leistung von folgenden Faktoren beeinflusst:

- Von den im System verfügbaren Ressourcen
- Von der Auslastung und vom Ausmaß der gemeinsamen Nutzung dieser Ressourcen

Im Allgemeinen optimieren Sie Ihr System, um das Kosten-Nutzen-Verhältnis zu verbessern. Die folgenden speziellen Optimierungsziele können dabei verfolgt werden:

- Verarbeiten einer größeren oder anspruchsvolleren Arbeitsbelastung ohne steigende Verarbeitungskosten

  Zum Beispiel zur Steigerung der Auslastung ohne neue Hardware erwerben oder mehr Prozessorzeit in Kauf nehmen zu müssen.
- Erreichen schnellerer Systemantwortzeiten bzw. eines höheren Durchsatzes ohne Steigerung der Verarbeitungskosten
- Reduzieren von Verarbeitungskosten ohne negative Auswirkungen für Ihre Benutzer

Die Übersetzung von Leistung vom technischen Aspekt in den wirtschaftlichen Aspekt ist schwierig. Eine Leistungsoptimierung kostet natürlich Geld in Form von Zeitaufwand für Personal und Prozessorzeit, sodass vor einem Optimierungsprojekt die Kosten gegen die möglichen Vorteilsgewinne abgewogen werden müssen. Einige dieser Vorteile sind konkret messbar:

- Effizientere Ressourcennutzung
- Die Möglichkeit, weitere Benutzer dem System hinzuzufügen

Andere Vorteile, wie größere Zufriedenheit seitens der Benutzer aufgrund schnellerer Antwortzeiten, sind weniger fassbar. Jedoch sollten alle diese Vorteile in die Überlegungen mit einbezogen werden.

Kapitel 1. Richtlinien zur Leistungsoptimierung

Die folgenden Richtlinien sind als Hilfe für die Entwicklung eines allgemeinen Ansatzes zur Leistungsoptimierung zu verstehen.

Behalten Sie das Gesetz der abnehmenden Ertragsgewinne im Hinterkopf: Die größten Leistungsvorteile werden in der Regel durch die ersten Maßnahmen erzielt. Weitere Änderungen erbringen im Allgemeinen immer kleinere Vorteile und erfordern immer höheren Aufwand.

Optimieren Sie nicht nur des Optimierens wegen: Optimieren Sie, um erkannten Engpässen abzuhelfen. Das Optimieren von Ressourcen, die nicht den Hauptgrund für Leistungsprobleme darstellen, hat wenig oder gar keine Wirkung auf Antwortzeiten, solange Sie nicht die wichtigeren Probleme behoben haben, und kann tatsächlich eine nachfolgende Optimierungsarbeit erschweren. Wenn es ein bedeutendes Verbesserungspotenzial gibt, liegt es in der Verbesserung der Leistung von Ressourcen, die wichtige Faktoren in der Antwortzeit bilden.

Betrachten Sie das gesamte System: Ein Parameter bzw. System lässt sich nicht isoliert optimieren. Bevor Sie Anpassungen vornehmen, überlegen Sie, wie sich diese Anpassungen auf das System als Ganzes auswirken werden.

Ändern Sie jeweils nur einen Parameter gleichzeitig: Ändern Sie nicht mehrere Parameter zur Leistungsoptimierung in einem Schritt. Selbst wenn Sie sich sicher sind, dass alle Änderungen vorteilhaft sind, haben Sie hinterher keine Möglichkeit, den Beitrag jeder Änderung zu bewerten. Darüber hinaus können Sie bei gleichzeitiger Änderung mehrerer Parameter den erzielten Vorteil nicht effektiv den möglicherweise in Kauf genommenen Einbußen gegenüberstellen. Von jeder Anpassung eines Parameters zur Optimierung eines Bereichs ist fast immer auch mindestens ein anderer Bereich betroffen, der vorher vielleicht nicht bedacht wurde. Wenn Sie jeweils nur einen Parameter ändern, haben Sie einen Vergleichspunkt und können feststellen, ob die Änderung die gewünschte Wirkung hat.

Führen Sie Messungen und Neukonfigurierungen nach Ebenen durch: Aus denselben Gründen, aus denen nur jeweils ein Parameter geändert werden soll, empfiehlt sich auch, die einzelnen Ebenen des Systems getrennt zu optimieren. Die folgende Liste von Ebenen innerhalb eines Systems kann Ihnen dabei als Richtlinie dienen:

- Hardware
- Betriebssystem
- Anwendungsserver und -requester
- Datenbankmanager
- SQL- und XQuery-Anweisungen
- Anwendungsprogramme

Prüfen Sie auf Hardware- und Softwareprobleme: Einige Leistungsprobleme können vielleicht durch Wartung der Hardware oder Korrektur der Software oder durch beides behoben werden. Verwenden Sie nicht zu viel Zeit auf die Überwachung und Optimierung des Systems, wenn eine Hardwarewartung oder Softwarekorrektur dies unnötig machen könnte.

Ermitteln Sie die Ursache eines Problems, bevor Sie Ihre Hardware aufrüsten: Auch wenn es so aussieht, als könnten zusätzliche Speicher- und Prozessorkapazitäten die Leistung sofort verbessern, sollten Sie sich die Zeit nehmen, die Engpässe zu lokalisieren und zu verstehen. Sie könnten ansonsten Geld für zusätzlichen Plattenspeicher ausgeben und anschließend feststellen, dass Sie nicht über die Prozessorkapazitäten oder die Kanäle verfügen, um den Speicher vorteilhaft zu nutzen.

Bereiten Sie Zurücksetzungsprozeduren vor, bevor Sie mit der Optimierung beginnen: Wie bereits früher erwähnt, können einige Optimierungsmaßnahmen zu unerwarteten Leistungsergebnissen führen. Wenn sich daraus eine schlechtere Leistung ergibt, sollten die Maßnahmen rückgängig gemacht und alternative Optimierungsmaßnahmen versucht werden. Wenn der vorige Stand in einer Weise gesichert wurde, dass er einfach wiederhergestellt werden kann, ist die Rücknahme der nicht korrekten Informationen wesentlich einfacher.

Kapitel 2. Entwickeln eines Prozesses zur Leistungsverbesserung

Ein Leistungsverbesserungsprozess ist ein iteratives und langfristig angelegtes Verfahren zur Überwachung und Optimierung von Leistungsbereichen. Abhängig von den Überwachungsergebnissen passen Sie und Ihr Optimierungsteam die Konfiguration des Datenbankservers an und nehmen Änderungen an den Anwendungen vor, die den Datenbankserver verwenden.

Gehen Sie bei der Leistungsüberwachung und den Optimierungsentscheidungen von Ihren Kenntnissen über die Arten von Anwendungen, die mit den Daten arbeiten, sowie von den Datenzugriffsmustern aus. Verschiedene Arten von Anwendungen haben unterschiedliche Leistungsanforderungen.

Betrachten Sie die folgende Verfahrensstruktur für den Leistungsverbesserungsprozess als Richtlinie.

Gehen Sie zur Entwicklung eines Leistungsverbesserungsprozesses wie folgt vor:

1. Definieren Sie Leistungsziele.
2. Legen Sie Leistungsindikatoren für die wichtigsten Leistungsprobleme im System fest.
3. Entwickeln Sie einen Leistungsüberwachungsplan und führen Sie ihn aus.
4. Analysieren Sie die Ergebnisse der Überwachung fortlaufend, um zu ermitteln, welche Ressourcen optimiert werden müssen.
5. Nehmen Sie jeweils nur eine Anpassung vor.

   Selbst wenn Sie davon überzeugt sind, dass mehr als eine Ressource eine Optimierung erfordert, oder wenn mehrere Optimierungsoptionen für die zu optimierende Ressource möglich sind, sollten Sie nur eine Änderung pro Optimierungsschritt vornehmen, sodass Sie sicherstellen können, dass Ihre Optimierungsmaßnahmen die gewünschte Wirkung erzielen. An einem bestimmten Punkt lässt sich keine weitere Verbesserung der Leistung durch Optimieren des Datenbankservers und der Anwendungen mehr realisieren. Wenn dieser Punkt erreicht ist, bleibt Ihnen nur die Möglichkeit, Ihre Hardware aufzurüsten.

Zur tatsächlichen Leistungsoptimierung müssen Kompromisslösungen für verschiedene Systemressourcen gefunden werden. Zum Beispiel könnten Sie zur Verbesserung der E/A-Leistung die Pufferpools vergrößern, jedoch benötigen größere Pufferpools mehr Speicher, was wiederum andere Bereiche der Leistung beeinträchtigen könnte.

Kapitel 3. Leistungsinformationen, die Benutzer liefern können

Die ersten Anzeichen dafür, dass Ihr System optimiert werden müsste, könnten Klagen von Benutzern sein. Wenn Sie nicht genügend Zeit zur Definition von Leistungszielen sowie zur Überwachung und Optimierung in umfassender Weise haben, können Sie sich mit der Leistung auseinander setzen, indem Sie Ihren Benutzern zuhören. In der Regel können Sie bestimmen, wo mit der Untersuchung eines Problems zu beginnen ist, indem Sie einige einfache Fragen stellen. Sie könnten Ihre Benutzer zum Beispiel Folgendes fragen:

- Was meinen sie mit „langsamer Reaktion"? Heißt dies, um zehn Prozent langsamer, als Sie erwarten, oder um das Zigfache langsamer?
- Wann haben Sie das Problem bemerkt? Tritt es erst seit kurzem auf oder war es immer da?
- Haben andere Benutzer das gleiche Problem? Handelt es sich bei diesen Benutzern um einen oder zwei Einzelpersonen oder um eine ganze Gruppe?
- Wenn eine Gruppe von Benutzern die gleichen Probleme hat, sind sie mit demselben lokalen Netzwerk (LAN) verbunden?
- Scheinen die Probleme mit einem bestimmten Transaktions- oder Anwendungsprogramm zusammenzuhängen?
- Erkennen Sie ein Muster im Auftreten des Problems? Zum Beispiel: Tritt dieses Problem zu einer bestimmten Tageszeit, z. B. in der Mittagspause, auf oder ist es mehr oder weniger permanent spürbar?

Kapitel 4. Grenzen der Leistungsoptimierung

Eine Optimierung kann die Effizienz eines Systems nur um einen bestimmten Betrag ändern. Überlegen Sie, wie viel Zeit und Geld Sie in die Optimierung der Systemleistung investieren sollten, und wie viel Aufwand an Zeit und Geld den Benutzern des Systems tatsächlich hilft.

Zum Beispiel kann eine Optimierung häufig die Leistung verbessern, wenn das System auf einen Leistungsengpass stößt. Wenn Ihr System nahe an den Leistungsgrenzen arbeitet und sich die Anzahl von Benutzern am System um zehn Prozent erhöht, verlängern sich die Antwortzeiten wahrscheinlich um wesentlich mehr als zehn Prozent. In dieser Situation müssen Sie feststellen, wie Sie dieser Beeinträchtigung der Leistung durch Optimieren des Systems entgegenwirken können.

Es gibt allerdings einen Punkt, ab dem das Optimieren keine weiteren Leistungsgewinne erzielen kann. An diesem Punkt müssen Sie Ihre Ziele und Erwartungen innerhalb der Grenzen Ihrer Umgebung überprüfen. Wenn Sie bedeutsame Leistungsverbesserungen erreichen wollen, müssen Sie vielleicht mehr Plattenspeicher, eine schnellere CPU, zusätzliche CPUs, mehr Arbeitsspeicher, schnellere Kommunikationsverbindungen oder eine Kombination aus diesen Möglichkeiten hinzufügen.

Kapitel 5. Übersicht über die Architektur und Prozesse von DB2

Allgemeine Informationen zur Architektur und zu den Prozessen von DB2 können Ihnen beim Verständnis der ausführlichen Informationen zu bestimmten Themen behilflich sein.

Die folgende Abbildung zeigt eine allgemeine Übersicht über die Architektur und die Prozesse für IBM DB2 Version 9.1.

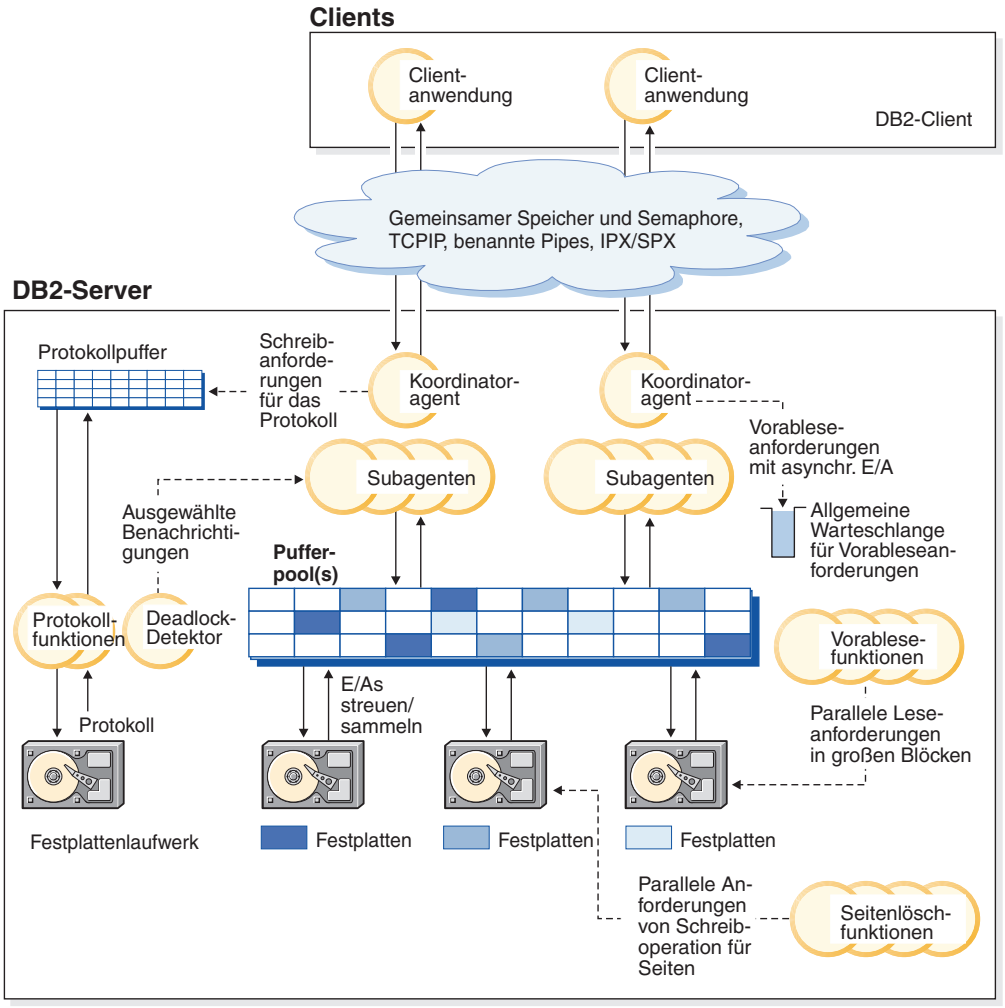


Abbildung 1. Architektur- und Prozessübersicht

Auf der Clientseite werden lokale und/oder ferne Anwendungen mit der Clientbibliothek von DB2 verbunden. Lokale Clients kommunizieren über gemeinsamen Speicher und Semaphore, ferne Clients verwenden ein Protokoll wie benannte Pipes (Named Pipes - NPIPE) oder TCP/IP.

Auf der Serverseite werden die Aktivitäten durch zuteilbare Einheiten der Steuerkomponente, so genannte EDUs (Engine Dispatchable Units), gesteuert. In allen Abbildungen dieses Abschnitts werden EDUs als Kreise oder Gruppen von Kreisen

dargestellt. EDUs werden in Version 9.5 auf allen Plattformen als Threads implementiert. DB2-Agenten sind die gängigste Art von EDUs. Diese Agenten führen den größten Teil der SQL- und XQuery-Verarbeitung für Anwendungen aus. Vorablesefunktionen und Seitenlöschfunktionen sind weitere häufig genutzte Arten von EDUs.

Die Verarbeitung der Anforderungen einer Clientanwendung kann einer Gruppe von Subagenten übertragen werden. Mehrere Subagenten können zugeordnet werden, wenn die Maschine, auf der sich der Server befindet, mehrere Prozessoren hat oder Teil einer partitionierten Datenbank ist. In einer symmetrischen Multiprozessorumgebung (SMP) können mehrere SMP-Subagenten zum Beispiel die höhere Anzahl der Prozessoren ausnutzen.

Alle Agenten und Subagenten werden mithilfe eines Poolfunktionsalgorithmus verwaltet, der die Häufigkeit der Erstellung und Vernichtung von EDUs so gering wie möglich hält.

Pufferpools sind Bereiche des Datenbankserverspeichers, in die Datenbankseiten mit Benutzertabellendaten, Indexdaten und Katalogdaten temporär eingelesen und dort modifiziert werden können. Pufferpools sind eine Schlüsseldeterminante der Datenbankleistung, weil der Zugriff auf Daten im Hauptspeicher wesentlich schneller als auf Daten auf dem Plattenspeicher erfolgen kann. Wenn größere Teile der Daten, die von Anwendungen benötigt werden, in einem Pufferpool vorhanden sind, ist für den Zugriff auf die Daten weniger Zeit erforderlich, als wenn sie auf einer Festplatte gesucht werden müssten.

Die Konfiguration der Pufferpools sowie der EDUs für Vorablesefunktionen und Seitenlöschfunktionen steuert, wie schnell auf Daten zugegriffen werden kann und wie rasch sie Anwendungen verfügbar gemacht werden können.

- **Vorablesefunktionen** rufen Daten von der Festplatte ab und lesen sie in den Pufferpool ein, bevor Anwendungen die Daten benötigen. Beispielsweise müssten Anwendungen, die große Volumina von Daten durchsuchen müssen, darauf warten, dass Daten vom Plattenspeicher in den Pufferpool gelesen werden, wenn es keine Vorablesefunktionen für Daten gäbe. Agenten der Anwendung senden asynchrone Vorableseanforderungen an eine allgemeine Vorablesewarteschlange. Wenn Vorablesefunktionen verfügbar werden, implementieren sie diese Anforderungen. Dabei verwenden sie Eingabeverfahren wie das Lesen großer Blöcke (Big-Block Read) oder gestreutes Lesen (Scatter Read), um die angeforderten Seiten vom Plattenspeicher in den Pufferpool zu laden. Wenn Sie mehrere Plattendatenträger zum Speichern der Datenbankdaten haben, können die Daten über die Plattendatenträger einheitenübergreifend gespeichert werden (Striping). Durch dieses einheitenübergreifende Lesen und Schreiben von Daten können Vorablesefunktionen mehrere Festplatten gleichzeitig zum Abrufen von Daten verwenden.
- **Seitenlöschfunktionen** speichern Daten aus dem Pufferpool zurück auf den Plattenspeicher. Seitenlöschfunktionen sind im Hintergrund aktive EDUs, die unabhängig von den Anwendungsagenten arbeiten. Sie suchen nach Seiten, die geändert wurden, und schreiben diese geänderten Seiten auf die Platte. Seitenlöschfunktionen stellen sicher, dass im Pufferpool Platz für Seiten ist, die von Vorablesefunktionen eingelesen werden.

Ohne diese unabhängigen EDUs für Vorablesezugriffe und Seitenlöschfunktionen müssten Anwendungsagenten die gesamten Lese- und Schreiboperationen für Daten zwischen dem Pufferpool und dem Plattenspeicher selbst durchführen.

Das DB2-Prozessmodell

Wenn Sie mit dem DB2-Prozessmodell vertraut sind, kann dieses Wissen Ihnen bei der Bestimmung des Fehlertyps behilflich sein, da Sie dann verstehen, wie der Datenbankmanager mit den zugehörigen Komponenten interagiert.

Das Prozessmodell, das von allen DB2-Servern verwendet wird, erleichtert die Kommunikation, die zwischen Datenbankservern und Clients und lokalen Anwendungen stattfindet. Es stellt zudem sicher, dass Datenbankanwendungen von Ressourcen wie Datenbanksteuerblöcken und kritischen Datenbankdateien isoliert werden.

Der DB2-Server hat viele verschiedene Tasks auszuführen. Er muss zum Beispiel Anwendungsanforderungen verarbeiten oder sicherstellen, dass Protokollsätze auf die Platte geschrieben werden. Jede Task wird in der Regel durch eine separate Engine-Dispatchable-Unit (EDU, von der Steuerkomponente zuteilbare Einheit) ausgeführt. In früheren Releases wurden die meisten EDUs durch separate Prozesse in Linux- und UNIX-Umgebungen bzw. durch Betriebssystemsthreads innerhalb des Hauptprozesses des DB2-Servers unter Windows implementiert. Ab Version 9.5 arbeitet der DB2-Server in Linux- und UNIX-Umgebungen ebenfalls mit Threads, sodass EDUs jetzt unter UNIX und Windows durch Betriebssystemthreads implementiert werden.

Die Verwendung einer Multithread-Architektur im DB2-Server bietet zahlreiche Vorteile. Ein neuer Thread erfordert weniger Speicher- und Betriebssystemressourcen als ein Prozess, da einige Betriebssystemressourcen von allen Threads innerhalb desselben Prozesses gemeinsam genutzt werden können. Auf einigen Plattformen ist darüber hinaus der Zeitaufwand von Kontextwechseln für Threads günstiger als für Prozesse, was zur einer Leistungsverbesserung führen kann. Der wesentlichste Vorteil besteht jedoch darin, dass die Verwendung eines Multithread-Modells auf allen Plattformen die Konfiguration des DB2-Servers vereinfacht, da es leichter ist, mehr EDUs als nötig zuzuordnen. Außerdem ist es möglich, Speicher dynamisch zuzuordnen, der von mehreren EDUs gemeinsam genutzt werden muss. (In einem prozessbasierten Modell hat eine EDU keinen Zugriff auf Speicher, der von einer anderen EDU zugeordnet wurde.) Weitere Details zu diesen Vorteilen finden Sie in den Abschnitten zur vereinfachten Speicherkonfiguration und zur Agenten- und Prozessmodellkonfiguration.

In früheren Releases konnte auf Linux- und UNIX-Systemen der Systembefehl 'ps' oder der Befehl 'db2_local_ps' zum Auflisten aller für DB2 aktiven EDUs verwendet werden. Ab Version 9.5 listen diese Befehle keine EDU-Threads innerhalb des Prozesses 'db2sysc' mehr auf. Alternativ können Sie jetzt den Befehl 'db2pd' mit der Option '-edus' verwenden, um alle aktiven EDU-Threads aufzulisten. Dieser Befehl funktioniert auf UNIX-Systemen und auf Windows-Systemen.

Für jede Datenbank, auf die zugegriffen wird, werden verschiedene EDUs gestartet, um die verschiedenen Datenbankaufgaben, wie Vorablesen, Kommunikation und Protokollierung, wahrzunehmen. Datenbankagenten sind eine spezielle Klasse von EDUs, die zur Verarbeitung von Anwendungsanforderungen für eine Datenbank erstellt werden.

Jede Clientanwendungsverbindung hat genau einen **Koordinatoragenten**, der mit der Datenbank arbeitet. Ein Koordinatoragent arbeitet im Auftrag einer Anwendung und kommuniziert mit anderen Agenten, indem er je nach Bedarf privaten Speicher, Interprozesskommunikation (IPC) oder Protokolle zur Fernkommunikation nutzt.

Die DB2-Architektur stellt eine **Firewall** bereit, die dafür sorgt, dass Anwendungen in einem anderen Adressraum als DB2 ausgeführt werden. Die Firewall schützt die Datenbank und den Datenbankmanager vor Anwendungen, gespeicherten Prozeduren und benutzerdefinierten Funktionen (UDFs). Eine Firewall wahrt die Integrität der Daten in den Datenbanken, weil sie verhindert, dass interne Puffer oder Dateien des Datenbankmanagers durch Programmierfehler in Anwendungen überschrieben werden. Die Firewall erhöht außerdem die Zuverlässigkeit, weil Anwendungsfehler den Datenbankmanager nicht zum Absturz bringen können.

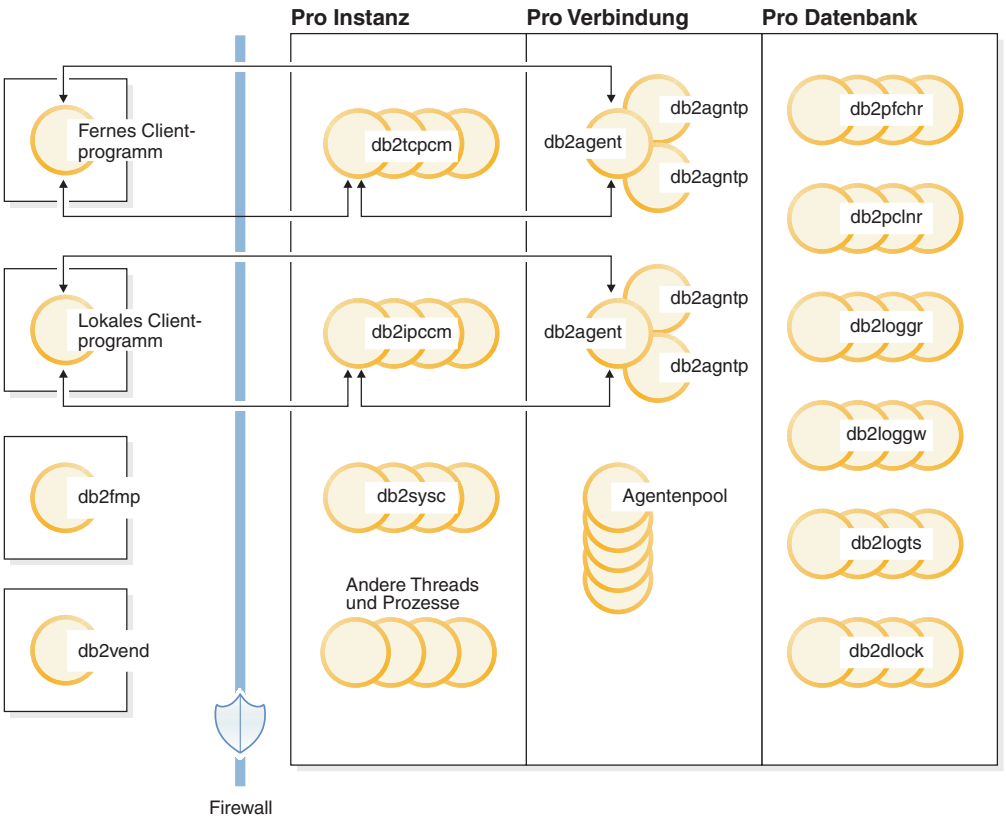


Abbildung 2. Prozessmodell für DB2-Systeme

Die folgende Liste enthält weitere Einzelheiten zu den in der Abbildung gezeigten Objekten:

Clientprogramme

Clientprogramme werden fern oder auf dem gleichen System wie der Datenbankserver ausgeführt. Sie stellen den ersten Kontakt mit der Datenbank über eine Listenerfunktion her. Anschließend wird ihnen ein Koordinatoragent (db2agent) zugeordnet.

Listenerfunktionen

Clientprogramme stellen den einleitenden Kontakt mit kommunikationsbereiten Agenten her, die gestartet werden, wenn DB2 gestartet wird. Für jedes Kommunikationsprotokoll ist eine Listenerfunktion konfiguriert. Eine Listenerfunktion für Interprozesskommunikation (db2ipccm) ist für lokale Clientprogramme konfiguriert. Folgende Listenerfunktionen sind vorhanden:

- **db2ipccm** für lokale Clientverbindungen

- **db2tcpcm** für TCP/IP-Verbindungen
- **db2tcpdm** für TCP/IP-Anforderungen des Tools Discovery

Agenten

Allen Verbindungsanforderungen von Clientanwendungen, d. h. lokalen und fernen, wird ein entsprechender Koordinatoragent (**db2agent**) zugeordnet. Wenn der Koordinatoragent erstellt ist, führt er alle Datenbankanforderungen im Auftrag der Anwendung aus.

In Umgebungen, in denen die Datenbankpartitionierungsfunktion (DPF) aktiviert ist, oder in Umgebungen, in denen die *abfrageinterne* Parallelität aktiviert ist, verteilt der Koordinatoragent die Datenbankanforderungen auf Subagenten (db2agntp bzw. db2agnts). Diese Agenten führen die Anforderungen für die Anwendung aus. Wenn der Koordinatoragent erstellt ist, verarbeitet er alle Datenbankanforderungen für seine Anwendung, indem er die Subagenten (db2agntp) koordiniert, die Anforderungen in der Datenbank ausführen. Subagenten, die zwar einer Anwendung zugeordnet, jedoch momentan inaktiv sind, werden durch den Namen **db2agnta** identifiziert.

Ein Koordinatoragent kann folgende Verbindungen haben:

- Eine Verbindung (CONNECT) zur Datenbank mit einem Aliasnamen. Zum Beispiel ist ″db2agent (DATA1)″ mit dem Datenbankaliasnamen ″DATA1″ verbunden.
- Eine Verbindung (ATTACH) zu einer Instanz. Zum Beispiel ist ″db2agent (user1)″ mit der Instanz ″user1″ verbunden.

Das DB2-Prozessmodell instanziiert weitere Typen von Agenten, um bestimmte Operationen auszuführen. Dies sind zum Beispiel unabhängige Koordinatoragenten oder Subkoordinatoragenten. Der unabhängige Koordinatoragent **db2agnti** wird beispielsweise verwendet, um Ereignismonitore auszuführen, während der Subkoordinatoragent **db2agnsc** dazu dient, die Ausführung des Neustarts einer Datenbank nach einer abrupten Beendigung zu parallelisieren.

Nicht zugeordnete Agenten befinden sich im Agentenpool. Diese Agenten sind für Anforderungen von Koordinatoragenten, die für Clientprogramme aktiv sind, oder für Anforderungen von Subagenten, die für vorhandene Koordinatoragenten aktiv sind, verfügbar. Eine geeignete Größe des Pools für inaktive Agenten kann die Leistung in Konfigurationen verbessern, die beträchtliche Anwendungsauslastungen aufweisen, da inaktive Agenten bei Bedarf sofort verfügbar sind und nicht erst ein völlig neuer Agent für jede Anwendungsverbindung zugeordnet werden muss, wobei Aufwand zur Erstellung eines Threads sowie zur Zuordnung und Initialisierung von Speicher und anderen Ressourcen anfiele. Ab Version 9.5 kann DB2 außerdem die Größe des Pools für inaktive Agenten automatisch verwalten, wenn dies erwünscht ist.

db2fmp

Der Prozess im abgeschirmten Modus. In seine Zuständigkeit fällt die Ausführung abgeschirmter gespeicherter Prozeduren und benutzerdefinierter Funktionen außerhalb der Firewall. Der Prozess db2fmp ist immer ein separater Prozess, kann jedoch je nach Art der ausgeführten Routinen mehrere Threads (Multithreading) enthalten.

db2vend

Dies ist ein Prozess zur Ausführung von Code anderer Anbieter im Auftrag einer EDU, zum Beispiel zur Ausführung des Benutzerexitprogramms für die Protokollarchivierung (nur UNIX).

Datenbank-EDUs

Die folgende Liste führt einige der wichtigen EDUs auf, die von jeder Datenbank verwendet werden:

- **db2pfchr** für Vorablesefunktionen für Pufferpools.
- **db2pclnr** für Seitenlöschfunktionen für Pufferpools.
- **db2loggr** zur Behandlung von Protokolldateien für die Durchführung der Transaktionsverarbeitung und -recovery.
- **db2loggw** zum Schreiben von Protokollsätzen in Protokolldateien.
- **db2logts** zur Verfolgung, welche Tabellenbereiche Protokollsätze in welchen Protokolldateien haben. Diese Informationen werden in der Datei DB2TSCHG.HIS im Datenbankverzeichnis aufgezeichnet. Sie dienen der Beschleunigung der aktualisierenden Recovery (Rollforward) von Tabellenbereichen.
- **db2dlock** zur Erkennung von Deadlocks. In einer Umgebung mit partitionierten Datenbanken wird ein zusätzlicher Thread **db2glock** zur Koordinierung der Informationen verwendet, die aus den db2dlock-EDUs der einzelnen Partitionen gesammelt werden. Der Thread 'db2glock' wird nur in der Katalogtabellenpartition ausgeführt.
- **db2stmm** für die automatische Speicheroptimierungsfunktion.
- **db2taskd** zur Verteilung von im Hintergrund ausgeführten Datenbanktasks. Die Tasks werden durch Threads mit dem Namen **db2taskp** ausgeführt.
- **db2hadrp** ist ein Thread für den primären HADR-Server.
- **db2hadrs** ist ein Thread für den HADR-Bereitschaftsserver.
- **db2lfr** für Protokolldatei-Leseeinheiten, die einzelne Protokolldateien verarbeiten.
- **db2shred** verarbeitet einzelne Protokollsätze in Protokollseiten.
- **db2redom** für den Master für aktualisierende Wiederherstellung. Bei der Recovery werden Protokollsätze für die aktualisierende Wiederherstellung verarbeitet und Protokollsätze Worker-Threads für die aktualisierende Wiederherstellung zugeordnet.
- **db2redow** für den Worker für aktualisierende Wiederherstellung. Bei der Recovery werden Protokollsätze für die aktualisierende Wiederherstellung bei der Anforderung des Masters für aktualisierende Wiederherstellung verarbeitet.
- **db2logmgr** für den Protokollmanager. Verwaltet Protokolldateien für eine wiederherstellbare Datenbank.
- **db2wlmd** für die automatische Erfassung von Auslastungsmanagementstatistiken.
- Ereignismonitorthreads werden wie folgt angegeben:
  - **db2evm%1%2 (%3)**: Dabei kann **%1** für Folgendes stehen:
    - **g** - globaler Dateiereignismonitor
    - **l** - lokaler Dateiereignismonitor
    - **t** - Tabellenereignismonitor
    - **gp** - globaler, über eine Pipe geleiteter Ereignismonitor

- **lp** - lokaler, über eine Pipe geleiteter Ereignismonitor

%2 kann für Folgendes stehen:

- **i** - Koordinator
- **p** - kein Koordinator

und **%3** ist der Name des Ereignismonitors.

- Backup- und Restore-Threads werden wie folgt angegeben:
  - **db2bm.%1.%2**: Puffermanipulator für Backup und Restore, und **db2med.%1.%2**: Mediencontroller für Backup und Restore. Dabei gilt Folgendes:
    - **%1** - Die EDU-ID des Agenten, der die Backup- bzw. Restoresitzung steuert.
    - **%2** - Ein sequenzieller Wert, der zur eindeutigen Unterscheidung der (möglicherweise vielen) Threads verwendet wird, die zu einer bestimmten Backup- bzw. Restoresitzung gehören.

    Beispiel: `db2bm.13579.2` gibt den zweiten Thread 'db2bm' an, der durch den Thread 'db2agent' mit der EDU-ID 13579 gesteuert wird.

Threads und Prozesse des Datenbankservers

Der Systemcontroller (**UNIX: db2sysc, Windows: db2syscs.exe**) muss vorhanden sein, damit der Datenbankserver funktioniert. Darüber hinaus können die folgenden Threads und Prozesse gestartet werden, um verschiedene Aufgaben zu übernehmen:

- **db2resync** ist der Resynchronisationsagent, der die globale Resynchronisationsliste (resync) durchsucht.
- **db2wdog** ist der Wachprozess unter UNIX- und Linux-Betriebssystemen, der abnormale Beendigungen behandelt.
- **db2fcms** ist der FCM-Senderdämon.
- **db2fcmr** ist der FCM-Empfängerdämon.
- **db2pdbc** ist der parallele Systemcontroller, der Parallelanforderungen von fernen Knoten verarbeitet (nur in einer partitionierten Datenbankumgebung).
- **db2cart** dient zur Archivierung von Protokolldateien, wenn auf eine Datenbank zugegriffen wird, für die der Benutzerexit (USEREXIT) aktiviert ist.
- **db2fmtlg** dient zur Formatierung von Protokolldateien, wenn auf eine Datenbank zugegriffen wird, in deren Konfiguration LOGRETAIN aktiviert, USEREXIT jedoch inaktiviert ist.
- **db2panic** ist der Notfallagent, der dringende Anforderungen verarbeitet, nachdem Agentengrenzwerte auf einem bestimmten Knoten erreicht wurden (nur in einer partitionierten Datenbankumgebung).
- **db2srvlst** verwaltet Listen mit Adressen für Systeme, wie z. B. DB2 für z/OS.
- **db2fmd** ist der Fehlermonitordämon.
- **db2disp** ist der Dispatcher für den Clientverbindungskonzentrator.
- **db2acd** ist der Autonomic Computing-Dämon, der den Host für die Dienstprogramme für den Diagnosemonitor und zur automatischen Pflege darstellt. Dieser Prozess hieß früher **db2hmon**.
- **db2licc** verwaltet die installierten DB2-Lizenzen.
- **db2thcln** macht Ressourcen wieder verfügbar, wenn eine EDU beendet wird (nur UNIX).
- **db2aiothr** verwaltet asynchrone E/A-Anforderungen für die Datenbankpartition (nur UNIX).

- **db2alarm** benachrichtigt EDUs, wenn der angeforderte Zeitgeber abgelaufen ist (nur UNIX).
- **db2sysc** ist die Hauptsystemcontroller-EDU, die kritische Ereignisse des DB2-Servers verarbeitet.

Deadlocks

Ein Deadlock entsteht, wenn zwei Anwendungen Daten sperren, die die jeweils andere Anwendung benötigt. Daraufhin kommt es zu einer Situation, in der weder die eine noch die andere Anwendung ihre Ausführung fortsetzen kann. Beispiel: Im folgenden Diagramm werden zwei Anwendungen gleichzeitig ausgeführt: Anwendung A und Anwendung B. Der erste Schritt von Anwendung A ist die Aktualisierung der ersten Zeile von Tabelle 1. Der zweite Schritt ist die Aktualisierung der zweiten Zeile von Tabelle 2. Anwendung B aktualisiert zuerst die zweite Zeile von Tabelle 2 und anschließend die erste Zeile von Tabelle 1. Zu einem bestimmten Zeitpunkt T1 führt Anwendung A den ersten Schritt aus, wobei die erste Zeile von Tabelle 1 zur Aktualisierung gesperrt wird. Gleichzeitig sperrt Anwendung B die zweite Zeile in Tabelle 2, um eine Aktualisierung durchzuführen. Zum Zeitpunkt T2 versucht Anwendung A den nächsten Schritt auszuführen und fordert eine Sperre für die zweite Zeile in Tabelle 2 für eine Aktualisierung an. Allerdings versucht Anwendung B gleichzeitig, die erste Zeile in Tabelle 1 zu sperren und zu aktualisieren. Da Anwendung A ihre Sperre für die erste Zeile von Tabelle 1 erst entriegeln wird, wenn sie eine Aktualisierung der zweiten Zeile in Tabelle 2 ausführen kann, und Anwendung B ihre Sperre für die zweite Zeile in Tabelle 2 erst entriegeln wird, wenn sie die erste Zeile von Tabelle 1 sperren und aktualisieren kann, tritt ein Deadlock auf. Die Anwendungen können unbegrenzt darauf warten, dass eine der Anwendungen die aktive Sperre für die Daten freigibt.

Deadlock-Konzept

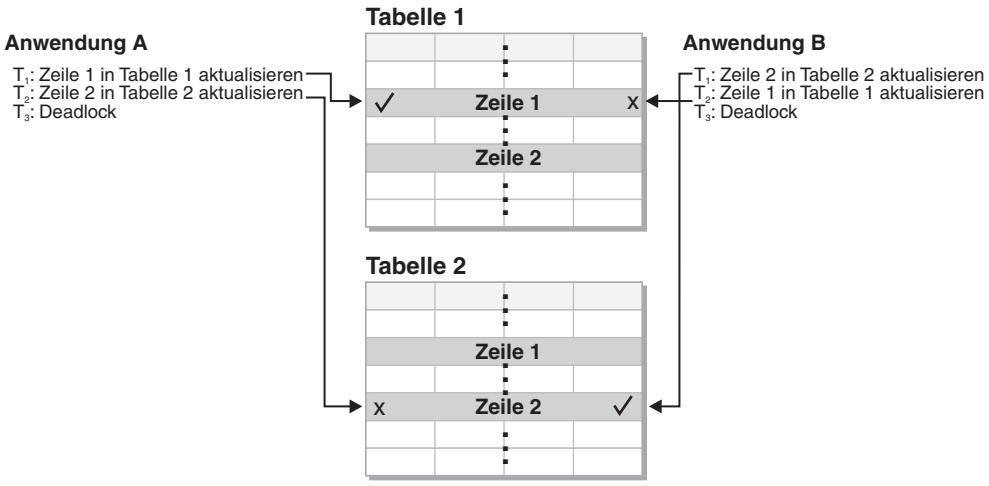


Abbildung 3. Deadlock zwischen Anwendungen

Da Anwendungen Sperren für Daten, die sie benötigen, nicht von sich aus freigeben, ist ein Detektorprozess erforderlich, der Deadlocks auflöst und eine Fortsetzung der Anwendungsverarbeitung ermöglicht. Wie der Name bereits andeutet, überwacht ein Deadlock-Detektor die Informationen zu Agenten, die an Sperren warten. Der Deadlock-Detektor wird in Intervallen aktiviert, die durch den Konfigurationsparameter *dlchktime* festgelegt werden.

Wenn der Deadlock-Detektor einen Deadlock ermittelt, definiert er einen der Prozesse im Deadlock nach dem Zufallsprinzip als *ausgewählten Prozess*, für den ein Rollback durchgeführt werden muss. Der ausgewählte Prozess wird aktiviert und gibt den SQLCODE -911 (SQLSTATE 40001) mit Ursachencode 2 an die aufrufende Anwendung zurück. Der Datenbankmanager macht nicht festgeschriebene Transaktionen aus dem ausgewählten Prozess automatisch rückgängig (ROLLBACK). Wenn der Rollback beendet ist, werden die Sperren freigegeben, die zu dem betroffenen Prozess gehörten, und die anderen Prozesse, die ebenfalls am Deadlock beteiligt waren, können fortfahren.

Zur Gewährleistung einer guten Leistung müssen Sie ein geeignetes Intervall für den Deadlock-Detektor auswählen. Ein zu kurzes Intervall verursacht einen unnötigen Systemaufwand, ein zu langes Intervall verlängert die Verzögerung eines Prozesses durch einen Deadlock auf ein nicht tolerierbares Maß. Zum Beispiel lässt ein Aktivierungsintervall von fünf Minuten möglicherweise zu, dass ein Deadlock annähernd fünf Minuten bestehen kann, was für eine kurze Transaktionsverarbeitung bereits ein langer Zeitraum sein kann. Es ist wichtig, die möglichen Verzögerungen durch die Auflösung von Deadlocks gegen den Aufwand ihrer Erkennung abzuwägen.

Anmerkung:

1. In einer Umgebung mit partitionierten Datenbanken wird das durch den Konfigurationsparameter *dlchktime* definierte Intervall nur auf den Katalogknoten angewendet. Werden in einer Umgebung mit partitionierten Datenbanken zahlreiche Deadlocks festgestellt, erhöhen Sie den Wert des Parameters *dlchktime*, um den Wartestatus für Sperren und die Übertragung Rechnung zu tragen.
2. In einer partitionierten Datenbank sendet jede Datenbankpartition *Sperrendiagramme* an die Datenbankpartition, die die Systemkatalogsichten enthält. In dieser Datenbankpartition findet eine umfassende Deadlock-Erkennung statt.

Ein anderes Problem tritt auf, wenn eine Anwendung mit mehr als einem unabhängigen Prozess, der auf die Datenbank zugreift, so strukturiert ist, dass die Entstehung von Deadlocks wahrscheinlich ist. Ein Beispiel wäre eine Anwendung, in der mehrere Prozesse auf dieselbe Tabelle zuerst zu Leseoperationen und anschließend zu Schreiboperationen zugreifen. Wenn die Prozesse schreibgeschützte SQL- oder XQuery-Abfragen durchführen und anschließend SQL-Aktualisierungen für dieselbe Tabelle verarbeiten, steigt die Wahrscheinlichkeit von Deadlocks, weil es zwischen den Prozessen zu Konkurrenzsituationen beim Zugriff auf dieselben Daten kommen kann. Wenn zum Beispiel zwei Prozesse die Tabelle lesen und anschließend aktualisieren, könnte Prozess A versuchen, eine Sperre des Modus X für eine Zeile zu erhalten, für die Prozess B eine Sperre des Modus S hat. Zur Vermeidung solcher Deadlocks sollten Anwendungen, die auf Daten zugreifen, um diese zu ändern, auf eine der folgenden Arten verfahren:

- Verwenden der Klausel FOR UPDATE OF bei der Ausführung einer SELECT-Anweisung. Durch diese Klausel wird sichergestellt, dass eine Sperre des Modus U aktiviert wird, wenn der Prozess A versucht, die Daten zu lesen. Die Zeilenblockung ist inaktiviert.
- Verwenden der Klausel WITH RR USE AND KEEP UPDATE LOCKS oder der Klausel WITH RS USE AND KEEP UPDATE LOCKS bei der Ausführung der Abfrage. Durch beide Klauseln wird sichergestellt, dass eine Sperre des Modus U aktiviert wird, wenn der Prozess A versucht, die Daten zu lesen, wobei Zeilenblockung zulässig ist.

Bei der Erstellung einer Datenbank wird gleichzeitig ein detaillierter Ereignismonitor für Deadlocks erstellt. Wie bei jedem Monitor fällt für diesen Ereignismonitor etwas Systemaufwand an.

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 der Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

Wenn Sie den detaillierten Ereignismonitor für Deadlocks entfernen möchten, kann er mit dem folgenden Befehl gelöscht werden:

```
DROP EVENT MONITOR db2detaildeadlock
```

In einer föderierten Systemumgebung, in der eine Anwendung auf Kurznamen zugreift, ist es möglich, dass die von der Anwendung angeforderten Daten aufgrund eines Deadlocks an einer Datenquelle nicht verfügbar sind. In diesem Fall ist DB2 von den Einrichtungen zur Behandlung von Deadlocks an der Datenquelle abhängig. Wenn Deadlocks in mehreren Datenquellen auftreten, ist DB2 für die Auflösung der Deadlocks auf die Zeitlimitmechanismen der Datenquellen angewiesen.

Wenn Sie mehr Informationen über Deadlocks protokollieren wollen, setzen Sie den Konfigurationsparameter *diaglevel* des Datenbankmanagers auf den Wert 4. Die protokollierten Informationen geben das gesperrte Objekt, den Sperrmodus und die Anwendung an, die die Sperre aktiviert hat. Die aktuellen dynamischen SQL- und XQuery-Anweisung bzw. die Namen der statischen Pakete können ebenfalls protokolliert werden. Dynamische SQL- und XQuery-Anweisungen werden nur protokolliert, wenn *diaglevel* gleich vier ist.

Standard-Deadlock-Ereignismonitor

Bei der Erstellung einer Datenbank wird standardmäßig ein Deadlock-Ereignismonitor mit dem Namen DB2DETAILDEADLOCK erstellt und aktiviert. Er wird automatisch gestartet, wenn die Instanz gestartet wird. Wenn dieser Monitor aktiv ist, werden Diagnoseinformationen beim ersten Vorkommen eines Deadlocks erfasst, sodass eine Untersuchung der Ursache möglich ist, ohne den Deadlock reproduzieren zu müssen.

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 der Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

Der Ereignismonitor wird mit der folgenden Anweisung erstellt:

```
db2 create event monitor db2detaildeadlock for deadlocks with details write to file
'db2detaildeadlock' maxfiles 20 maxfilesize 512 buffersize 17 blocked append autostart
```

Durch die Klausel WITH DETAILS werden Informationen bereitgestellt, wie zum Beispiel die Anweisung, die gerade ausgeführt wurde, als der Deadlock auftrat, und die Sperrenliste (sofern im Zwischenspeicher für den Datenbankmonitor genügend Speicherbereich verfügbar ist).

Die Ausgabedateien werden unter dem Verzeichnis 'db2event' in Ihrem Datenbankverzeichnis erstellt. Wenn Sie beim Erstellen der Datenbank keine Position angeben, kann die Position des Datenbankverzeichnisses dem Konfigurationsparameter *dftdbpath* des Datenbankmanagers entnommen werden. Zum Beispiel können sich die Ausgabedateien des Ereignismonitors in einer Beispieldatenbank unter AIX® im Verzeichnis `NODE0000/SQL00001/db2event/db2detaildeadlock` befinden.

Der Ereignismonitor schreibt in Dateien bis zu einer maximalen Anzahl von 20 Dateien, wobei jede eine Größe von 2 MB (512 4-KB-Seiten) hat. Wenn die maximale Dateigröße (*maxfilesize*, 2 MB) erreicht wird, wird die Ausgabedatei geschlossen und eine neue geöffnet. Wenn die maximale Anzahl (*maxfiles*, 20) erstellter Dateien erreicht wird, beendet der Monitor sich selbst und eine Nachricht ähnlich der folgenden wird im Protokoll mit Benachrichtigungen für die Systemverwaltung aufgezeichnet:

```
2004-12-01-22.58.24.968000 Instance: DB2 Node: 000 PID: 1116(db2syscs.exe) TID: 2540
Appid: *LOCAL.DB2.041202080328 database monitor sqm__evmgr::log_ev_err Probe:2
Database:XXX ADM2001W Der Ereignismonitor "DB2DETAILDEADLOCK" wurde inaktiviert, da die
Grenzwerte der Parameter MAXFILES und MAXFILESIZE CREATE EVENT MONITOR erreicht wurden.
```

Durch das Entfernen von Ausgabedateien, die nicht mehr benötigt werden, kann der Monitor bei der nächsten Datenbankaktivierung erneut aktiviert werden. Wenn der detaillierte Ereignismonitor für Deadlocks nicht aktiviert sein soll, kann er mit dem folgenden Befehl gelöscht werden:

```
DROP EVENT MONITOR db2detaildeadlock
```

Wenn Sie das Auftreten von Deadlocks befürchten, löschen Sie diesen Monitor nicht.

Plattenspeicher - Übersicht

Leistungsfaktoren für Plattenspeicher

Die Hardware, aus der sich Ihr System zusammensetzt, kann die Leistung Ihres Systems beeinflussen. Als Beispiel für den Einfluss, den die Hardware auf die Leistung hat, werden im Folgenden einige der Auswirkungen betrachtet, die mit dem Plattenspeicher zusammenhängen.

Vier Aspekte der Plattenspeicherverwaltung können sich auf die Leistung auswirken:

- **Speichereinteilung**

  Wie Sie eine begrenzte Speichergröße zwischen Indizes und Daten sowie unter Tabellenbereichen aufteilen, bestimmt in hohem Maße, welche Leistung die einzelnen Komponenten in verschiedenen Situationen erreichen.
- **Speicherverschwendung**

  Verschwendeter Speicher wirkt sich vielleicht als solcher nicht auf die Leistung des Systems aus, das ihn besitzt, aber er ist eine Ressource, die zur Verbesserung der Leistung an anderer Stelle genutzt werden könnte.
- **Verteilung der Platten-E/A**

  Wie ausgewogen Sie den Bedarf an Platten-E/A auf mehrere Plattenspeichereinheiten und Controller verteilen, kann sich auf die Geschwindigkeit auswirken, mit der der Datenbankmanager Informationen von Platten abrufen kann.

- **Mangel an verfügbarem Speicher**
  Bei Erreichen der Grenze des verfügbaren Speichers kann die allgemeine Leistung beeinträchtigt werden.

Teil 2. Tabellen und Indizes

Kapitel 6. Tabellen- und Indexverwaltung für Standardtabellen

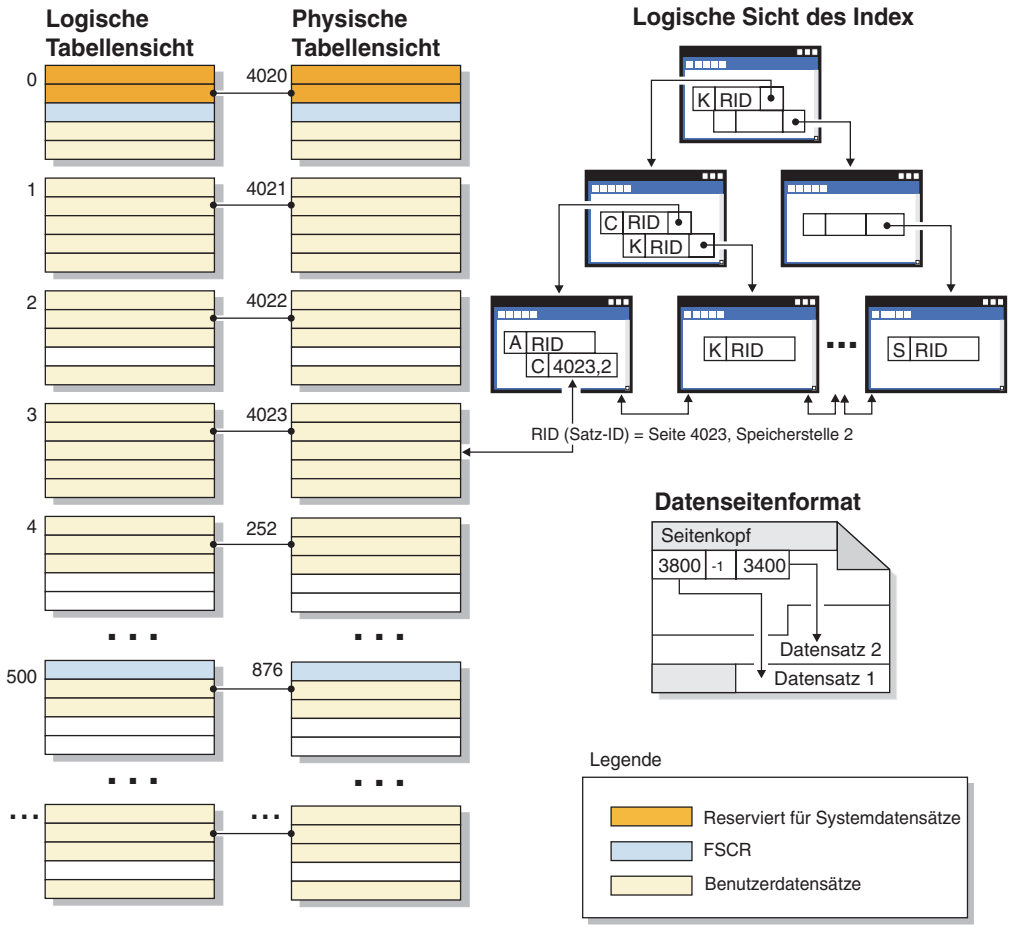


Abbildung 4. Logische Tabellen-, Datensatz- und Indexstruktur für Standardtabellen

In Standardtabellen werden Tabellendaten logisch in Form einer Liste von Datenseiten organisiert. Diese Datenseiten werden logisch auf der Grundlage der EXTENTSIZE-Größe des Tabellenbereichs zu Gruppen zusammengefasst. Wenn die EXTENTSIZE-Größe beispielsweise vier beträgt, sind die Seiten null bis drei Teil des ersten EXTENTSIZE-Speicherbereichs, die Seiten vier bis sieben sind Teil des zweiten usw.

Die Zahl der Datensätze, die in den einzelnen Datenseiten enthalten sind, kann je nach Größe der Datenseite und Größe der Datensätze variieren. Die maximale Anzahl von Datensätzen, die auf eine Seite passen, ist der Tabelle 1 zu entnehmen. Die meisten Seiten enthalten nur Benutzersätze. Eine kleine Zahl von Seiten enthält jedoch besondere interne Datensätze, die von DB2 zur Verwaltung der Tabelle verwendet werden. Auf jeder 500sten Seite befindet sich beispielsweise ein FSCR-Datensatz (Free Space Control Record, Steuersatz für freien Speicherbereich). Diese Datensätze enthalten eine Zuordnung des freien Speicherbereichs für neue Datensätze auf jeder der folgenden 500 Datenseiten (bis zum nächsten FSCR-Datensatz). Dieser verfügbare freie Speicherbereich wird verwendet, wenn Datensätze in die Tabelle eingefügt werden.

Logisch werden Indexseiten als B-Baumstruktur organisiert, durch die Datensätze in der Tabelle, die einen angegebenen Schlüsselwert besitzen, auf effiziente Weise lokalisiert werden können. Die Zahl der Entitäten auf einer Indexseite ist nicht festgelegt, sondern hängt von der Größe des Schlüssels ab. Bei Tabellen in DMS-Tabellenbereichen verwenden RIDs (Record Identifiers, Satz-IDs) auf den Indexseiten Seitennummern, die nicht zum Objekt, sondern zum Tabellenbereich relativ sind. Dadurch kann bei einer **Indexsuche** direkt auf die Datenseiten zugegriffen werden, ohne dass eine Speicherbereichsmaske (EMP) für die Zuordnung erforderlich ist.

Jede Datenseite besitzt dasselbe Format. Alle Datenseiten beginnen mit einem Seitenkopf. Nach dem Seitenkopf folgt ein Speicherstellenverzeichnis. Jeder Eintrag im Speicherstellenverzeichnis entspricht einem anderen Datensatz auf der Seite. Der Eintrag selbst ist die relative Byteadresse auf der Datenseite, an der der Datensatz beginnt. Einträge mit minus eins (-1) entsprechen gelöschten Datensätzen.

Satz-IDs und Seiten

Bei Satz-IDs (Record Identifier, RID) handelt es sich um eine Seitennummer gefolgt von einer Bereichsnummer (slot). Indexdatensätze des Typs 2 enthalten ein zusätzliches Feld mit der Bezeichnung ridFlag. Die ridFlag-Markierung speichert Informationen über den Status von Schlüsseln im Index, zum Beispiel ob dieser Schlüssel als gelöscht markiert wurde. Wenn der Index zur Identifizierung einer Satz-ID verwendet wird, wird diese Satz-ID dazu verwendet, zur richtigen Datenseite und Speicherstellennummer auf dieser Seite zu gelangen. Wenn dem Datensatz eine Satz-ID zugeordnet ist, wird diese erst wieder bei einer Reorganisation der Tabelle geändert.

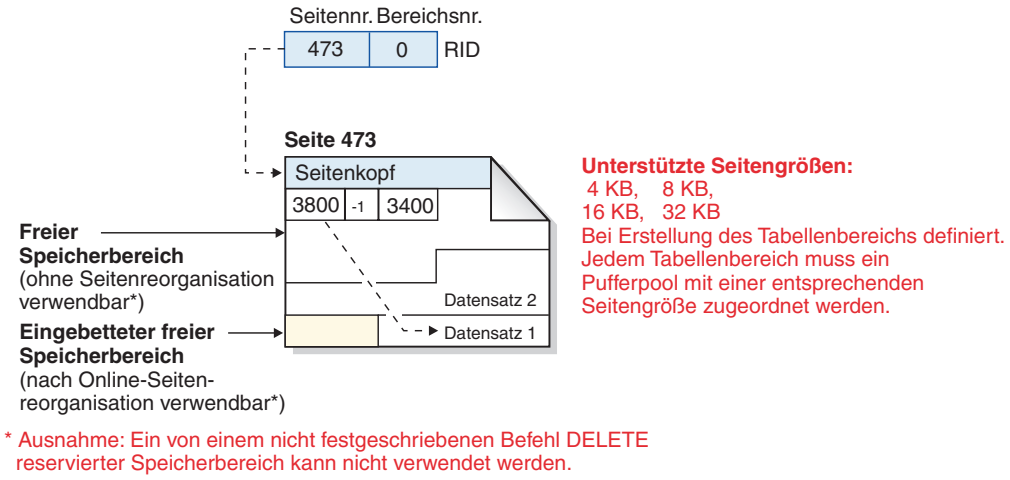


Abbildung 5. Format der Datenseite und der Satz-ID (RID)

Wenn eine Tabellenseite reorganisiert wird, wird eingebetteter freier Speicherbereich, der nach dem physischen Löschen eines Datensatzes auf der Seite verbleibt, in verwendbaren freien Speicherbereich umgewandelt. Satz-IDs werden durch Verschieben von Datensätzen auf einer Datenseite erneut definiert, damit der verwendbare freie Speicherbereich genutzt werden kann.

DB2 unterstützt verschiedene Seitengrößen. Verwenden Sie größere Seiten für Auslastungen, bei denen auf Zeilen normalerweise sequenziell zugegriffen wird. Sequenzieller Zugriff wird beispielweise für Anwendungen zur Entscheidungshilfe

oder in Fällen verwendet, in denen ein starker Gebrauch von temporären Tabellen gemacht wird. Verwenden Sie geringere Seitengrößen für Auslastungen, deren Zugriff normalerweise wahlfrei erfolgt. Wahlfreier Zugriff wird beispielsweise in OLTP-Umgebungen verwendet.

Indexverwaltung in Standardtabellen

DB2-Indizes verwenden eine optimierte B-Baumstrukurimplementierung auf der Basis einer effizienten Indexverwaltungsmethode mit einem hohen Grad des gemeinsamen Zugriffs und einer im Voraus schreibenden Protokollierung (Write Ahead Logging).

Die optimierte B-Baumstrukturimplementierung verfügt über bidirektionale Zeiger auf den Blattseiten, mit denen ein einzelner Index sowohl vorwärts als auch rückwärts gerichtete Suchoperationen unterstützen kann. Indexseiten werden normalerweise in der Mitte geteilt, wobei die HIGHKEY-Seite eine Ausnahme bildet, bei der eine 90/10-Teilung erfolgt. Das heißt, dass die hohen zehn Prozent der Indexschlüssel auf der neuen Seite Platz finden. Diese Art der Indexseitenteilung ist bei Auslastungen nützlich, bei denen oft INSERT-Anforderungen mit neuen HIGHKEY-Werten durchgeführt werden.

Seit Version 8.1 arbeitet DB2 mit Indizes des Typs 2. Wenn Sie eine Migration von früheren Versionen von DB2 durchführen, werden sowohl Indizes des Typs 1 als auch Indizes des Typs 2 verwendet, bis Sie die Indizes reorganisieren oder andere Aktionen durchführen, die Indizes des Typs 1 in Indizes des Typs 2 konvertieren. Der Indextyp bestimmt, wie gelöschte Schlüssel physisch von den Indexseiten entfernt werden.

- Für Indizes des Typs 1 werden Schlüssel von den Indexseiten beim Löschen der Schlüssel entfernt und Indexseiten freigegeben, wenn der letzte Indexschlüssel von der Seite entfernt ist.
- Für Indizes des Typs 2 werden Indexschlüssel beim Löschen des Schlüssels nur dann von der Seite entfernt, wenn eine exklusive Sperre (Modus X) für die Tabelle aktiviert ist. Wenn Schlüssel nicht sofort entfernt werden können, werden sie als gelöscht markiert und später entfernt. Weitere Informationen enthält der Abschnitt zu Indizes des Typs 2.

Wenn Sie die Online-Indexdefragmentierung aktiviert haben, indem Sie bei der Erstellung des Index die Klausel MINPCTUSED auf einen höheren Wert als null gesetzt haben, können Blattseiten online zusammengefügt werden. Der von Ihnen angegebene Wert ist der Schwellenwert für den minimalen Prozentsatz an Speicherplatz, der auf den Indexblattseiten genutzt wird. Wenn ein Schlüssel von einer Indexseite entfernt wurde und der Prozentsatz an Speicherplatz auf der Seite bei oder unter dem angegebenen Wert liegt, versucht der Datenbankmanager die verbleibenden Schlüssel mit denen einer benachbarten Seite zu einer Seite zusammenzufügen. Wenn genügend Platz vorhanden ist, wird die Operation zum Zusammenfügen ausgeführt und eine Indexseite gelöscht. Die Online-Indexdefragmentierung kann die Wiederverwendung von Speicherbereich verbessern. Wenn jedoch der Wert für MINPCTUSED zu hoch ist, wird mehr Zeit auf Versuche verwendet, Seiten zusammenzufügen, die jedoch mit geringerer Wahrscheinlichkeit erfolgreich sind. Der empfohlene Wert für diese Klausel ist 50 % oder weniger.

Anmerkung: Da die Onlinedefragmentierung nur stattfindet, wenn Schlüssel von einer Indexseite entfernt werden, findet sie in einem Index des Typs 2 nicht statt, wenn Schlüssel nur als gelöscht markiert, jedoch nicht physisch von der Seite entfernt wurden.

Die Klausel INCLUDE der Anweisung CREATE INDEX ermöglicht das Einfügen einer angegebenen Spalte bzw. von Spalten auf den Indexseiten zusätzlich zu den Schlüsselspalten. Dadurch kann die Zahl der Abfragen steigen, bei denen ein reiner Indexzugriff möglich ist. Gleichzeitig können jedoch die Anforderungen an Indexspeicherplatz und möglicherweise auch der Verwaltungsaufwand für den betreffenden Index steigen, wenn die eingefügten Spalten häufig aktualisiert werden. Der Verwaltungsaufwand zur Aktualisierung von INCLUDE-Spalten ist geringer als der zur Aktualisierung von Schlüsselspalten, jedoch höher als der zur Aktualisierung von Spalten, die nicht im Index vertreten sind. Das Ordnen der Index-B-Baumstruktur geschieht nur unter Verwendung der Schlüsselspalten, nicht der eingeschlossenen Spalten.

Kapitel 7. Tabellen- und Indexverwaltung für MDC-Tabellen

Die Tabellen- und Indexorganisation für Tabellen mit mehrdimensionalem Clustering (MDC) basiert auf den gleichen logischen Strukturen wie die Organisation von Standardtabellen. Ebenso wie Standardtabellen werden MDC-Tabellen in Seiten organisiert, die Datenzeilen enthalten, die wiederum in Spalten unterteilt sind. Die Zeilen jeder Seite werden durch Zeilen-IDs (RIDs) gekennzeichnet. Darüber hinaus werden die Seiten von MDC-Tabellen in EXTENTSIZE große Blöcke gruppiert. In der folgenden Abbildung, die eine Tabelle mit dem EXTENTSIZE-Wert 4 darstellt, bilden die ersten vier Seiten mit den Nummern 0 bis 3 den ersten Block in der Tabelle. Die nächste Gruppe von Seiten mit den Nummern 4 bis 7 bildet den zweiten Block der Tabelle.

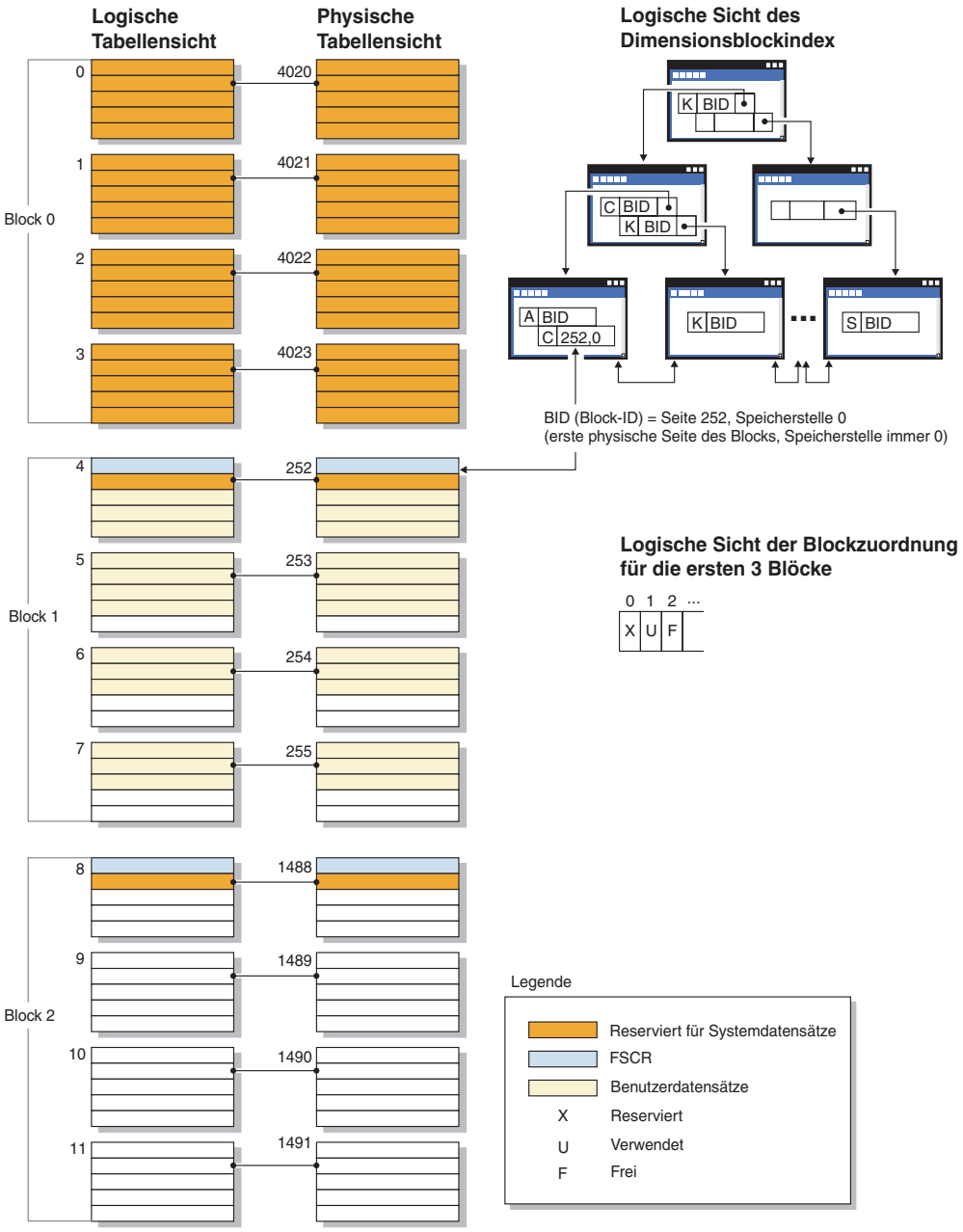


Abbildung 6. Logische Tabellen-, Datensatz- und Indexstruktur für MDC-Tabellen

Der Block enthält besondere interne Datensätze, die von DB2 zur Verwaltung der Tabelle verwendet werden. Dazu gehören auch die FSCR-Sätze, d. h., die Steuerdatensätze für freie Speicherbereiche (FSCR - Free-Space Control Record). In den nachfolgenden Blöcken enthält jeweils die erste Seite die FSCR-Sätze. Ein FSCR-Satz ordnet den freien Speicher für neue Datensätze zu, der auf den jeweiligen Seiten im Block vorhanden ist. Dieser verfügbare freie Speicherbereich wird verwendet, wenn Datensätze in die Tabelle eingefügt werden.

Wie am Namen ersichtlich, ordnen MDC-Tabellen ihre Daten in mehr als einer Dimension in so genannten Clustern (d. h. in Datengruppen) an. Jede Dimension wird durch eine Spalte bzw. eine Gruppe von Spalten bestimmt, die Sie in der Klausel ORGANIZE BY DIMENSIONS der Anweisung CREATE TABLE angeben. Bei der Erstellung einer MDC-Tabelle werden die beiden folgenden Arten von Indizes automatisch erstellt:

- Ein Dimensionsblockindex, der Zeiger auf jeden belegten Block für eine einzelne Dimension enthält.
- Einen zusammengesetzten Blockindex, der alle Dimensionsschlüsselspalten enthält. Der zusammengesetzte Blockindex dient zur Erhaltung der Clusterbildung bei Einfüge- und Aktualisierungsaktivitäten.

Das Optimierungsprogramm zieht Zugriffspläne unter Nutzung von Dimensionsblockindizes in Betracht, wenn es den effizientesten Zugriffsplan für eine bestimmte Abfrage ermittelt. Wenn Abfragen Vergleichselemente für Dimensionswerte enthalten, kann das Optimierungsprogramm den Dimensionsblockindex verwenden, um die (EXTENTSIZE großen) Speicherbereiche, die diese Werte enthalten, zu ermitteln und aus diesen Daten abzurufen. Da diese Speicherbereiche physisch aufeinander folgende Seiten auf der Platte darstellen, führt dieses Verfahren zu einer effizienteren Leistung und minimiert den E/A-Aufwand.

Darüber hinaus können Sie auch spezielle Zeilen-ID-Indizes (RID-Indizes) erstellen, wenn die Analyse der Datenzugriffspläne darauf hinweist, dass solche Indizes die Abfrageleistung verbessern würden.

Neben den Dimensionsblockindizes und dem zusammengesetzten Blockindex pflegen MDC-Tabellen eine Blockzuordnung (engl. block map), die eine Bitmaske enthält, die den Verfügbarkeitsstatus jedes Blocks angibt. Die folgenden Attribute werden in der Liste der Bitmaske codiert:

- X (reserviert): Der erste Block enthält nur Systeminformationen für die Tabelle.
- U (in Gebrauch): Dieser Block wird verwendet und ist einem Dimensionsblockindex zugeordnet.
- L (geladen): In diesen Block wurden durch eine aktuelle Ladeoperation Daten geladen.
- C (Integritätsbedingung prüfen): Dieser Block wurde durch die Ladeoperation so eingestellt, dass er eine inkrementelle Prüfung auf Integritätsbedingungen während des Ladens angibt.
- T (Tabelle aktualisieren): Dieser Block wurde durch die Ladeoperation so eingestellt, dass er eine erforderliche Pflege von AST-Tabellen angibt.
- F (frei): Wenn kein anderes Attribut gesetzt ist, wird der Block als frei betrachtet.

Da jeder Block einen Eintrag in der Blockzuordnungsdatei hat, wächst die Datei, wenn die Tabelle größer wird. Diese Datei wird als separates Objekt gespeichert. In einem SMS-Tabellenbereich hat sie einen neuen Dateityp. In einem DMS-Tabellenbereich hat sie einen neuen Objektdeskriptor in der Objekttabelle.

Kapitel 8. 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 9. Indexstruktur

Der Datenbankmanager verwendet eine B+-Baumstruktur zur Indexspeicherung. Eine B+-Baumstruktur hat eine oder mehrere Stufen, wie die folgende Abbildung zeigt (`rid` steht für Satz/Zeilen-ID):

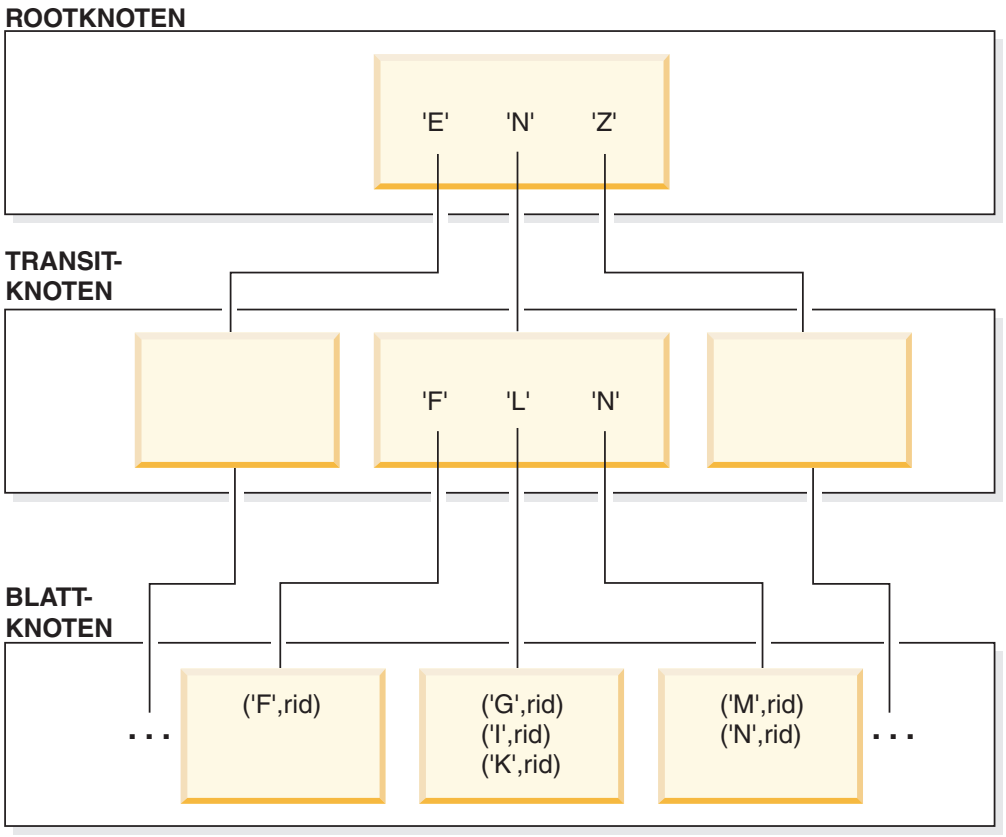


Abbildung 7. B+-Baumstruktur

Die oberste Stufe wird als *Rootknoten* bezeichnet. Die unterste Stufe besteht aus *Blattknoten*, in denen die Werte des Indexschlüssels jeweils mit Zeigern auf die Zeile in der Tabelle gespeichert sind, die den Schlüsselwert enthält. Die Stufen zwischen dem Rootknoten und dem Blattknoten werden als *Transitknoten* bezeichnet.

Bei der Suche nach einem bestimmten Indexschlüsselwert durchsucht der Indexmanager den Indexbaum ausgehend vom Rootknoten. Der Rootknoten enthält einen Schlüssel für jeden Knoten auf der folgenden Stufe. Der Wert jedes dieser Schlüssel ist jeweils der größte vorhandene Schlüsselwert für den entsprechenden Knoten auf der nächsten Stufe. Wenn zum Beispiel ein Index drei Stufen hat, wie in der Abbildung gezeigt, dann durchsucht der Indexmanager zum Auffinden eines Indexschlüsselwerts den Rootknoten nach dem ersten Schlüsselwert, der größer oder gleich dem gesuchten Schlüsselwert ist. Der Rootknotenschlüssel enthält dann einen Zeiger auf einen bestimmten Transitknoten. Der Indexmanager setzt dieses Verfahren durch die Transitknoten fort, bis er den Blattknoten findet, der den benötigten Indexschlüssel enthält.

In der Abbildung wird zum Beispiel nach dem Schlüssel „I" gesucht. Der erste Schlüssel im Rootknoten, der größer oder gleich „I" ist, ist „N". Dieser Wert zeigt auf den mittleren Knoten auf der nächsten Stufe. Der erste Schlüssel in diesem Transitknoten, der größer als oder gleich „I" ist, ist der Wert „L". Dieser Wert zeigt auf einen bestimmten Blattknoten, in dem der Indexschlüssel für „I" zusammen mit der entsprechenden Zeilen-ID (RID) gefunden wird. Die Zeilen-ID (oder Satz-ID) gibt die entsprechende Zeile in der Basistabelle an. Die Blattknotenstufe kann auch Zeiger auf frühere Blattknoten enthalten. Diese Zeiger geben dem Indexmanager die Möglichkeit, die Blattknoten in beide Richtungen zu durchsuchen, um einen Bereich von Werten abzurufen, nachdem er einen Wert des Bereichs gefunden hat. Die Möglichkeit, den Index in beide Richtungen zu durchsuchen, ist nur gegeben, wenn der Index mit der Klausel ALLOW REVERSE SCANS erstellt wurde.

Für MDC-Tabellen (MDC - Mehrdimensionales Clustering) wird automatisch ein Blockindex für jede Clusteringdimension erstellt, die Sie für die Tabelle angeben. Außerdem wird ein zusätzlicher zusammengesetzter Blockindex erstellt, der einen Schlüsselbestandteil für jede Spalte enthält, die an einer Dimension der Tabelle beteiligt ist. Solche Indizes enthalten Zeiger auf Block-IDs (BIDs) anstelle von Satz-IDs (RIDs) und bieten Verbesserungen beim Datenzugriff.

In DB2 Version 8.1 und späteren Versionen können Indizes den Typ 1 oder den Typ 2 aufweisen. Ein *Index des Typs 1* ist die ältere Indexdarstellung. Indizes, die in früheren Versionen von DB2 erstellt wurden, sind vom Typ 1.

Ein Index des Typs 2 ist etwas größer als ein Index des Typs 1 und verfügt über Zusatzfunktionen, die das Sperren des jeweils nächsten Schlüssels minimieren. Mit der ein Byte großen *ridFlag*-Markierung, die mit jeder Satz-ID auf der Blattseite eines Index des Typs 2 gespeichert wird, wird eine Satz-ID ggf. als logisch gelöscht markiert, sodass sie später physisch gelöscht werden kann. Für jede Spalte variabler Länge, die im Index enthalten ist, wird in einem zusätzlichen Byte die tatsächliche Länge des Spaltenwerts gespeichert. Indizes des Typs 2 können auch deshalb größer als Indizes des Typs 1 sein, weil einige Schlüssel als gelöscht markiert, jedoch noch nicht von der Indexseite physisch gelöscht wurden. Nach dem Commit der DELETE- oder UPDATE-Transaktion können die als gelöscht markierten Schlüssel bereinigt werden.

Teil 3. Prozesse

Kapitel 10. Verringern des Protokollierungsaufwands zur Verbesserung der Abfrageleistung

Alle Datenbanken pflegen Protokolldateien, die Datensätze über Datenbankänderungen aufzeichnen. Zwei Protokollierungsstrategien stehen zur Auswahl:

- Die Umlaufprotokollierung, bei der die Protokollsätze die Protokolldateien vollständig füllen und anschließend die ersten Protokollsätze in der ersten Protokolldatei überschreiben. Die überschriebenen Protokollsätze sind nicht wiederherstellbar.
- Behalten und Speichern der Protokollsätze, bei dem eine Protokolldatei archiviert wird, wenn sie mit Protokollsätzen gefüllt ist. Für weitere Protokollsätze werden neue Protokolldateien verfügbar gemacht. Die Speicherung der Protokolldateien ermöglicht eine **aktualisierende Recovery**. Die aktualisierende Recovery wendet Änderungen an der Datenbank auf der Basis der abgeschlossenen und im Protokoll aufgezeichneten UOWs (UOW = Unit of Work, Arbeitseinheit) bzw. Transaktionen erneut an. Sie können angeben, dass die aktualisierende Recovery bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt vor dem Ende der Protokolle durchgeführt wird.

Unabhängig von der Protokollierungsstrategie werden alle Änderungen an regulären Daten- und Indexseiten in den Protokollpuffer geschrieben. Die Daten im Protokollpuffer werden durch den Protokollprozess auf die Platte geschrieben. Unter folgenden Umständen muss die Abfrageverarbeitung darauf warten, dass Protokolldaten auf die Platte geschrieben werden:

- Bei einem Commit.
- Bevor die entsprechenden Datenseiten auf Platte geschrieben werden, da DB2 mit einer Vorabschreibprotokollierung (write ahead) arbeitet. Ein Vorteil der Vorabschreibprotokollierung besteht darin, dass bei Beendigung einer Transaktion durch Ausführen einer COMMIT-Anweisung nicht alle geänderten Daten und Indexseiten auf Platte geschrieben werden müssen.
- Bevor einige Änderungen an Metadaten vorgenommen werden, von denen sich die meisten aus der Ausführung von DDL-Anweisungen ergeben.
- Beim Schreiben von Protokollsätzen in den Protokollpuffer, wenn der Protokollpuffer voll ist.

DB2 verwaltet so das Schreiben von Protokolldaten auf Platte, um die Verarbeitungsverzögerung zu minimieren. In einer Umgebung, in der viele kurze gleichzeitige Transaktionen stattfinden, wird der größte Teil der Verarbeitungsverzögerung durch COMMIT-Anweisungen verursacht, die darauf warten müssen, dass Protokolldaten auf Platte geschrieben werden. Infolgedessen schreibt der Protokollprozess häufig kleine Mengen von Protokolldaten auf Platte, wobei eine zusätzliche Verzögerung durch den Systemaufwand für Protokoll-E/A entsteht. Um die Anwendungsantwortzeit gegen eine solche Protokollierungsverzögerung zu verbessern, setzen Sie den Datenbankkonfigurationsparameter *mincommit* auf einen höheren Wert als 1. Diese Einstellung kann zwar längere Verzögerungen für Commits aus einigen Anwendungen verursachen, jedoch können mehr Protokolldaten in einer Operation auf Platte geschrieben werden.

Änderungen an großen Objekten (LOBs) und LONG VARCHAR-Daten werden durch Erstellen einer Spiegelkopie für Speicherseiten (Shadow Paging) protokolliert. Änderungen an LOB-Spalten werden nur dann protokolliert, wenn Sie die

Beibehaltung der Protokolldateien (log_retain) angeben und die LOB-Spalte in der Anweisung CREATE TABLE ohne die Klausel NOT LOGGED definiert wurde. Änderungen an den Zuordnungsseiten für LONG- oder LOB-Datentypen werden wie reguläre Datenseiten protokolliert.

Kapitel 11. Verbessern der Einfügeleistung

Wenn SQL-Anweisungen mithilfe von INSERT neue Informationen in eine Tabelle einfügen, durch ein INSERT-Suchalgorithmus zunächst die FSCR-Sätze (Free Space Control Records), um eine Seite mit genügend Speicherplatz zu finden. Selbst wenn der FSCR jedoch angibt, dass auf einer bestimmten Seite genügend freier Speicherbereich vorhanden ist, kann dieser jedoch möglicherweise nicht verwendet werden, da er durch eine nicht festgeschriebene Anweisung DELETE von einer anderen Transaktion reserviert ist. Um sicherzustellen, dass nicht festgeschriebener freier Speicherplatz verwendbar ist, sollten Sie Transaktionen häufig durch COMMIT festschreiben.

Die Einstellung der Registrierdatenbankvariablen DB2MAXFSCRSEARCH bestimmt die Anzahl von FSCRs in einer Tabelle, die für eine INSERT-Anweisung durchsucht werden. Der Standardwert für diese Registrierdatenbankvariable ist fünf. Wenn in der angegebenen Anzahl von FSCRs kein Speicherplatz gefunden wird, wird der einzufügende Datensatz an das Ende der Tabelle angehängt. Zur Optimierung der Geschwindigkeit der Anweisung INSERT werden nachfolgende Datensätze ebenfalls an das Ende der Tabelle angehängt, bis zwei EXTENTSIZE-Speicherbereiche gefüllt sind. Wenn diese beiden Speicherbereiche gefüllt sind, setzt die nächste Anweisung INSERT die Suche bei dem FSCR fort, bei dem die letzte Suche beendet wurde.

Anmerkung: Setzen Sie die Registrierdatenbankvariable DB2MAXFSCRSEARCH auf einen kleinen Wert, um die INSERT-Geschwindigkeit, allerdings mit der möglichen Folge eines schnelleren Anwachsens von Tabellen, zu optimieren. Zur Optimierung der Wiederverwendung von Speicherplatz, möglicherweise zu Lasten der INSERT-Geschwindigkeit, setzen Sie die Registrierdatenbankvariable DB2MAXFSCRSEARCH auf einen höheren Wert.

Wenn alle FSCRs in der gesamten Tabelle auf diese Weise durchsucht wurden, werden die einzufügenden Datensätze ohne weiteres Suchen an das Ende angehängt. Es findet kein weiteres Suchen mithilfe der FSCRs mehr statt, sofern nicht Speicherplatz an einer Stelle in der Tabelle frei wird, wie zum Beispiel nach einer DELETE-Operation.

Es gibt zwei weitere INSERT-Algorithmusoptionen:

- Anhängemodus (APPEND MODE)

  In diesem Modus werden neue Zeilen immer an das Ende der Tabelle angehängt. Es finden weder Such- noch Pflegeoperationen für FSCRs statt. Diese Option wird mit der Anweisung ALTER TABLE APPEND ON aktiviert und kann die Leistung für ausschließlich wachsende Tabellen wie Journale verbessern.
- Ein Clusterindex ist für die Tabelle definiert.

  In diesem Fall versucht der Datenbankmanager, Datensätze auf derselben Seite wie andere Datensätze mit ähnliche Indexschlüsselwerten einzufügen. Wenn auf der betreffenden Seite kein Speicherbereich verfügbar ist, wird versucht, den Datensatz auf den umgebenden Seiten unterzubringen. Wenn dies keinen Erfolg hat, wird der oben beschriebene FSCR-Suchalgorithmus verwendet, jedoch mit dem Unterschied, dass der am schlechtesten passende Speicherplatz, nicht der erste passende gesucht wird. Durch dieses Verfahren werden normalerweise Sei-

ten ausgewählt, die mehr freien Speicherbereich enthalten. Diese Methode richtet einen neuen Clustering-Bereich für Zeilen mit diesem Schlüsselwert ein.

Wenn Sie einen Clusterindex für eine Tabelle definieren, verwenden Sie die Anweisung ALTER TABLE... PCTFREE, bevor Sie Daten in die Tabelle laden oder die Tabelle reorganisieren. Die Klausel PCTFREE definiert einen Prozentsatz an freiem Speicherplatz, der auf der Datenseite der Tabelle nach dem Laden und Reorganisieren verbleiben soll. Dadurch steigt die Wahrscheinlichkeit, dass bei der Clusterindexoperation auf der richtigen Seite freier Speicherplatz gefunden wird.

Kapitel 12. Aktualisierungsverarbeitung

Wenn ein Agent eine Seite aktualisiert, arbeitet der Datenbankmanager mit dem folgenden Protokoll, um die für die Transaktion erforderliche E/A zu minimieren und die Wiederherstellbarkeit zu gewährleisten.

1. Die zu aktualisierende Seite wird fixiert und mit einer exklusiven Sperre belegt. In den Protokollpuffer wird ein Protokollsatz geschrieben, der beschreibt, wie die Änderung wiederholt und widerrufen werden kann. Während dieser Aktion wird auch eine Protokollfolgenummer (Log Sequence Number, LSN) empfangen und in den Kopfdaten der zu aktualisierenden Seite gespeichert.
2. Die Änderung wird an der Seite vorgenommen.
3. Die Sperre und die Fixierung der Seite werden aufgehoben.

   Die Seite gilt als „benutzt", da Änderungen an der Seite noch nicht auf Platte gespeichert wurden.
4. Der Protokollpuffer wird aktualisiert.

   Sowohl die Daten im Protokollpuffer als auch die „benutzte" Datenseite werden zwangsweise auf Platte geschrieben.

Zur Erzielung einer besseren Leistung werden diese E/A-Operationen verzögert, bis ein geeigneter Zeitpunkt eintritt, zum Beispiel eine Phase mit niedriger Systemauslastung, bis sie erforderlich sind, um die Wiederherstellbarkeit sicherzustellen, oder um die Dauer der Recovery zu begrenzen. Speziell wird eine „benutzte" Seite zu folgenden Zeitpunkten gezwungenermaßen auf Platte geschrieben:

- Wenn ein anderer Agent sie auswählt.
- Wenn eine Seitenlöschfunktion die Seite aus einem der folgenden Gründe bearbeitet:
  - Ein anderer Agent wählt sie aus.
  - Der durch den Datenbankkonfigurationsparameter *chngpgs_thresh* angegebene Prozentsatz wird überschritten. Wenn dieser Wert überschritten wird, werden asynchrone Seitenlöschfunktionen aktiv und schreiben geänderte Seiten auf Platte.

    Wenn die proaktive Seitenbereinigung aktiviert ist, ist dieser Wert irrelevant und löst keine Seitenlöschfunktionen aus.
  - Der durch den Datenbankkonfigurationsparameter *softmax* angegebene Prozentsatz wird überschritten. Wenn dieser Wert überschritten ist, werden asynchrone Seitenlöschfunktionen aktiv und schreiben geänderte Seiten auf Platte.

    Wenn die proaktive Seitenbereinigung für die Datenbank aktiviert und die Anzahl von Seitenlöschfunktionen für die Datenbank richtig konfiguriert ist, sollte dieser Wert nie überschritten werden.
  - Die Anzahl sauberer Seiten auf der Vermeidungsliste fällt auf einen zu niedrigen Wert. Seitenlöschfunktionen reagieren auf diese Bedingung nur unter der Methode der proaktiven Seitenbereinigung.
  - Wenn benutzte Seiten zurzeit oder absehbar zu einer LSNGAP-Bedingung führen. Seitenlöschfunktionen reagieren auf diese Bedingung nur unter der Methode der proaktiven Seitenbereinigung.
- Wenn die Seite als Teil einer Tabelle aktualisiert wurde, für die die Klausel NOT LOGGED INITIALLY aufgerufen wird, und eine COMMIT-Anweisung abgesetzt wird. Wenn die COMMIT-Anweisung ausgeführt wird, werden alle geänderten Seiten auf Platte geschrieben, um die Wiederherstellbarkeit sicherzustellen.

Kapitel 13. Client-/Serververarbeitungsmodell

Lokale und ferne Anwendungsprozesse können mit derselben Datenbank arbeiten. Eine ferne Anwendung initiiert eine Datenbankaktion von einer Maschine aus, die sich von der Datenbankmaschine entfernt befindet. Lokale Anwendungen sind auf der Servermaschine direkt mit der Datenbank verbunden.

Wie DB2 Clientverbindungen verwaltet, ist davon abhängig, ob der Verbindungskonzentrator aktiviert ist oder nicht. Der Verbindungskonzentrator ist aktiviert, wenn der Konfigurationsparameter *max_connections* des Datenbankmanagers auf einen höheren Wert als der Konfigurationsparameter *max_coordagents* gesetzt ist.

- Wenn der Verbindungskonzentrator inaktiviert ist, wird jeder Clientanwendung eine eindeutige EDU (Engine Dispatchable Unit) zugeordnet, die als *Koordinatoragent* bezeichnet wird und die die Verarbeitung für diese Anwendung koordiniert und mit der Anwendung kommuniziert.
- Wenn der Verbindungskonzentrator aktiviert ist, kann jeder Koordinatoragent viele Clientverbindungen jeweils einzeln verwalten und ggf. die anderen Arbeitsagenten zur Erledigung dieser Arbeit koordinieren. Für Internetanwendungen mit vielen relativ kurzfristigen Verbindungen oder ähnliche Anwendungen mit zahlreichen relativ kleinen Transaktionen verbessert der Verbindungskonzentrator die Leistung, indem er wesentlich mehr Clientanwendungen die Herstellung einer Verbindung ermöglicht. Darüber hinaus verringert er die Belastung der Systemressourcen durch jede einzelne Verbindung.

Alle Kreise in dem DB2-Server der folgenden Abbildung stellen EDUs dar, die durch Betriebssystemthreads implementiert werden.

Bevor die Arbeit ausgeführt werden kann, die für die Anwendung in der Datenbank durchgeführt werden soll, muss zwischen einer Anwendung und dem Datenbankmanager ein Kommunikationsmittel eingerichtet werden.

Bei A1 in der folgenden Abbildung richtet ein lokaler Client eine Kommunikationsverbindung zunächst über eine db2ipccm-EDU für ein. Bei A2 arbeitet die db2ipccm-EDU mit einer db2agent-EDU, die zum Koordinatoragenten für die Anwendungsanforderungen vom lokalen Client wird. Der Koordinatoragent nimmt anschließend bei A3 mit der Clientanwendung Kontakt auf, um zwischen der Clientanwendung und dem Koordinator eine Kommunikation über gemeinsamen Speicher einzurichten. Die Anwendung auf dem lokalen Client wird bei A4 mit der Datenbank verbunden.

Bei B1 in der folgenden Abbildung stellt ein ferner Client eine Kommunikationsverbindung über eine db2tcpcm-EDU her. Wenn ein anderes Kommunikationsprotokoll ausgewählt wird, wird der entsprechende Kommunikationsmanager verwendet. Die db2tcpcm-EDU richtet eine TCP/IP-Kommunikation zwischen der Clientanwendung und der db2tcpcm-EDU ein. Dann arbeitet sie mit einer db2agent-EDU bei B2, die zum Koordinatoragenten für die Anwendung wird, und übergibt die Verbindung an diesen Agenten. Bei B4 nimmt der Koordinatoragent Kontakt mit der fernen Clientanwendung auf, die anschließend bei B5 mit der Datenbank verbunden wird.

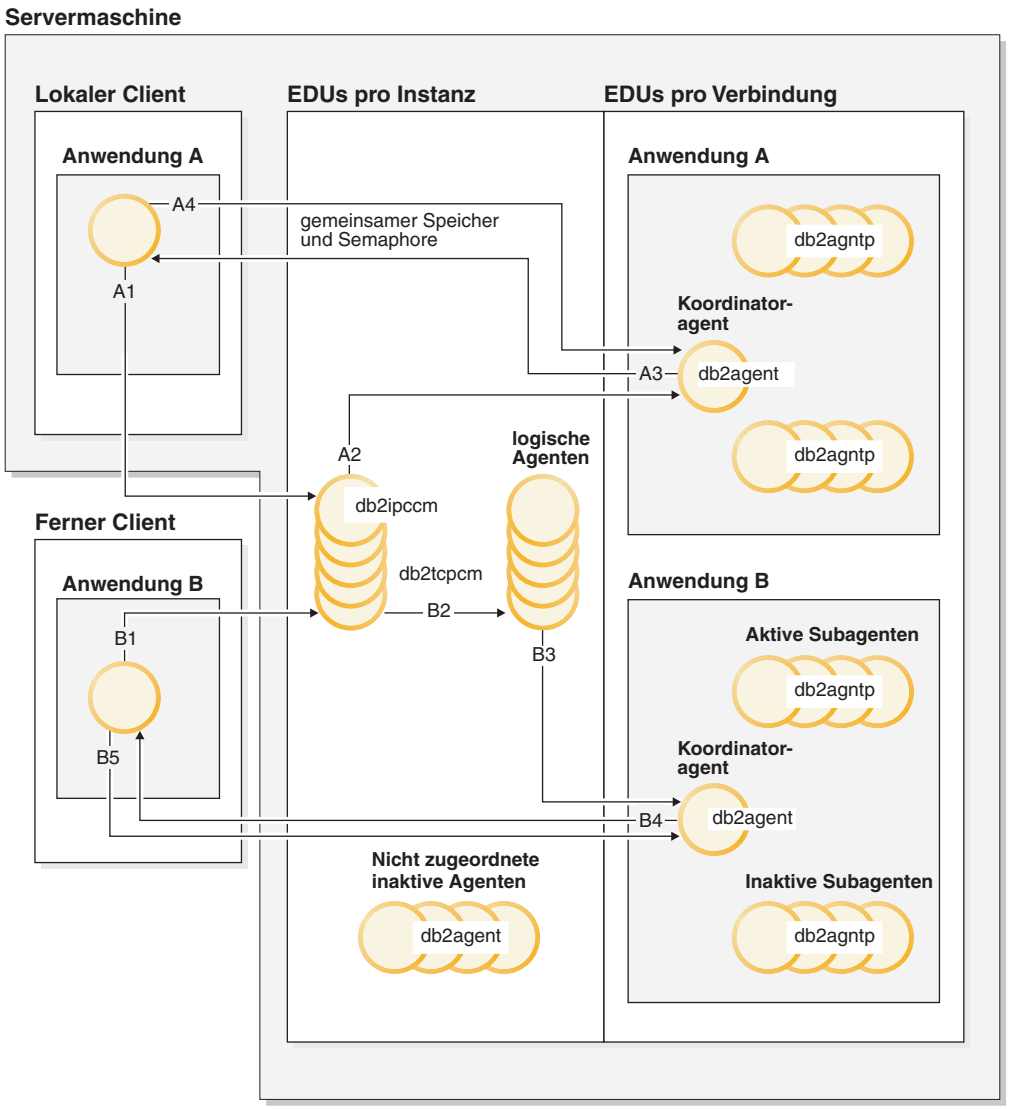


Abbildung 8. Übersicht über das Prozessmodell

Ferner sind folgende Details dieser Abbildung zu beachten:

- Arbeitsagenten führen Anwendungsanforderungen aus.
- Es gibt vier Typen von Arbeitsagenten: aktive Koordinatoragenten, aktive Subagenten, zugeordnete Subagenten und nicht zugeordnete Agenten in Bereitschaft.
- Jede Clientverbindung wird mit einem aktiven Koordinatoragenten verknüpft.
- In einer Umgebung mit partitionierten Datenbanken sowie in Umgebungen mit aktivierter partitionsinterner Parallelität verteilen die Koordinatoragenten Datenbankanforderungen an Subagenten (db2agntp). Die Subagenten führen die Anforderungen für die jeweilige Anwendung aus.
- Es gibt einen Agentenpool (db2agent), in dem inaktive Agenten zusammengefasst im Bereitschaftsmodus auf neue Arbeit warten.
- Andere EDUs verwalten Clientverbindungen, Protokolle, zweiphasige Commits, Backup- und Restore-Tasks sowie weitere Tasks.

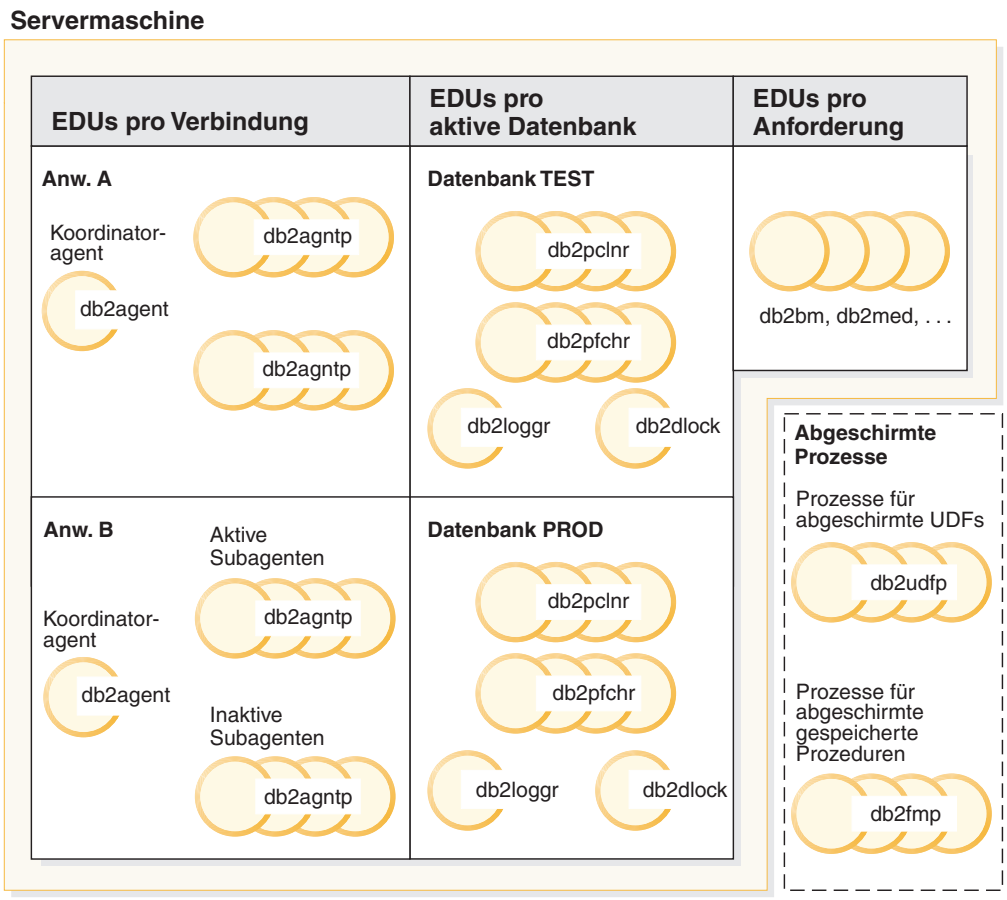


Abbildung 9. Prozessmodell, Teil 2

In dieser Abbildung werden zusätzliche EDUs (Engine Dispatchable Units, zuteilbare Einheiten der Steuerkomponente) gezeigt, die Teil der Servermaschinenumgebung sind. Alle aktiven Datenbanken verfügen über eigene gemeinsame Pools mit Vorablesefunktionen (db2pfchr) und Seitenlöschfunktionen (db2pclnr) sowie eigene Protokollfunktionen (db2loggr) und Deadlock-Detektoren (db2dlock).

Abgeschirmte benutzerdefinierte Funktionen (UDFs) und gespeicherte Prozeduren, die in der Abbildung nicht gezeigt werden, werden verwaltet, um den Aufwand zu minimieren, der mit ihrer Erstellung und Löschung verbunden ist. Der Standardwert für den Konfigurationsparameter *keepfenced* des Datenbankmanagers ist „YES", wodurch der Prozess der gespeicherten Prozedur zur erneuten Verwendung beim nächsten Aufruf einer gespeicherten Prozedur verfügbar bleibt.

Anmerkung: Nicht abgeschirmte benutzerdefinierte Funktionen (UDFs) und gespeicherte Prozeduren werden zwecks besserer Leistung direkt im Adressraum des Agenten ausgeführt. Da diese UDFs und gespeicherten Prozeduren jedoch über unbeschränkten Zugriff auf den Adressraum des Agenten verfügen, müssen sie vor ihrer Verwendung umfassend getestet werden.

Das Prozessmodell mit mehreren Datenbankpartitionen ist eine logische Erweiterung des Prozessmodells mit einer einzigen Datenbankpartition. Beide Betriebsarten werden durch eine gemeinsame Codebasis unterstützt.

In der folgenden Abbildung werden die Ähnlichkeiten und Unterschiede zwischen dem Prozessmodell mit einer einzigen Datenbankpartition, wie in den beiden vorangehenden Abbildungen gezeigt, und dem Prozessmodell mit mehreren Datenbankpartitionen angezeigt.

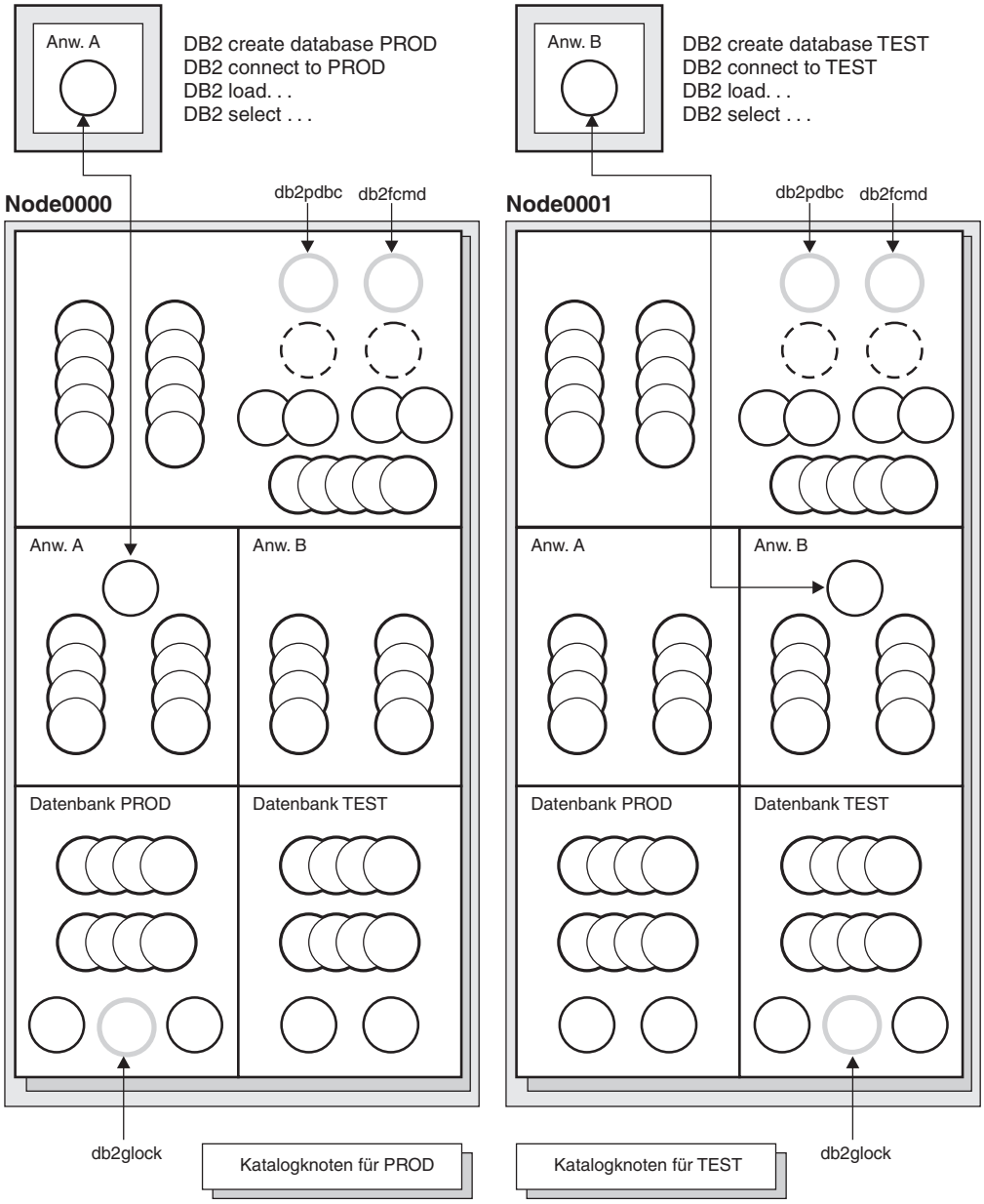


Abbildung 10. Prozessmodell und mehrere Datenbankpartitionen

Die meisten EDUs stimmen zwischen dem Prozessmodell mit einer einzigen Datenbankpartition und dem Prozessmodell mit mehreren Datenbankpartitionen überein.

In einer Umgebung mit mehreren Datenbankpartitionen (bzw. Knoten) ist eine der Datenbankpartitionen der Katalogknoten. Der Katalog speichert alle Information in Bezug auf Objekte in der Datenbank.

Wie in der vorherigen Abbildung gezeigt, wird der Katalog der PROD-Datenbank im Knoten Node0000 erstellt, weil Anwendung A die PROD-Datenbank auf diesem Knoten erstellt. Analog wird der Katalog für die TEST-Datenbank im Knoten Node0001 erstellt, weil Anwendung B die TEST-Datenbank auf diesem Knoten erstellt. Die Datenbanken sollten in verschiedenen Knoten erstellt werden, um so die zusätzlichen mit den Katalogen verbundenen Aktivitäten für jede Datenbank gleichmäßig auf die Knoten in der Systemumgebung zu verteilen.

Der Instanz sind zusätzliche EDUs (db2pdbc und db2fcmd) zugeordnet, die sich in jedem Knoten einer Datenbankumgebung mit mehreren Partitionen befinden. Diese EDUs werden für die Koordination von Anforderungen zwischen Datenbankpartitionen und zum Aktivieren des FCM (Fast Communications Manager) benötigt.

Dem Katalogknoten der Datenbank ist ebenfalls eine zusätzliche EDU (db2glock) zugeordnet. Diese EDU steuert globale Sperren für die Knoten, in denen sich die aktive Datenbank befindet.

Jede CONNECT-Verbindung von einer Anwendung wird durch eine Verbindung dargestellt, die einem Koordinatoragenten zur Verwaltung der Verbindung zugeordnet ist. Der *Koordinatoragent* ist der Agent, der mit der Anwendung kommuniziert, indem er Anforderungen empfängt und Antworten sendet. Er kann die Anforderung selbst erfüllen oder mehrere Subagenten zur Bearbeitung der Anforderung koordinieren. Die Datenbankpartition, in der sich der Koordinatoragent befindet, wird als *Koordinatorknoten* dieser Anwendung bezeichnet. Der Koordinatorknoten kann auch mit dem Befehl SET CLIENT CONNECT_NODE festgelegt werden.

Teile der Datenbankanforderungen von der Anwendung werden vom Koordinatorknoten an Subagenten der anderen Datenbankpartitionen gesendet. Alle Ergebnisse der anderen Datenbankpartitionen werden auf dem Koordinatorknoten zusammengeführt, bevor sie zurück zur Anwendung gesendet werden.

Die Datenbankpartition, in der der Befehl CREATE DATABASE ausgeführt wurde, wird „Katalogknoten" der Datenbank genannt. In dieser Datenbankpartition werden die Katalogtabellen gespeichert. Normalerweise werden alle Benutzertabellen über eine Gruppe von Knoten verteilt.

Anmerkung: Eine beliebige Anzahl von Datenbankpartitionen kann zur Ausführung auf einer einzigen Maschine konfiguriert werden. Dies wird als Konfiguration mit „mehreren logischen Partitionen" oder mit „mehreren logischen Knoten" bezeichnet. Eine solche Konfiguration ist bei großen SMP-Maschinen (SMP - Symmetric Multiprocessor) mit einem sehr großen Hauptspeicher sehr nützlich. In dieser Umgebung kann die Kommunikation zwischen Datenbankpartitionen zur Verwendung von gemeinsam genutztem Speicher und Semaphoren optimiert werden.

Teil 4. Einstiegstipps für die Leistungsoptimierung

Wenn Sie eine neue Instanz von DB2 starten, ziehen Sie die folgenden Empfehlungen für eine Basiskonfiguration in Betracht:

- Verwenden Sie den Konfigurationsadvisor in der Steuerzentrale, um Empfehlungen für sinnvolle Ausgangsstandardwerte für Ihr System zu erhalten. Die Standardwerte, die für DB2 voreingestellt sind, sollten für Ihre eindeutige Hardwareumgebung optimiert werden.

  Stellen Sie Informationen zur Hardware an Ihrem Standort zusammen, sodass Sie auf die Fragen des Assistenten antworten können. Sie können die vorgeschlagenen Einstellungen für Konfigurationsparameter sofort anwenden oder den Assistenten ein Script erstellen lassen, das auf Ihren Antworten beruht, und dieses Script später ausführen.

  Dieses Script stellt auch eine Liste der am häufigsten optimierten Parameter zur späteren Referenz bereit.
- Verwenden Sie andere Assistenten in der Steuerzentrale und in „Clientkonfiguration - Unterstützung" für leistungsrelevante Verwaltungsaufgaben. Solche Aufgaben sind in der Regel diejenigen, bei denen Sie beträchtliche Leistungsverbesserungen mit geringem Zeitaufwand und wenig Mühe erreichen.

  Andere Assistenten können Ihnen helfen, die Leistung einzelner Tabellen und des allgemeinen Datenzugriffs zu verbessern. Solche Assistenten sind: Datenbank erstellen, Tabelle erstellen, Index erstellen und Aktualisierung auf mehreren Systemen konfigurieren. Die Diagnosezentrale stellt einen Satz von Überwachungs- und Optimierungstools bereit.
- Verwenden Sie das Designadvisor-Tool in der Steuerzentrale oder über den Befehl `db2advis`, um zu ermitteln, welche Indizes, MQTs (MQT - Materialized Query Table, gespeicherten Abfragetabellen), MDC-Tabellen und Datenbankpartitionen die Abfrageleistung verbessern würden.
- Verwenden Sie den Befehl ACTIVATE DATABASE, um Datenbanken zu starten. In einer partitionierten Datenbank aktiviert dieser Befehl die Datenbank in allen Datenbankpartitionen und vermeidet die Startzeit, die zur Initialisierung der Datenbank erforderlich ist, wenn die erste Anwendung eine Verbindung herstellt.

  Anmerkung: Wenn Sie den Befehl ACTIVATE DATABASE verwenden, müssen Sie die Datenbank mit dem Befehl DEACTIVATE DATABASE herunterfahren. Die letzte Anwendung, die ihre Verbindung zur Datenbank trennt, bewirkt nicht, dass die Datenbank heruntergefahren wird.
- Lesen Sie die Übersichtstabellen, die jeden für den Datenbankmanager und für jede Datenbank verfügbaren Konfigurationsparameter auflisten und kurz beschreiben.

  Diese Übersichtstabellen enthalten eine Spalte, die angibt, ob eine Optimierung des jeweiligen Parameters einen hohen, mittleren, niedrigen oder keinen Einfluss auf die Leistung in positivem oder negativem Sinn hat. Verwenden Sie diese Tabellen, um die Parameter zu ermitteln, die Sie optimieren können, um die größten Leistungsverbesserungen zu erzielen.

Kapitel 14. Betriebsleistung

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.

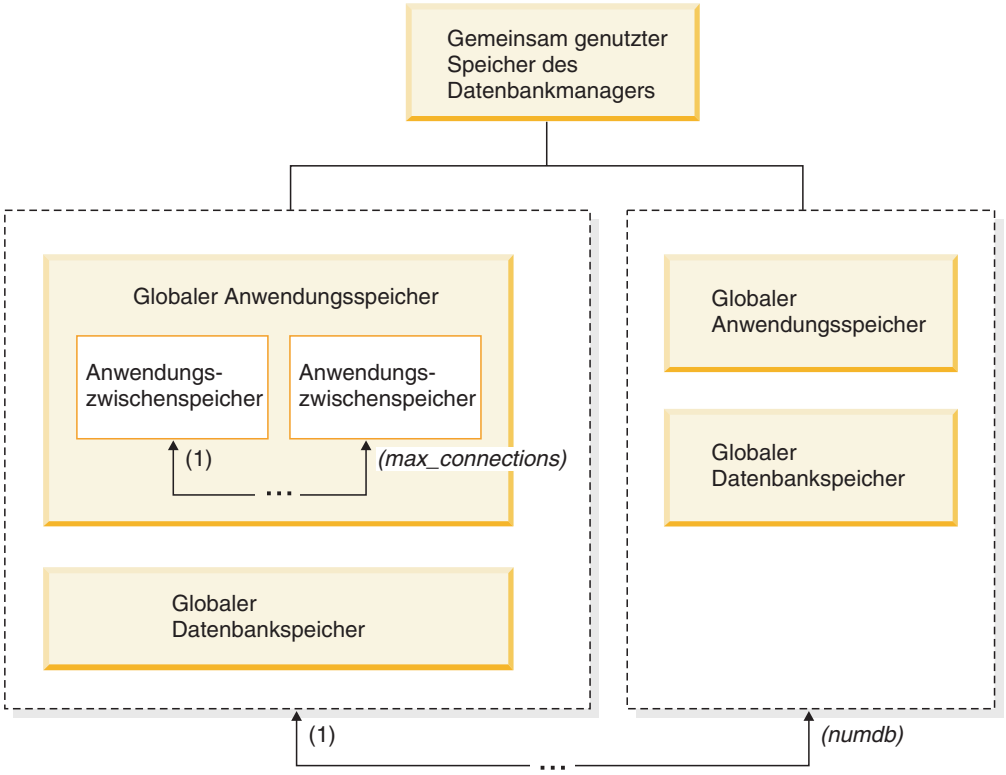


Abbildung 11. Vom Datenbankmanager verwendete Speichertypen

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**

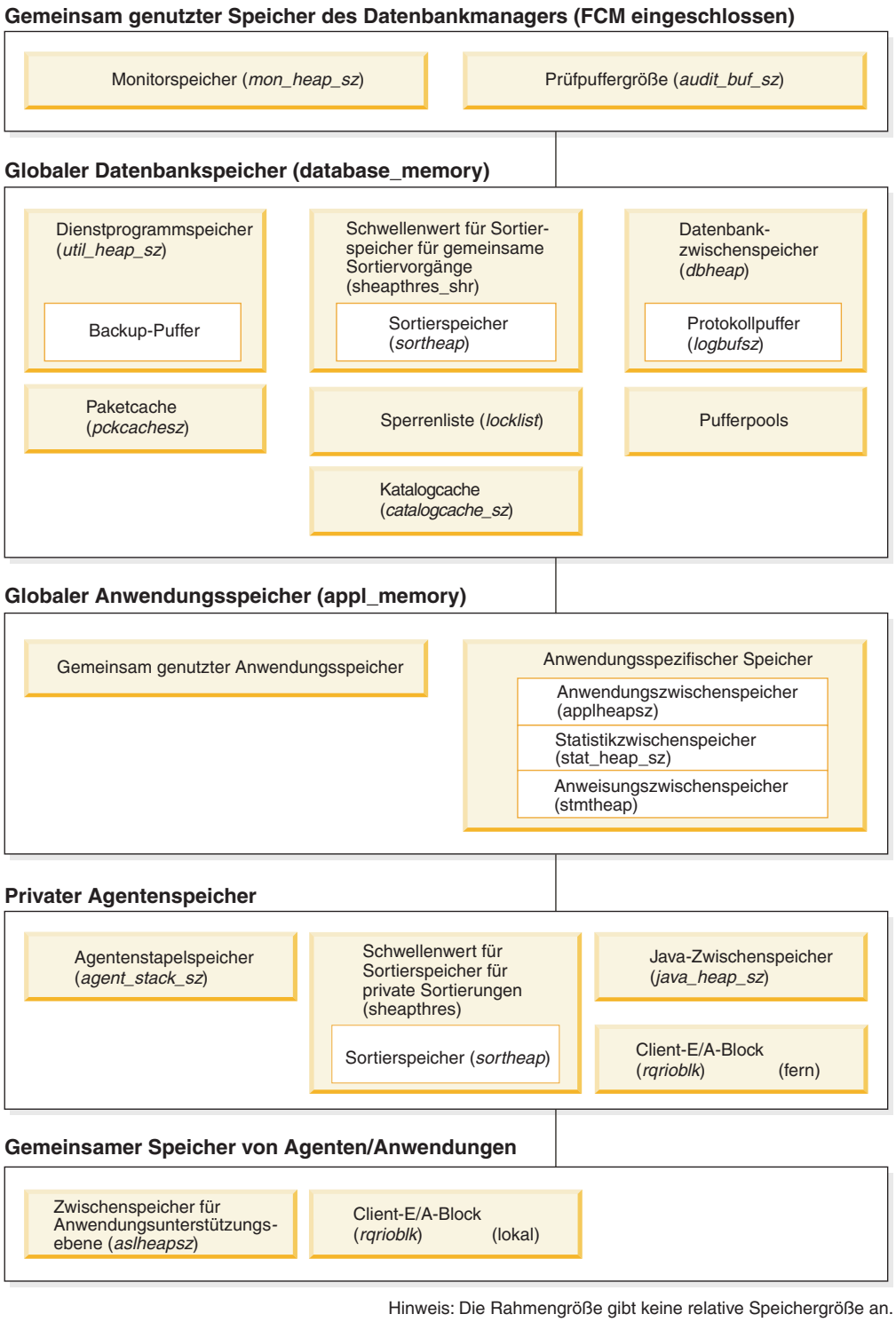
  Dieser Parameter begrenzt die Anzahl von koordinierenden Agenten des Datenbankmanagers, die gleichzeitig für alle aktiven Datenbanken in einer Instanz (in einer DPF-Instanz pro Knoten) vorhanden sein können. Zusammen mit den Parametern **maxappls** und **max_connections** begrenzt dieser Parameter die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

Gemeinsam genutzter Speicher des Datenbankmanagers

Der Speicher des Datenbankmanagers wird in vielen verschiedenen Speicherbereichen verwaltet. In der folgenden Abbildung sehen Sie, wie Datenbankmanagerspeicher zugeordnet wird. Mit den angezeigten Konfigurationsparametern können Sie die Größen der verschiedenen Speicherbereiche steuern.



Hinweis: Die Rahmengröße gibt keine relative Speichergröße an.

Abbildung 12. Verwendung von Speicher durch den Datenbankmanager

Prüfpuffer

Dieser Speicherbereich wird bei Aktivitäten der Datenbankprüffunktion verwendet. Die Größe dieses Puffers wird durch den Konfigurationsparameter **audit_buf_sz** festgelegt.

Zwischenspeicher für Monitor

Dieser Speicherbereich wird für Datenbanksystemmonitordaten verwendet. Die Größe dieses Bereichs wird durch den Konfigurationsparameter **mon_heap_sz** festgelegt.

FCM-Pufferpool (FCM = Fast Communication Manager)

Bei partitionierten Datenbanksystemen benötigt FCM (Fast Communications Manager) einen beträchtlichen Speicherbereich, insbesondere wenn der Wert des Parameters **fcm_num_buffers** hoch eingestellt ist. Der FCM-Speicherbereich wird aus dem FCM-Pufferpool zugeordnet.

Globaler Datenbankspeicher

Der globale Datenbankspeicher wird durch folgende Konfigurationsparameter beeinflusst:

- Der Parameter **database_memory** stellt einen unteren Grenzwert für die Größe des globalen Datenbankspeichers bereit.
- Die folgenden Parameter oder Faktoren steuern die Größe des globalen Datenbankspeicherbereichs:
  - Die Größe der Pufferpools.
  - Maximaler Speicher für Sperrenliste (**locklist**)
  - Datenbankzwischenspeicher (**dbheap**)
  - Zwischenspeichergröße für Dienstprogramme (**util_heap_sz**)
  - Größe des Paketcache (**pckcachesz**)
  - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge (**sheapthres_shr**)
  - Katalogcache (**catalogcache_sz**)

Globaler Anwendungsspeicher

Der globale Anwendungsspeicher kann mit dem Konfigurationsparameter **appl_memory** gesteuert werden. Die folgenden Konfigurationsparameter können zur Begrenzung des Speichers verwendet werden, der von einer beliebigen Anwendung in Anspruch genommen werden kann:

- Zwischenspeichergröße für Anwendungen (**applheapsz**)
- Größe des Anweisungszwischenspeichers (**stmtheap**)
- Größe des Statistikzwischenspeichers (**stat_heap_sz**)

Privater Agentenspeicher

- Jeder Agent benötigt einen eigenen privaten Speicherbereich. Der DB2-Server erstellt die erforderliche Anzahl von Agenten und verteilt die konfigurierten Speicherressourcen entsprechend. Sie können die maximale Anzahl von Koordinatoragenten über den Parameter **max_coordagents** steuern.
- Die maximale Größe des privaten Speicherbereichs jedes Agenten wird durch die Werte der folgenden Parameter bestimmt:
  - Größe des privaten Sortierspeichers (**sheapthres** und **sortheap**)
  - Größe des Agentenstacks (**agent_stack_sz**)

Gemeinsamer Agenten-/Anwendungsspeicher

- Die Gesamtanzahl der Segmente für gemeinsamen Agenten-/Anwendungsspeicher für lokale Clients wird durch den niedrigeren der folgenden Datenbankkonfigurationsparameter begrenzt:
  - Die Summe der Parameter **maxappls** für alle aktiven Datenbanken
  - Der Wert des Parameters **max_coordagents**

Anmerkung: In Konfigurationen, in denen die Konzentratorfunktion der Steuerkomponente aktiviert ist (**max_connections** > **max_coordagents**), wird die Anwendungsspeicherkapazität durch den Parameter **max_connections** begrenzt.

- Der gemeinsame Agenten-/Anwendungsspeicher wird außerdem durch folgende Datenbankkonfigurationsparameter beeinflusst:
  - Zwischenspeichergröße für Anwendungsunterstützungsebene (**aslheapsz**)
  - E/A-Blockgröße für Clients (**rqrioblk**)

Der FCM-Pufferpool und Speicheranforderungen

In einem partitionierten Datenbanksystem werden der gemeinsame Speicher des Datenbankmanagers und der FCM-Pufferpool wie folgt genutzt.

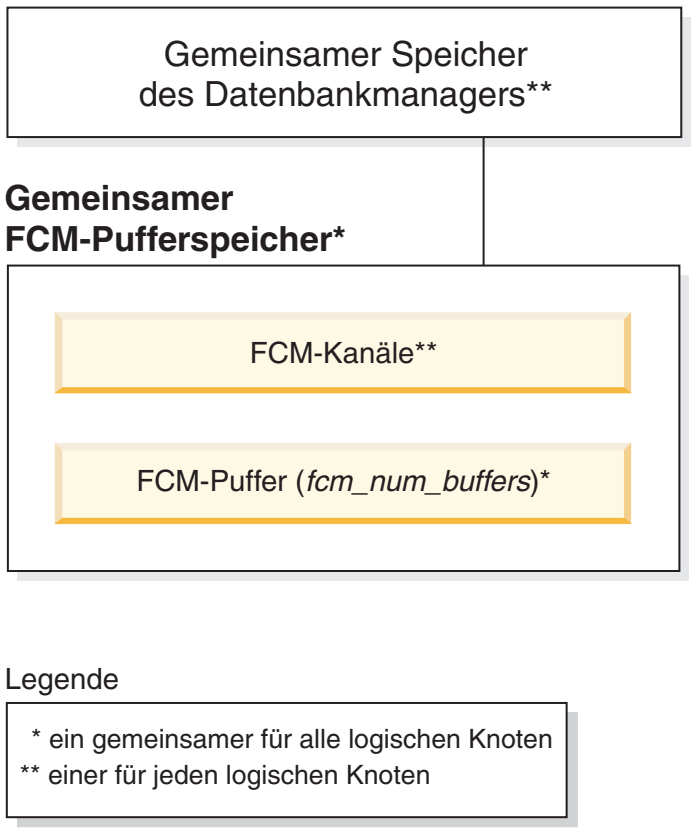


Abbildung 13. FCM-Pufferpool bei Verwendung mehrerer logischer Knoten

Die Anzahl der FCM-Puffer für jede Datenbankpartition wird durch den Konfigurationsparameter *fcm_num_buffers* gesteuert. Standardmäßig ist dieser Parameter auf den Wert AUTOMATIC gesetzt. Zur manuellen Optimierung dieses Parameters verwenden Sie die Daten aus den Systemmonitorelementen 'buff_free' (Derzeit freie FCM-Puffer) und 'buff_free_bottom' (Mindestanzahl freier FCM-Puffer).

Die Anzahl der FCM-Kanäle für jede Datenbankpartition wird durch den Konfigurationsparameter *fcm_num_channels* gesteuert. Standardmäßig ist dieser Parameter auf den Wert AUTOMATIC gesetzt. Zur manuellen Optimierung dieses Parameters verwenden Sie die Daten aus den Systemmonitorelementen 'ch_free' (momentan freie Kanäle) und 'ch_free_bottom' (Mindestanzahl freier Kanäle).

Optimieren der Parameter für die Hauptspeicherzuordnung

Die erste Regel zur Einstellung von Parametern zur Speicherzuordnung ist, sie nie auf die höchsten Werte setzen, sofern ein solcher Wert nicht sorgfältig auf seine Eignung geprüft wurde. Diese Regel gilt selbst für Systeme, die über maximale Speicherkapazitäten verfügen. Viele Parameter, die sich auf den Speicher auswirken, können zulassen, dass der Datenbankmanager leicht und schnell den gesamten auf einem Computer verfügbaren Speicher belegt. Darüber hinaus kann die Verwaltung einer großen Speichermenge wesentlich mehr Zusatzaufwand auf Seiten des Datenbankmanagers verursachen, sodass der Systemaufwand steigt.

Einige UNIX-Betriebssysteme ordnen Auslagerungsspeicher zu, wenn ein Prozess Speicher zuordnet, und nicht, wenn der Prozess in den Auslagerungsspeicher ausgelagert wird. Stellen Sie für solche Systeme sicher, dass Sie Paging-Bereich in der Größe des gesamten gemeinsamen Speicherbereichs bereitstellen.

Bei der Mehrzahl der Konfigurationsparameter wird der entsprechende Speicher erst festgeschrieben, wenn er erforderlich ist; die Parametereinstellungen legen die maximale Größe eines bestimmten Zwischenspeicherbereichs fest. In den folgenden Fällen wird jedoch die volle Größe des Speichers zugeordnet, die durch den Parameter angegeben wird:

- Maximaler Speicher für Sperrenliste (*locklist*)
- Zwischenspeichergröße für Anwendungsunterstützungsebene (*aslheapsz*)
- Anzahl der FCM-Puffer (*fcm_num_buffers*)
- Anzahl der FCM-Kanäle (*fcm_num_channels*)
- Pufferpools

Anmerkungen:

- Vergleichstests liefern die besten Informationen zur Einstellung geeigneter Werte für Speicherparameter. Bei der Durchführung von Vergleichstests werden repräsentative SQL-Anweisungen und Extremfall-SQL-Anweisungen für den Server ausgeführt und die Werte der Parameter modifiziert, bis der Punkt gefunden wird, an dem die Leistung wieder sinkt. Wenn die Leistung gegen einen Parameterwert in einem Diagramm aufgezeichnet würde, zeigt die Stelle, an dem die Kurve nicht weiter steigt bzw. wieder zu sinken beginnt, den Punkt an, an dem eine weitere Speicherzuordnung keinen weiteren Vorteil für die Anwendung bringt und daher nur zu einer Verschwendung von Speicher führen würde.
- Die oberen Grenzen der Speicherzuordnung einiger Parameter können über den Speicherkapazitäten der vorhandenen Hardware und des Betriebssystems liegen. Diese Grenzen tragen steigenden Anforderungen in der Zukunft Rechnung.
- Gültige Wertebereiche der Parameter finden Sie in den detaillierten Informationen zu den einzelnen Parametern.

Speicher mit automatischer Leistungsoptimierung - Übersicht

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 dynamisch unter mehreren Speicherkonsumenten, zu denen der Sortierspeicher, der Paketcache, der Sperrenlistenspeicher und Pufferpools zählen.

In der folgenden Tabelle sind die Abschnitte zum Speicher mit automatischer Leistungsoptimierung nach Kategorie aufgelistet:

Tabelle 1. Übersicht über die Informationen zum Speicher mit automatischer Leistungsoptimierung

Kategorie	Zugehörige Themen
Allgemeine Informationen und Einschränkungen	• „Speicher mit automatischer Leistungsoptimierung" • „Betriebsmerkmale und Einschränkungen von Speicher mit automatischer Leistungsoptimierung" auf Seite 64 • self_tuning_mem - Speicher mit automatischer Leistungsoptimierung (Konfigurationsparameter)
Aktivierung und Inaktivierung	• „Aktivieren des Speichers mit automatischer Leistungsoptimierung" auf Seite 61 • „Inaktivieren des Speichers mit automatischer Leistungsoptimierung" auf Seite 62
Überwachung	• „Ermitteln der Speicherkonsumenten mit aktivierter automatischer Leistungsoptimierung" auf Seite 63
DPF-Hinweise	• „Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken" auf Seite 65 • „Verwenden von Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken" auf Seite 67
Automatisch optimierbare Konfigurationsparameter	• „database_memory - Größe des gemeinsam genutzten Datenbankspeichers" in *Konfigurationsparameter - Referenzinformationen* • „locklist - Maximaler Speicher für Sperrenliste" in *Konfigurationsparameter - Referenzinformationen* • „maxlocks - Maximale Anzahl von Sperren vor Eskalation" in *Konfigurationsparameter - Referenzinformationen* • „pckcachesz - Größe des Paketcache" in *Konfigurationsparameter - Referenzinformationen* • „sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge" in *Konfigurationsparameter - Referenzinformationen* • „sortheap - Sortierspeichergröße" in *Konfigurationsparameter - Referenzinformationen*

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**.

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 der DPF-Umgebung 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 übertragen, sodass sie auf dem neuen primären Server erfolgt.

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.

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.

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 Optimierungs-

partition 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.

Pufferpoolverwaltung

Ein Pufferpool stellt einen Arbeitsspeicher und einen Cache für Datenbankseiten bereit. Der Pufferpool verbessert die Leistung des Datenbanksystems, indem er die Möglichkeit schafft, auf Daten im Hauptspeicher anstatt auf der Platte zuzugreifen. Da der Zugriff auf den Hauptspeicher wesentlich schneller als der Plattenzugriff arbeitet, ist die Leistung der Datenbank umso besser, je seltener der Datenbankmanager Daten von der Platte lesen bzw. auf die Platte schreiben muss. Aufgrund der Tatsache, dass die meisten Bearbeitungsschritte für Seitendaten in Pufferpools stattfinden, ist die Konfiguration von Pufferpools der wichtigste Einzelbereich bei der Optimierung.

Wenn eine Anwendung auf eine Zeile einer Tabelle zugreift, sucht der Datenbankmanager die Seite, die diese Zeile enthält, im Pufferpool. Wenn die Seite im Pufferpool nicht vorhanden ist, liest der Datenbankmanager die Seite von der Platte in den Pufferpool ein. Wenn sich die Seite im Pufferpool befindet, können die Daten vom Datenbankmanager zur Verarbeitung der Abfrage verwendet werden.

Der Speicher für den Pufferpool wird zugeordnet, wenn eine Datenbank aktiviert wird. Die erste Anwendung, die eine Verbindung herstellt, kann eine implizite Aktivierung der Datenbank bewirken. Pufferpools können auch erstellt, gelöscht

oder in der Größe geändert werden, während der Datenbankmanager aktiv ist. Mit der Anweisung ALTER BUFFERPOOL kann der Pufferpool vergrößert werden. Standardmäßig wird das Schlüsselwort IMMEDIATE für die Anweisung ALTER BUFFERPOOL angegeben, und der Speicher wird zugeordnet, sobald Sie den Befehl eingeben, sofern der Speicher verfügbar ist. Wenn der Speicher nicht verfügbar ist, wird die Anweisung verzögert (DEFERRED) ausgeführt, und der Speicher wird zugeordnet, wenn die Datenbank erneut aktiviert wird. Wenn Sie den Pufferpool verkleinern, wird die Zuordnung von Speicher aufgehoben, wenn die Transaktion festgeschrieben (Commit) wird. Der Pufferpoolspeicher wird freigegeben, wenn die Datenbank inaktiviert wird. Je nachdem, wie die Datenbank aktiviert wurde, kann sie implizit inaktiviert werden, wenn die letzte Anwendung die Verbindung trennt.

Anmerkung: Zur Verringerung der Notwendigkeit, den Wert des Datenbankkonfigurationsparameters *dbheap* zu erhöhen, wenn der Pufferpool vergrößert wird, wird beinahe der gesamte Pufferpoolspeicher dem gemeinsamen Datenbankspeicher entnommen und automatisch in der Größe angepasst.

Um sicherzustellen, dass in allen Situationen ein geeigneter Pufferpool verfügbar ist, erstellt DB2 kleine Systempufferpools, d. h. je einen mit der Seitengröße 4 KB, 8 KB, 16 KB und 32 KB. Die Größe der Pufferpools beträgt 16 Seiten. Diese Pufferpools sind für den Benutzer verdeckt. Sie treten weder in den Systemkatalogen noch in den Pufferpoolsystemdateien auf. Diese Pufferpools können von Ihnen auch nicht direkt verwendet oder geändert werden, sondern werden von DB2 in den folgenden Situationen genutzt:

- Wenn der angegebene Pufferpool nicht gestartet wird, weil er mit dem Schlüsselwort DEFERRED erstellt wurde, oder wenn ein Pufferpool der erforderlichen Seitengröße nicht aktiv ist, weil nicht genügend Speicher zu seiner Erstellung verfügbar ist.

  Eine Nachricht wird in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben. Falls erforderlich, werden Tabellenbereiche einem Systempufferpool zugeordnet. Die Leistung könnte sich beträchtlich verringern.
- Wenn die normalen Pufferpools bei der Herstellung einer Datenbankverbindung nicht aktiviert werden können.

  Dieses Problem hat wahrscheinlich eine ernsthafte Ursache, wie zum Beispiel, dass kein Speicher mehr verfügbar ist. Obwohl DB2 aufgrund der Systempufferpools den vollen Funktionsumfang bewahrt, verschlechtert sich die Leistung erheblich. Sie sollten dieses Problem unverzüglich untersuchen. Wenn dieser Fall eintritt, empfangen Sie eine Warnung und eine Nachricht wird in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben.

Wenn Sie einen Pufferpool erstellen, ist seine Seitengröße diejenige, die bei der Erstellung der Datenbank angegeben wurde, sofern Sie nicht explizit eine andere Seitengröße angeben. Da Seiten in einen Pufferpool nur eingelesen werden können, wenn die Seitengröße des Tabellenbereichs mit der Seitengröße des Pufferpools übereinstimmt, sollte die Seitengröße Ihrer Tabellenbereiche auch die Seitengröße sein, die Sie für Pufferpools angeben. Nach der Erstellung eines Pufferpools können Sie seine Seitengröße nicht mehr ändern. Sie müssen einen neuen Pufferpool mit einer anderen Seitengröße erstellen.

Mit dem Speichertracker, der über den Befehl 'db2mtrk' aufgerufen wird, können Sie die Menge an Datenbankspeicher anzeigen, die den Pufferpools zugeordnet ist. In der folgenden Beispielausgabe von 'db2mtrk' wird ein vom Benutzer erstellter

Pufferpool, der durch die Nummer ″1″ in der runden Klammer angegeben wird, sowie vier Systempufferpools, die durch die Seitengröße in der runden Klammer angegeben werden, erstellt:

```
> db2mtrk -d
Speicher wird überwacht am 2005/10/24 um 12:47:26

Speicher für Datenbank: XMLDB

   utilh      pckcacheh catcacheh bph (1)   bph (S32K)  bph (S16K)  bph (S8K)
   64,0K      576,0K    64,0K     4,2M      576,0K      320,0K      192,0K

   bph (S4K)  shsorth   lockh     dbh       other
   128,0K     0         640,0K    4,2M      192,0K
```

Pufferpoolverwaltung von Datenseiten

Seiten im Puffer können entweder *im Gebrauch* oder nicht, und sie können *benutzt* (geändert) oder *sauber* (ungeändert) sein:

- Seiten, die im Gebrauch sind, werden momentan gelesen oder aktualisiert. Wenn eine Seite aktualisiert wird, kann nur der aktualisierende Benutzer auf sie zugreifen. Wird die Seite jedoch nicht aktualisiert, kann sie von mehreren Benutzern gleichzeitig gelesen werden.
- Benutzte Seiten enthalten Daten, die geändert, jedoch noch nicht auf die Platte geschrieben wurden.

Seiten verbleiben im Pufferpool, bis die Datenbank gestoppt wird, bis der von einer Seite belegte Speicherbereich im Pufferpool für eine andere Seite benötigt wird, oder bis die Seite explizit aus dem Pufferpool entfernt wird, zum Beispiel wenn ein Objekt gelöscht wird. Die folgenden Bedingungen bestimmen, welche Seite entfernt wird, um ein andere Seite einzulesen:

- Der letzte Verweis auf die Seite
- Die Wahrscheinlichkeit, mit der erneut auf die Seite zugegriffen wird
- Der Typ der Daten auf der Seite
- Der Änderungsstatus der Seite, d. h. ob sie im Speicher geändert, jedoch noch nicht auf Platte geschrieben wurde (geänderte Seiten werden immer auf der Festplatte gespeichert, bevor sie im Pufferpool überschrieben werden)

Geänderte Seiten, die auf die Festplatte geschrieben werden, werden nicht automatisch aus dem Pufferpool entfernt, sofern der Speicherplatz nicht benötigt wird.

Agenten für Seitenlöschfunktionen

In einem entsprechend optimierten System schreiben meist Agenten für Seitenlöschfunktionen benutzte Seiten auf die Platte. Agenten für Seitenlöschfunktionen führen E/A-Operationen als Hintergrundprozesse aus und ermöglichen Anwendungen eine schnellere Ausführung, weil ihre Agenten die tatsächlichen Transaktionen ausführen können. Agenten für Seitenlöschfunktionen können auch als *asynchrone Seitenlöschfunktionen* oder *asynchrone Pufferschreibfunktionen* bezeichnet werden, da sie nicht mit der Arbeit anderer Agenten koordiniert werden und nur bei Bedarf aktiv werden.

Zur Verbesserung der Leistung für aktualisierungsintensive Auslastungen kann es nützlich sein, die proaktive Seitenbereinigung zu aktivieren. Die Leistung kann sich verbessern, da sich Seitenlöschfunktionen proaktiver bei der Auswahl der benutzten (geänderten) Seiten verhalten, die zu einem gegebenen Zeitpunkt ausgelesen und auf die Platte geschrieben werden. Dies gilt insbesondere, wenn Momentauf-

nahmen offenbaren, dass eine erhebliche Anzahl synchroner Schreiboperationen für Daten- oder Indexseiten im Verhältnis zur Anzahl asynchroner Schreiboperationen für Daten- oder Indexseiten stattfindet.

Weitere Informationen finden Sie in „Proaktive Seitenbereinigung” auf Seite 76.

Illustration der Datenseitenverwaltung in Pufferpools

Die folgende Abbildung veranschaulicht, wie die Arbeit zur Verwaltung des Pufferpools zwischen den Agenten für Seitenlöschfunktionen und den Datenbankagenten aufgeteilt werden kann, im Vergleich zu der Situation, in der die Datenbankagenten die gesamte Ein-/Ausgabe selbst durchführen.

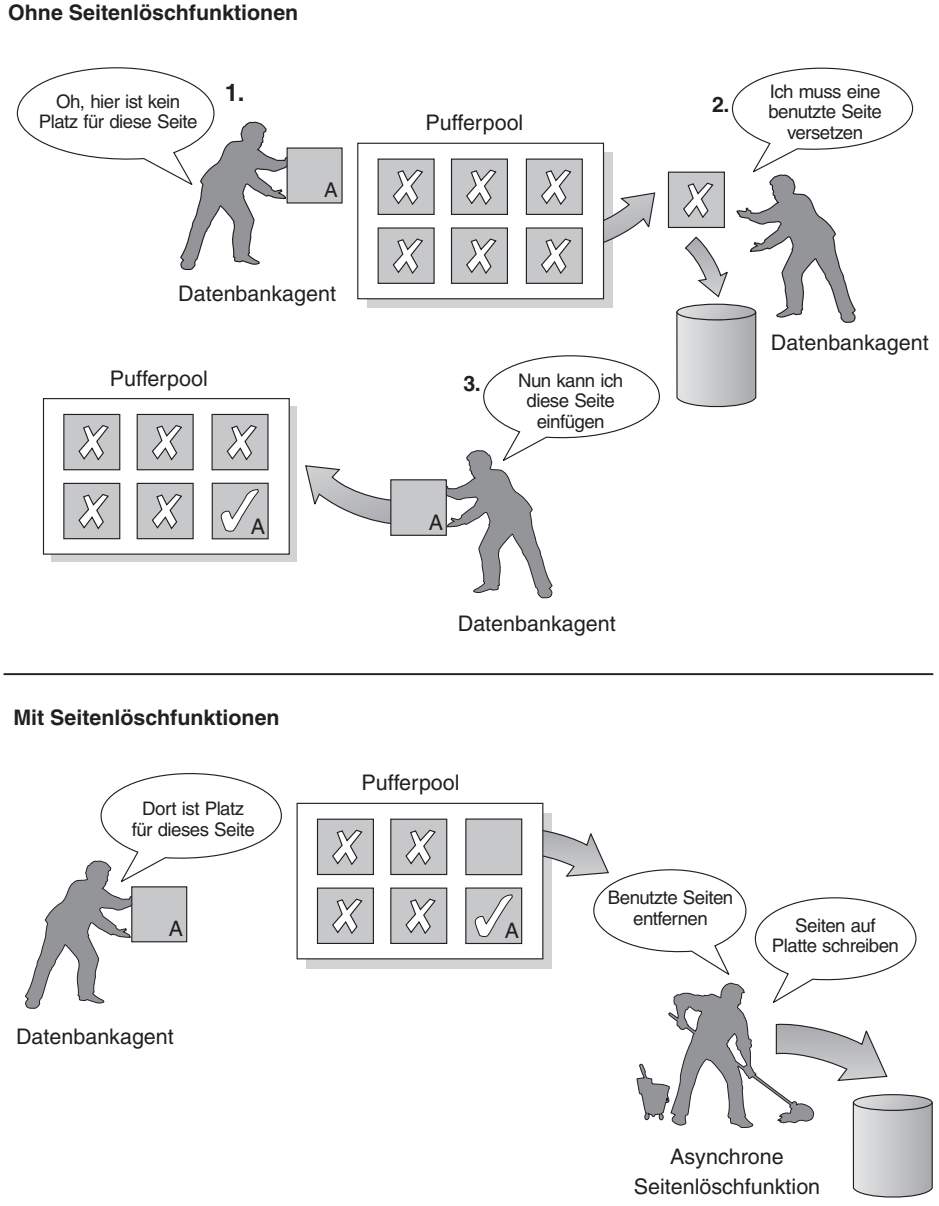


Abbildung 14. Asynchrone Seitenlöschfunktion. Geänderte Seiten werden auf Platte geschrieben.

Seitenbereinigung und schnelle Recovery

Wenn mehr Seiten auf die Festplatte geschrieben wurden, ist die Recovery der Datenbank nach einem Systemabsturz schneller, weil der Datenbankmanager größere Teile des Pufferpools von der Festplatte rekonstruieren kann, anstatt Transaktionen aus den Datenbankprotokolldateien erneut durchzuführen.

Die Größe des Protokolls, das während der Recovery gelesen werden muss, errechnet sich aus der Differenz der Positionen der folgenden Datensätze im Protokoll:

- Der zuletzt geschriebene Protokollsatz
- Der Protokollsatz, der die älteste Änderung an Daten im Pufferpool beschreibt

Die Seitenlöschfunktionen schreiben benutze Seiten auf die Platte, sodass die Größe des Protokolls, das bei einer Recovery wiederholt werden müsste, den nach folgender Formel berechneten Wert nie übersteigt:

```
logfilsiz * softmax / 100 (in 4-KB-Seiten)
```

Dabei gilt Folgendes:

- *logfilsiz* ist die Größe der Protokolldateien.
- *softmax* ist der Prozentsatz der Protokolldateien, die nach einem Datenbankabsturz wiederherzustellen sind. Wenn der Parameter *softmax* zum Beispiel den Wert 250 hat, enthalten 2,5 Protokolldateien die Änderungen, die wiederhergestellt werden müssen, wenn ein Systemabsturz auftritt.

Ermitteln Sie zur Minimierung der Zeit, die bei einer Recovery auf das Lesen der Protokolle verwendet wird, mithilfe des Datenbanksystemmonitors die Häufigkeit, mit der Seitenlöschfunktionen aktiv werden. Das Monitorelement *pool_lsn_gap_clns (für Pufferpoolprotokollbereich ausgelöste Seitenlöschfunktionen)* des Systemmonitors stellt diese Information bereit, wenn Sie nicht die proaktive Seitenbereinigung für Ihre Datenbank aktiviert haben. Wenn Sie diese alternative Seitenbereinigungsfunktion aktiviert haben, sollte diese Situation nicht eintreten und das Monitorelement *pool_lsn_gap_clns* zeigt immer den Wert 0.

Mithilfe des Monitorelements *log_held_by_dirty_pages* können Sie bestimmen, ob die Seitenlöschfunktionen nicht ausreichend Seiten bereinigen, um die vom Benutzer festgelegten Bedingungen für die Recovery zu erfüllen. Wenn das Monitorelement *log_held_by_dirty_pages* beständig einen erheblich größeren Wert als *logfilsiz * softmax* zeigt, sind entweder mehr Seitenlöschfunktionen erforderlich oder der Wert des Parameters *softmax* muss angepasst werden.

Verwaltung mehrerer Datenbankpufferpools

Obwohl für jede Datenbank mindestens ein Pufferpool erforderlich ist, können Sie für eine einzige Datenbank, die Tabellenbereiche mit mehr als einer Seitengröße hat, mehrere Pufferpools erstellen, die alle eine andere Größe oder Seitengröße aufweisen. In Abhängigkeit von der Plattform verfügt jeder Pufferpool über eine Mindestgröße. Mit dem Befehl ALTER BUFFERPOOL können Sie die Größe eines Pufferpools ändern.

Eine neue Datenbank verfügt über einen Standardpufferpool mit dem Namen IBMDEFAULTBP mit einer allgemeinen Größe, die von der Plattform festgelegt wird, und der Standardseitengröße, die auf der bei der Datenbankerstellung angegebenen Seitengröße basiert. Die Standardseitengröße wird als Informationsdatenbankkonfigurationsparameter *pagesize* gespeichert. Wenn Sie einen Tabellenbereich mit der Standardseitengröße erstellen und ihn keinem bestimmten Pufferpool zuord-

nen, wird der Tabellenbereich dem Standardpufferpool zugeordnet. Sie können die Größe des Standardpufferpools und seine Attribute ändern, jedoch können Sie ihn nicht löschen.

Seitengrößen für Pufferpools

Nach der Erstellung oder Migration einer Datenbank können Sie weitere Pufferpools erstellen. Sie können die Datenbank mit einer Standardseitengröße von 8 KB erstellen. Der Standardpufferpool wird dann mit der Standardseitengröße erstellt (in diesem Fall beträgt die Seitengröße 8 KB). Alternativ dazu können Sie einen Pufferpool mit einer Seitengröße von 8 KB sowie einen oder mehrere Tabellenbereiche mit derselben Seitengröße erstellen. Bei dieser Methode ist es nicht erforderlich, bei der Erstellung der Datenbank die Standardseitengröße von 4 KB zu ändern. Sie können einen Tabellenbereich keinem Pufferpool zuordnen, der eine andere Seitengröße verwendet.

Anmerkung: Wenn Sie einen Tabellenbereich mit einer Seitengröße über 4 KB (z. B. 8 KB, 16 KB oder 32 KB) erstellen, müssen Sie ihn einem Pufferpool zuordnen, der dieselbe Seitengröße verwendet. Wenn dieser Pufferpool momentan nicht aktiv ist, versucht DB2, den Tabellenbereich vorübergehend einem anderen aktiven Pufferpool mit der gleichen Seitengröße zuzuordnen, falls einer vorhanden ist, bzw. einem der Pufferpools, die DB2 standardmäßig erstellt, wenn der erste Client eine Verbindung zur Datenbank herstellt. Ist der ursprünglich angegebene Pufferpool beim nächsten Aktivieren der Datenbank aktiv, ordnet DB2 den Tabellenbereich zu diesem Pufferpool zu.

Wenn Sie einen Pufferpool mit der Anweisung CREATE BUFFERPOOL erstellen, können Sie eine bestimmte Pufferpoolgröße angeben. Wenn Sie keine Größe angeben, wird die Pufferpoolgröße auf AUTOMATIC gesetzt und von DB2 verwaltet. Wenn Sie die Pufferpoolgröße später ändern wollen, verwenden Sie dazu die Anweisung ALTER BUFFERPOOL.

In einer partitionierten Datenbankumgebung besitzt jeder Pufferpool für eine Datenbank in allen Datenbankpartitionen die gleiche Standarddefinition, sofern dies nicht in der Anweisung CREATE BUFFERPOOL anders angegeben oder die Größe des Pufferpools für eine bestimmte Datenbankpartition mit der Anweisung ALTER BUFFERPOOL geändert wurde.

Vorteile großer Pufferpools

Große Pufferpools bieten die folgenden Vorteile:

- Sie ermöglichen, dass häufig angeforderte Datenseiten im Pufferpool behalten werden können. Dadurch kann schneller auf sie zugegriffen werden. Durch weniger E/A-Operationen können E/A-Konkurrenzsituationen besser vermieden werden, sodass die Antwortzeiten verbessert und die für E/A-Operationen benötigten Prozessorressourcen reduziert werden.
- Sie bieten die Möglichkeit, höhere Transaktionsgeschwindigkeiten mit derselben Antwortzeit zu erzielen.
- Sie vermeiden Konkurrenzsituationen bei der Ein-/Ausgabe für häufig verwendete Plattenspeichereinheiten zum Beispiel für Katalogtabellen, häufig verwendete Benutzertabellen und Indizes. Auch Sortierungen durch Abfragen profitieren von verminderten E/A-Konkurrenzsituationen auf Plattenspeichereinheiten, die die temporären Tabellenbereiche enthalten.

Vorteile vieler Pufferpools

Wenn eine der folgenden Bedingungen auf Ihr System zutrifft, sollten Sie nur einen einzigen Pufferpool verwenden:
- Der gesamte Pufferspeicherbereich beträgt weniger als 10.000 4-KB-Seiten.
- Es stehen keine Fachleute mit Anwendungskenntnissen für die spezialisierte Optimierung zur Verfügung.
- Sie arbeiten auf einem Testsystem.

In allen anderen Fällen sollten Sie die Verwendung mehrerer Pufferpools aus folgenden Gründen in Betracht ziehen:
- Tabellenbereiche für temporäre Tabellen können einem getrennten Pufferpool zugeordnet werden, um eine bessere Leistung für Abfragen zu erzielen, die temporären Speicherbereich benötigen, insbesondere Abfragen mit zahlreichen Sortierungen.
- Wenn auf Daten wiederholt und schnell von vielen kurzen Anwendungen mit Aktualisierungstransaktionen zugegriffen werden muss, ziehen Sie in Betracht, den Tabellenbereich, der die Daten enthält, einem getrennten Pufferpool zuzuordnen. Wenn dieser Pufferpool eine geeignete Größe hat, ist die Wahrscheinlichkeit höher, dass die Seiten im Pufferpool gefunden werden. Dies trägt zur Verkürzung der Antwortzeiten und zur Reduzierung der Transaktionskosten bei.
- Sie können Daten in getrennten Pufferpools isolieren, um eine bevorzugte Verarbeitung bestimmter Anwendungen, Daten und Indizes zu erreichen. Zum Beispiel könnte es sinnvoll sein, Tabellen und Indizes, die häufig aktualisiert werden, in einen Pufferpool einzulesen, der von anderen Tabellen und Indizes, die zwar häufig abgefragt, aber nicht häufig aktualisiert werden, getrennt ist. Durch diese Änderung werden die Auswirkungen verringert, die häufige Aktualisierungen an der ersten Gruppe von Tabellen auf häufige Abfragen auf die zweite Gruppe von Tabellen haben.
- Sie können kleinere Pufferpools für die Daten verwenden, auf die Anwendungen zugreifen, die sehr selten ausgeführt werden, insbesondere wenn eine solche Anwendung einen sehr wahlfreien Zugriff auf eine sehr umfangreiche Tabelle benötigt. In diesem Fall brauchen die Daten nicht länger als für eine einzige Abfrage im Pufferpool behalten zu werden. Es ist günstiger, einen kleinen Pufferpool für diese Daten zu verwenden und den übrigen Speicher für andere Zwecke freizugeben, wie zum Beispiel für andere Pufferpools.
- Nach der Trennung der verschiedenen Aktivitäten und Daten in verschiedene Pufferpools können gute und relativ unaufwendige Daten zur Leistungsdiagnose aus den Statistiken und aus Tracefunktionen für Benutzeraktivitäten gewonnen werden.

Zuordnung von Pufferpoolspeicher beim Start

Wenn Sie einen Pufferpool mit dem Befehl CREATE BUFFERPOOL erstellen oder Pufferpools mit der Anweisung ALTER BUFFERPOOL ändern, muss der gesamte Speicherbereich, der von allen Pufferpools benötigt wird, dem Datenbankmanager zur Verfügung stehen, damit alle Pufferpools beim Starten der Datenbank zugeordnet werden können. Wenn Sie Pufferpools erstellen oder modifizieren, während der Datenbankmanager online ist, sollte zusätzlicher Speicherplatz im globalen Datenbankspeicher verfügbar sein. Wenn Sie das Schlüsselwort DEFERRED bei der Erstellung eines neuen Pufferpools bzw. bei der Vergrößerung eines vorhandenen Pufferpools angeben und der erforderliche Speicherplatz nicht verfügbar ist, führt der Datenbankmanager die Änderung bei der nächsten Aktivierung der Datenbank durch.

Wenn dieser Speicher beim Start einer Datenbank nicht verfügbar ist, startet der Datenbankmanager nur mit Systempufferpools (einer pro Seitengröße) mit einer Minimalgröße von 16 Seiten. Außerdem wird die Warnung SQL1478W (SQLSTATE01626) zurückgegeben. Die Datenbank setzt den Betrieb in diesem Status fort, bis ihre Konfiguration geändert wird und die Datenbank vollständig neu gestartet werden kann. Die Leistung ist möglicherweise beeinträchtigt. Der Datenbankmanager wird mit den minimal dimensionierten Werten nur deshalb gestartet, um Ihnen die Möglichkeit zu geben, eine Verbindung zur Datenbank herzustellen und die Pufferpoolgrößen anders zu konfigurieren bzw. andere wichtige Maßnahmen durchzuführen. Starten Sie die Datenbank neu, sobald Sie diese Maßnahmen durchgeführt haben. Lassen Sie die Datenbank nicht über längere Zeit in einem solchen Status arbeiten.

Damit die Datenbank nicht nur mit Systempufferpools gestartet wird, können Sie die Registrierdatenbankvariable DB2_OVERRIDE_BPF verwenden, um den erforderlichen Speicher so anzupassen, dass er in die verfügbaren Speicherkapazitäten passt.

Proaktive Seitenbereinigung

Seit Version 8.1.4 steht eine alternative Methode zur Konfiguration der Seitenbereinigung in Ihrem System zur Verfügung. Diese alternative Methode unterscheidet sich insofern von der Standardfunktionsweise, als dass die Seitenlöschfunktionen sich proaktiver bei der Auswahl der benutzten (geänderten) Seiten verhalten, die zu einem gegebenen Zeitpunkt ausgelesen und auf die Platte geschrieben werden. Diese neue Methode der Seitenbereinigung unterscheidet sich von der Standardseitenbereinigung in zwei wesentlichen Punkten:

1. Die Seitenlöschfunktionen werden nicht durch den Konfigurationsparameter 'chngpgs_thresh' beeinflusst.

   Bei dieser alternativen Methode der Seitenbereinigung reagieren Seitenlöschfunktionen nicht mehr auf den Wert des Konfigurationsparameters 'chngpgs_thresh'. Anstatt zu versuchen, einen Prozentsatz des Pufferpools sauber zu halten, stellt die alternative Methode der Seitenbereinigung einen Mechanismus bereit, durch den Agenten über die Position gut geeigneter Seiten, die gerade auf die Platte geschrieben wurden, informiert werden, sodass Agenten den Pufferpool nicht durchsuchen müssen, um geeignete Seiten zu finden. Wenn die Anzahl gut geeigneter Seiten unter einen akzeptablen Wert fällt, werden die Seitenlöschfunktionen ausgelöst, die daraufhin den gesamten Pufferpool durchsuchen und potenziell geeignete Seiten auf die Platte schreiben und die Agenten über die Positionen dieser Seiten informieren.
2. Seitenfunktionen reagieren nicht mehr auf Trigger bei LSN-Abstimmungsverlusten (LSN - Protokollfolgenummer), die von der Protokollfunktion abgesetzt werden.

   Wenn die Größe des Protokollspeichers, der den Protokollsatz enthält, der die älteste Seite im Pufferpool aktualisiert hat, und die aktuelle Protokollposition den vom Parameter 'softmax' zugelassenen Wert überschreiten, wird diese Situation als LSN-Abstimmungsverlust (LSNGAP-Bedingung) der Datenbank bezeichnet. Wenn die Protokollfunktion unter der Standardmethode der Seitenbereinigung erkennt, dass ein LSN-Abstimmungsverlust aufgetreten ist, startet sie die Seitenlöschfunktionen, um alle Seiten, die zu dieser Situation beitragen, auf die Platte zu schreiben. Das heißt, sie schreibt die Seiten, die älter sind, als durch den Parameter 'softmax' zugelassen wird, auf die Platte. Seitenlöschfunktionen sind über einen gewissen Zeitraum, in dem kein LSN-Abstimmungsverlust auftritt, nicht aktiv. Wenn dann ein LSN-Abstimmungsverlust auftritt, werden die Seitenlöschfunktionen aktiviert und schreiben eine große Anzahl

von Seiten auf die Platte, bevor sie wieder inaktiv werden. Dies kann zu einer Sättigung des E/A-Subsystems führen, die wiederum andere Agenten beeinträchtigen kann, die gerade Seiten lesen oder schreiben. Darüber hinaus ist es möglich, dass bis zu dem Zeitpunkt, zu dem ein LSN-Abstimmungsverlust ausgelöst wird, die Seitenlöschfunktionen die Seiten schon nicht mehr schnell genug bereinigen können und DB2 am Ende keinen Protokollspeicherbereich mehr zur Verfügung hat.

Die alternative Methode der Seitenbereinigung moduliert diese Funktionsweise, indem sie die gleiche Anzahl von Schreiboperationen über einen größeren Zeitraum verteilt. Die Löschfunktionen führen dies aus, indem sie proaktiv nicht nur Seiten bereinigen, die sich zurzeit in einer Situation mit LSN-Abstimmungsverlust befinden, sondern auch die Seiten, die in Anbetracht des aktuellen Aktivitätsvolumens bald in eine solche Situation kommen werden.

Zur Verwendung der neuen Methode der Seitenbereinigung setzen Sie die Registrierdatenbankvariable DB2_USE_ALTERNATE_PAGE_CLEANING auf den Wert ON.

Vorablesen von Daten in den Pufferpool

Vorablesen von Seiten bedeutet, dass eine oder mehrere Seiten von der Platte in der Erwartung abgerufen werden, dass sie von einer Anwendung angefordert werden. Durch das Vorablesen von Index- und Datenseiten in den Pufferpool kann die Leistung verbessert werden, indem die E/A-Wartezeit verringert wird. Darüber hinaus wird die Effizienz des Vorablesezugriffs durch parallele E/A-Operationen erhöht.

Es gibt zwei Kategorien des Vorablesezugriffs:

- **Sequenzieller Vorablesezugriff:** Ein Mechanismus, mit dem aufeinander folgende Seiten in den Pufferpool gelesen werden, bevor die Seiten von der Anwendung angefordert werden.
- **Vorablesezugriff über Listen:** Wird manchmal als *sequenzieller Vorablesezugriff über Listen* bezeichnet. Ist eine effiziente Methode zum Vorablesen von Datenseiten, die nicht aufeinander folgen.

Diese beiden Methoden zum Lesen von Datenseiten erfolgen zusätzlich zu einem Lesevorgang durch den Datenbankmanageragenten. Ein Lesevorgang des Datenbankmanageragenten wird verwendet, wenn nur eine oder wenige aufeinander folgende Seiten abgerufen werden, jedoch nur eine Seite von Daten übertragen wird.

Vorablesezugriff und partitionsinterne Parallelität

Das Vorablesen ist wichtig für die Leistung der partitionsinternen Parallelität, bei der mehrere Subagenten beim Durchsuchen eines Index oder einer Tabelle verwendet werden. Solche parallelen Suchoperationen ziehen einen größeren Datenverarbeitungsdurchsatz nach sich, wodurch höhere Vorablesegeschwindigkeiten erforderlich werden.

Der Nachteil durch ungeeignetes Vorablesen ist bei parallelen Suchoperationen höher als bei seriellen Suchoperationen. Wenn für eine serielle Suchoperation kein Vorablesezugriff stattfindet, arbeit die Abfrage langsamer, weil der Agent immer auf E/A-Operationen warten muss. Wenn für eine parallele Suchoperation kein Vorablesezugriff stattfindet, müssen eventuell alle Subagenten warten, weil ein Subagent auf eine E/A-Operation wartet.

Aufgrund seiner Bedeutung wird der Vorablesezugriff bei partitionsinterner Parallelität intensiver durchgeführt. Die Funktion zur Sequenzerkennung toleriert größere Lücken zwischen benachbarten Seiten, sodass die Seiten als sequenziell betrachtet werden können. Die Breite dieser Lücken erhöht sich mit der Anzahl der an der Suchoperation beteiligten Subagenten.

Sequenzieller Vorablesezugriff

Durch das Lesen mehrerer aufeinander folgender Seiten in den Pufferpool in einer einzigen E/A-Operation kann der Systemaufwand für Ihre Anwendung wesentlich reduziert werden. Darüber hinaus können mehrere parallele E/A-Operationen zum Lesen mehrerer Bereiche von Seiten in den Pufferpool bei der Verringerung der E/A-Wartezeiten helfen.

Der Vorablesezugriff beginnt, wenn der Datenbankmanager bestimmt, dass sequenzielle E/A-Operationen zweckmäßig sind und dass durch den Vorablesezugriff die Leistung verbessert werden könnte. In solchen Fällen wie Tabellensuchen und Sortierungen in Tabellen kann der Datenbankmanager leicht feststellen, dass durch den sequenziellen Vorablesezugriff die E/A-Leistung verbessert wird. In diesen Fällen startet der Datenbankmanager den sequenziellen Vorablesezugriff automatisch. Das folgende Beispiel zeigt eine Abfrage, die wahrscheinlich eine Tabellensuche erforderlich macht und für die deshalb der sequenzielle Vorablesezugriff die nahe liegende Methode wäre:

```
SELECT NAME FROM EMPLOYEE
```

Auswirkungen des PREFETCHSIZE-Werts für Tabellenbereiche

Zur Definition der Anzahl vorabgelesener Seiten für jeden Tabellenbereich können Sie die Klausel PREFETCHSIZE in den Anweisungen CREATE TABLESPACE oder ALTER TABLESPACE verwenden. Der von Ihnen definierte Wert wird in der Spalte PREFETCHSIZE der Systemkatalogtabelle SYSCAT.TABLESPACES gespeichert.

Es empfiehlt sich, den Wert für PREFETCHSIZE explizit als ein Vielfaches der Anzahl von Tabellenbereichscontainern, der Anzahl physischer Platten unter jedem Container (wenn eine RAID-Einheit eingesetzt wird) und des Werts für EXTENTSIZE für Ihren Tabellenbereich zu definieren. Der Wert für EXTENTSIZE gibt die Anzahl von Seiten an, die der Datenbankmanager in einen Container schreibt, bevor er einen anderen Container verwendet. Ist zum Beispiel für EXTENTSIZE ein Wert von 16 Seiten festgelegt und hat der Tabellenbereich zwei Container, könnten Sie den Wert für die Anzahl der vorab gelesenen Seiten (PREFETCHSIZE) auf den Wert 32 setzen. Wenn fünf physische Platten pro Container vorhanden sind, können Sie die Menge der vorab gelesenen Daten auf 160 Seiten setzen.

Der Datenbankmanager überwacht die Verwendung des Pufferpools, um sicherzustellen, dass durch den Vorablesezugriff keine Seiten aus dem Pufferpool entfernt werden, wenn eine andere UOW (Unit of Work, Arbeitseinheit) sie noch benötigt. Zur Vermeidung von Problemen kann der Datenbankmanager die Anzahl der vorabgelesenen Seiten auf einen kleineren als den von Ihnen für den Tabellenbereich angegebenen Wert begrenzen.

Die Vorablesegröße (PREFETCHSIZE) kann erhebliche Auswirkungen auf die Leistung insbesondere für Suchoperationen in großen Tabellen haben. Verwenden Sie den Datenbanksystemmonitor und andere Systemmonitortools als Unterstützung bei der Optimierung des PREFETCHSIZE-Werts für Ihre Tabellenbereiche. Zum Beispiel könnten Sie folgende Arten von Informationen sammeln:

- Mit Überwachungsprogrammen für Ihr Betriebssystem können Sie feststellen, ob E/A-Wartezeiten für die Abfrage auftreten.
- Mithilfe des Datenelements *pool_async_data_reads (asynchrone Leseoperationen für Pufferpooldaten)*, das vom Datenbanksystemmonitor bereitgestellt wird, können Sie feststellen, ob Vorablesezugriffe stattfinden.

Wenn E/A-Wartezeiten auftreten und für die Abfrage der Vorablesezugriff aktiv ist, könnten Sie den Wert für PREFETCHSIZE erhöhen. Wenn die Vorablesefunktion nicht die Ursache für die E/A-Wartezeiten ist, verbessert eine Erhöhung des Werts für PREFETCHSIZE die Leistung Ihrer Abfrage nicht.

Bei allen Arten des Vorablesezugriffs können mehrere E/A-Operationen parallel ausgeführt werden, wenn für PREFETCHSIZE ein Vielfaches des Werts für EXTENTSIZE für den Tabellenbereich angegeben ist und sich die durch EXTENTSIZE definierten Bereiche des Tabellenbereichs in separaten Containern befinden. Konfigurieren Sie die Container so, dass sie getrennte physische Einheiten verwenden, um eine bessere Leistung zu erzielen.

Sequenzerkennung

In einigen Fällen ist es nicht von vornherein offensichtlich, dass durch einen Vorablesezugriff eine Leistungsverbesserung erreicht werden kann. In diesen Fällen kann der Datenbankmanager die E/A-Operationen überwachen und den Vorablesezugriff aktivieren, wenn sequenzielles Lesen von Seiten auftritt. In solchen Fällen wird der Vorablesezugriff vom Datenbankmanager nach Zweckmäßigkeit aktiviert und inaktiviert. Diese Art des sequenziellen Vorablesens wird als *Sequenzerkennung* bezeichnet und wird für Index- und Datenseiten angewandt. Mit dem Konfigurationsparameter *seqdetect* können Sie steuern, ob der Datenbankmanager mit Sequenzerkennung arbeiten soll.

Wenn die Sequenzerkennung zum Beispiel aktiviert ist, könnte die folgende SQL-Anweisung von einem sequenziellen Vorablesezugriff profitieren:

```
SELECT NAME FROM EMPLOYEE
   WHERE EMPNO BETWEEN 100 AND 3000
```

In diesem Beispiel könnte das Optimierungsprogramm vielleicht begonnen haben, die Tabelle mithilfe eines Index für die Spalte EMPNO zu durchsuchen. Wenn die Tabelle eine hohe Clusterbildung in Bezug auf diesen Index aufweist, dann sind die Leseoperationen für die Datenseiten beinahe sequenziell und ein Vorablesezugriff könnte zu einer Leistungsverbesserung führen. Infolgedessen findet ein Vorablesezugriff auf die Datenseiten statt.

In diesem Beispiel könnte auch ein Vorablesezugriff auf die Indexseiten erfolgen. Wenn eine große Anzahl von Indexseiten untersucht werden muss und der Datenbankmanager erkennt, dass ein sequenzielles Lesen der Indexseiten stattfindet, wird ein Vorablesezugriff auf Indexseiten durchgeführt.

Blockbasierte Pufferpools für besseren sequenziellen Vorablesezugriff

Das Vorablesen von Seiten vom Plattenspeicher ist aufgrund des E/A-Systemaufwands teuer. Der Durchsatz kann beträchtlich verbessert werden, wenn sich die Verarbeitung mit den E/A-Operationen überlappen kann. Die meisten Plattformen stellen Hochleistungsmechanismen bereit, die zusammenhängende Seiten vom Plattenspeicher in nicht zusammenhängende Bereiche des Hauptspeichers einlesen. Diese Mechanismen werden gewöhnlich als gestreutes Lesen (engl. *scattered read*)

oder *über einen Vektor definierte E/A* bezeichnet. Auf einigen Plattformen kann die Leistung dieser Mechanismen nicht mit E/A-Operationen mit großen Blockgrößen konkurrieren.

Standardmäßig sind Pufferpools seitenbasiert. Dies bedeutet, dass zusammenhängende Seiten auf dem Plattenspeicher in nicht zusammenhängende Seiten im Arbeitsspeicher vorab gelesen werden. Der sequenzielle Vorablesezugriff kann verbessert werden, wenn zusammenhängende Seiten vom Plattenspeicher in zusammenhängende Seiten in einem Pufferpool gelesen werden können.

Zu diesem Zweck können Sie blockbasierte Pufferpools erstellen. Ein blockbasierter Pufferpool besteht sowohl aus einem Seitenbereich als auch aus einem Blockbereich. Der Seitenbereich ist für nicht sequenzielle Vorableseoperationen erforderlich. Der Blockbereich besteht aus Blöcken, wobei jeder Block eine angegebene Anzahl zusammenhängender Seiten enthält, die als *Blockgröße* bezeichnet wird.

Die optimale Nutzung eines blockbasierten Pufferpools hängt von der angegebenen Blockgröße ab. Die Blockgröße ist die Granularität, mit der E/A-Server bei sequenziellen Vorablesezugriffen eine blockbasierte E/A-Operation in Betracht ziehen. Die EXTENTSIZE-Größe ist die Granularität, mit der Tabellenbereiche in Containern einheitenübergreifend gespeichert werden (Striping). Da mehrere Tabellenbereiche mit unterschiedlichen EXTENTSIZE-Werten an einen Pufferpool gebunden werden können, der mit der gleichen Blockgröße definiert ist, beachten Sie, in welcher Beziehung die EXTENTSIZE-Größe und die Blockgröße bei der effizienten Nutzung des Pufferpoolspeichers zueinander stehen. Pufferpoolspeicher kann unter folgenden Umständen verschwendet werden:

- Wenn der Wert für EXTENTSIZE, durch den die Größe für Vorableseanforderungen festgelegt wird, kleiner als der für den Pufferpool angegebene Wert für BLOCK_SIZE ist.
- Wenn sich einige Seiten, die durch eine Vorableseanforderung angefordert werden, bereits im Seitenbereich des Pufferpools befinden.

Der E/A-Server lässt einige verschwendete Seiten in jedem Pufferpoolblock zu. Wenn jedoch zu große Teile eines Blocks verschwendet würden, führt der E/A-Server ein nicht blockbasiertes Vorablesen in den Seitenbereich des Pufferpools durch. Dies führt nicht zu einer optimalen Leistung.

Zur Erzielung einer optimalen Leistung binden Sie Tabellenbereiche der gleichen EXTENTSIZE-Größe an einen Pufferpool mit einer Blockgröße, die der EXTENTSIZE-Größe der Tabellenbereiche entspricht. Eine gute Leistung lässt sich erreichen, wenn der Wert für EXTENTSIZE größer als die Blockgröße ist, jedoch nicht, wenn der Wert für EXTENTSIZE kleiner als die Blockgröße ist.

Verwenden Sie die Anweisungen CREATE und ALTER BUFFERPOOL zur Erstellung eines blockbasierten Pufferpools.

Anmerkung: Blockbasierte Pufferpools sind für den sequenziellen Vorablesezugriff gedacht. Wenn Ihre Anwendungen keinen sequenziellen Vorablesezugriff nutzen, wird der Blockbereich im Pufferpool verschwendet.

Vorablesezugriff über Listen

Der *Vorablesezugriff über Listen* oder *sequenzielle Vorablesezugriff über Listen* ist eine effiziente Methode, auf Daten zuzugreifen, auch wenn die benötigten Datenseiten nicht in aufeinander folgender Reihenfolge vorliegen. Der Vorablesezugriff über Listen kann in Verbindung mit dem Zugriff über einen oder mehrere Indizes verwendet werden.

Wenn das Optimierungsprogramm einen Index zum Zugriff auf Zeilen verwendet, kann es das Lesen der Datenseiten verzögern, bis alle Zeilen-IDs (RIDs) aus dem Index empfangen wurden. Zum Beispiel könnte das Optimierungsprogramm eine Indexsuche durchführen, um die abzurufenden Zeilen und Datenseiten zu ermitteln, wenn zuvor der Index IX1 definiert wurde:

```
INDEX IX1:  NAME    ASC,
                        DEPT    ASC,
                        MGR     DESC,
                        SALARY  DESC,
                        YEARS   ASC
```

Betrachten Sie folgende Suchbedingungen:

```
WHERE NAME BETWEEN 'A' and 'I'
```

Wenn die Daten in Bezug auf diesen Index keine Clusterbildung aufweisen, enthält der Vorablesezugriff über Listen einen Schritt zum Sortieren der durch die Indexsuche ermittelten Liste von Zeilen-IDs (RIDs).

Konfiguration von E/A-Servern für Vorablesezugriff und Parallelität

Zur Aktivierung des Vorablesezugriffs startet der Datenbankmanager zum Lesen von Datenseiten separate Steuerthreads, die als *E/A-Server* bezeichnet werden. Infolgedessen gliedert sich die Verarbeitung einer Abfrage in zwei parallele Aktivitäten: Datenverarbeitung (CPU) und E/A-Operationen für Datenseiten. Die E/A-Server warten auf Vorableseanforderungen aus der CPU-Verarbeitungsaktivität. Diese Vorableseanforderungen enthalten eine Beschreibung der E/A-Operationen, die zur Erfüllung der Abfrage benötigt werden. Die möglichen Vorablesemethoden bestimmen, wann und wie der Datenbankmanager die Vorableseanforderungen generiert.

Durch Konfigurieren einer ausreichenden Anzahl von E/A-Servern mithilfe des Konfigurationsparameters *num_ioservers* kann die Leistung von Abfragen, für die der Vorablesezugriff auf Daten verwendet werden kann, erheblich gesteigert werden. Setzen Sie den Wert des Parameters *num_ioservers* mindestens auf die Zahl der physischen Platten in der Datenbank, um die Möglichkeit für parallele E/A-Operationen zu maximieren.

Es ist besser, die Anzahl von E/A-Servern zu hoch als zu niedrig anzusetzen. Wenn Sie zusätzliche E/A-Server angeben, werden diese Server nicht verwendet, und ihre Speicherseiten werden ausgelagert. Infolgedessen wird die Leistung nicht beeinträchtigt. Jeder E/A-Serverprozess hat eine Nummer. Der Datenbankmanager verwendet immer den Prozess mit der niedrigsten Nummer, sodass einige der Prozesse mit höheren Nummern möglicherweise nie zum Einsatz kommen.

Bei der Schätzung, wie viele E/A-Server eventuell erforderlich sind, sollten Sie folgende Punkte beachten:

- Die Anzahl der Datenbankagenten, die Vorableseanforderungen an die E/A-Serverwarteschlange gleichzeitig schreiben könnten.
- Der höchste Grad, bis zu dem die E/A-Server parallel arbeiten können.

Sie sollten die Einstellung des Werts des Datenbankkonfigurationsparameters *num_ioservers* auf AUTOMATIC in Betracht ziehen, damit DB2 intelligente Werte auf der Basis der Systemkonfiguration auswählen kann.

Konfiguration für asynchrone E/A

Auf einigen Plattformen arbeitet DB2 mit asynchroner Ein-/Ausgabe (AIO), um die Leistung von Aktivitäten wie Seitenlöschfunktionen und Vorablesezugriffe zu verbessern. Asynchrone E/A (AIO) ist am effektivsten, wenn die Daten in den Containern über mehrere Platten verteilt sind. Die Leistung profitiert außerdem von einer Optimierung der AIO-Infrastruktur des zugrunde liegenden Betriebssystems.

Unter AIX können Sie zum Beispiel die AIO für das Betriebssystem optimieren. Wenn die asynchrone Ein-/Ausgabe entweder mit SMS-Containern oder mit DMS-Dateicontainern arbeitet, verwalten Betriebssystemprozesse, so genannte AIO-Server, die E/A-Operationen. Eine kleine Anzahl solcher Server kann den Vorteil der asynchronen Ein-/Ausgabe einschränken, indem sie die Anzahl von AIO-Anforderungen begrenzt. Verwenden Sie zur Konfiguration der Anzahl von AIO-Servern unter AIX die *smit*-AIO-Parameter *minservers* und *maxservers*.

Illustration des Vorablesens mit paralleler E/A: Die folgende Abbildung veranschaulicht, wie E/A-Server zum Vorablesen von Daten in den Pufferpool verwendet werden.

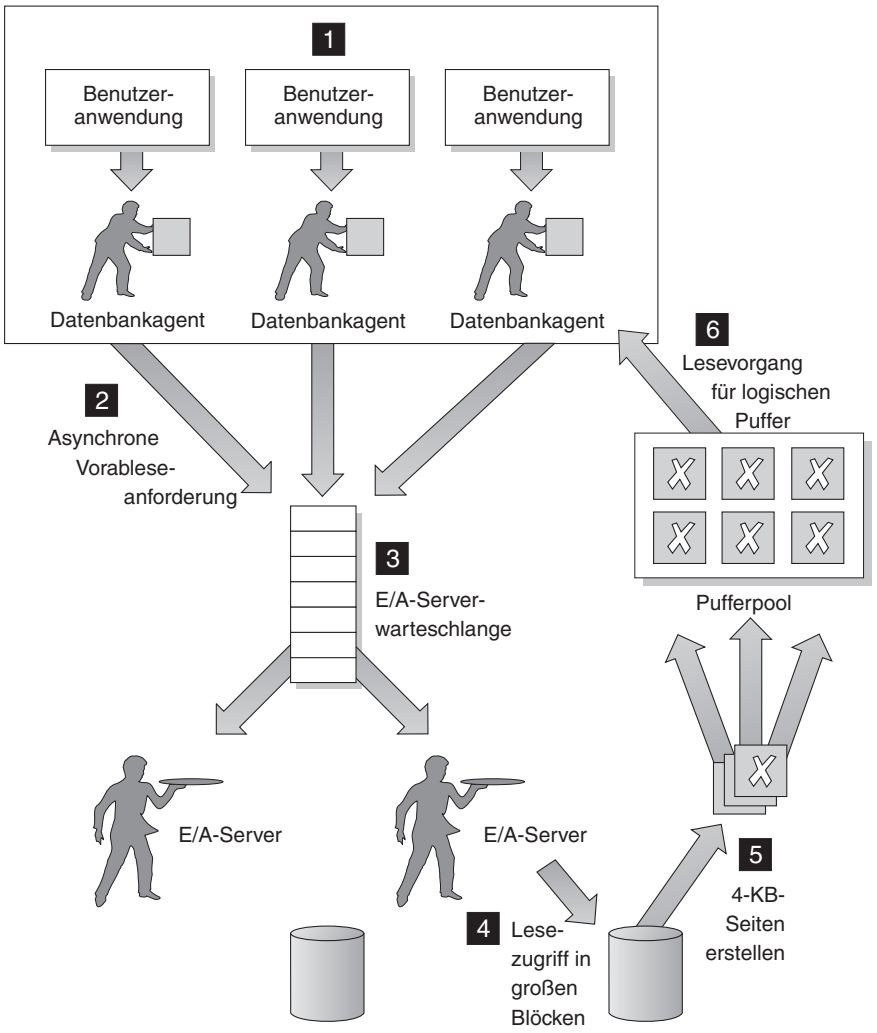


Abbildung 15. Vorablesen von Daten mithilfe von E/A-Servern

1 Die Benutzeranwendung übergibt die Anforderung an den Datenbankagenten, der der Benutzeranwendung vom Datenbankmanager zugeordnet wurde.

2, **3** Der Datenbankagent legt fest, dass ein Vorablesezugriff erfolgen soll, um die angeforderten Daten abzurufen und so die Anforderung zu erfüllen, und schreibt eine Vorableseanforderung an die E/A-Serverwarteschlange.

4, **5** Der erste verfügbare E/A-Server liest die Vorableseanforderung aus der Warteschlange und liest die Daten aus dem Tabellenbereich in den Pufferpool. Die Anzahl von E/A-Servern, die Daten gleichzeitig aus einem Tabellenbereich abrufen können, hängt von der Anzahl der Vorleseanforderungen in der Warteschlange sowie von der Anzahl von E/A-Servern ab, die durch den Konfigurationsparameter *num_ioservers* der Datenbank konfiguriert sind.

6 Der Datenbankagent führt die erforderlichen Operationen an den Datenseiten im Pufferpool durch und liefert das Ergebnis an die Benutzeranwendung zurück.

Verwaltung der parallelen Ein-/Ausgabe: Wenn mehrere Container für einen Tabellenbereich vorhanden sind, kann der Datenbankmanager die *parallele Ein-/Ausgabe* einleiten, bei der der Datenbankmanager mehrere E/A-Server zur Verarbeitung der E/A-Anforderungen einer einzelnen Abfrage verwendet. Jeder E/A-Server verarbeitet die E/A-Operationen für einen anderen Container, sodass mehrere Container parallel gelesen werden können. Die parallele Durchführung der E/A-Operationen kann zu bedeutenden Verbesserungen beim E/A-Durchsatz führen.

Obwohl ein getrennter E/A-Server die E/A-Operationen für jeden Container durchführen kann, ist die tatsächliche Anzahl der E/A-Server, die parallele E/A-Operationen durchführen können, auf die Anzahl der physischen Einheiten begrenzt, über die die angeforderten Daten verteilt sind. Aus diesem Grund benötigen Sie so viele E/A-Server, wie physische Einheiten vorhanden sind.

Die parallele Ein-/Ausgabe wird in folgenden Fällen unterschiedlich eingeleitet:

- **Sequenzieller Vorablesezugriff**

  Beim sequenziellen Vorablesezugriff wird die parallele E/A initialisiert, wenn der Wert für PREFETCHSIZE (Menge der vorab gelesenen Daten) ein Vielfaches des Werts für EXTENTSIZE eines Tabellenbereichs ist. Jede Vorableseanforderung wird dann in viele kleine Anforderungen aufgeteilt, die sich an den Grenzen der durch EXTENTSIZE definierten Bereiche orientieren. Diese kleinen Anforderungen werden dann verschiedenen E/A-Servern zugeordnet.

- **Vorablesezugriff über Listen**

  Beim Vorablesezugriff über Listen wird jede Liste von Seiten in Abhängigkeit von den Containern, in denen die Datenseiten gespeichert sind, in kleinere Listen unterteilt. Diese kleineren Listen werden dann verschiedenen E/A-Servern zugeordnet.

- **Backup und Restore von Datenbanken oder Tabellenbereichen**

  Für Backups oder Restores von Daten ist die Anzahl paralleler E/A-Anforderungen gleich der Größe des Backup-Puffers dividiert durch den Wert von EXTENTSIZE. Der Maximalwert ist gleich der Anzahl von Containern.

- **Restore einer Datenbank oder eines Tabellenbereichs**

  Für den Restore von Daten werden parallele E/A-Anforderungen in gleicher Weise wie beim sequenziellen Vorablesezugriff eingeleitet und aufgeteilt. Die Daten werden nicht im Pufferpool wiederhergestellt, sondern direkt aus dem Restorepuffer auf die Platte versetzt.
- **Laden von Daten**

  Beim Laden von Daten können Sie den Grad der E/A-Parallelität mit der Option DISK_PARALLELISM des Befehls LOAD angeben. Wenn diese Option nicht angegeben wird, verwendet der Datenbankmanager einen Standardwert, der auf der kumulativen Anzahl von Tabellenbereichscontainern für alle Tabellenbereiche basiert, die der Tabelle zugeordnet sind.

Für eine optimale Leistung bei paralleler E/A sollten Sie folgende Voraussetzungen erfüllen:

- Es ist eine ausreichende Anzahl E/A-Server vorhanden. Geben Sie geringfügig mehr E/A-Server an als die Anzahl von Containern, die für alle Tabellenbereiche in der Datenbank verwendet werden.
- Die Werte für EXTENTSIZE und PREFETCHSIZE sind für den Tabellenbereich angemessen. Zur Vermeidung einer übermäßigen Nutzung des Pufferpools sollte der Wert für PREFETCHSIZE nicht zu groß sein. Eine ideale Größe ist ein Vielfaches des Werts für EXTENTSIZE, der Anzahl physischer Platten unter jedem Container (wenn eine RAID-Einheit eingesetzt wird) und der Anzahl der Tabellenbereichscontainer. Der Wert für EXTENTSIZE sollte relativ klein sein, wobei sich ein Wert zwischen 8 und 32 Seiten empfiehlt.
- Die Container befinden sich auf separaten physischen Laufwerken.
- Alle Container haben dieselbe Größe, um einen konsistenten Grad an Parallelität zu gewährleisten.

  Wenn ein oder mehrere Container kleiner als die anderen sind, verringern sie das Potenzial für das optimierte parallele Vorablesen. Betrachten Sie die folgenden Beispiele:
  - Wenn ein kleinerer Container gefüllt ist, werden weitere Daten in den übrigen Containern gespeichert, wodurch sich eine ungleichmäßige Auslastung der Container ergibt. Ungleichmäßig ausgelastete Container beeinträchtigen die Leistung des parallelen Vorablesens, da die Anzahl von Containern, aus denen Daten vorab gelesen werden können, eventuell kleiner ist als die Gesamtanzahl von Containern.
  - Wenn ein kleinerer Container zu einem späteren Zeitpunkt hinzugefügt wird und die Daten neu verteilt werden, enthält der kleinere Container weniger Daten als die anderen Container. Diese im Verhältnis zu den anderen Containern kleine Menge von Daten führt nicht zu einer Optimierung des parallelen Vorablesens.
  - Wenn ein Container größer ist und alle anderen Container vollständig gefüllt werden, wird dieser Container zum einzigen Container, in dem weitere Daten gespeichert werden. Beim Zugriff auf diese weiteren Daten kann der Datenbankmanager keinen parallelen Vorablesezugriff durchführen.
- Es ist eine angemessene E/A-Kapazität vorhanden, wenn partitionsinterne Parallelität verwendet wird. Auf SMP-Maschinen kann die partitionsinterne Parallelität die für eine Abfrage benötigte Zeit reduzieren, indem die Abfrage auf mehreren Prozessoren ausgeführt wird. Es ist eine ausreichende E/A-Kapazität erforderlich, um jeden Prozessor auszulasten. In der Regel sind zusätzliche physische Laufwerke erforderlich, um diese E/A-Kapazität bereitzustellen.

  Der Wert für PREFETCHSIZE muss höher sein, um ein Vorablesen mit höheren Geschwindigkeiten und eine effektive Nutzung der E/A-Kapazität zu ermöglichen.

Die Anzahl der erforderlichen physischen Laufwerke hängt von der Geschwindigkeit und der Kapazität der Laufwerke und des E/A-Busses sowie von der Geschwindigkeit der Prozessoren ab.

Optimieren der Sortierleistung: Da Abfragen häufig sortierte oder gruppierte Ergebnisse erfordern, werden häufig Sortierungen angefordert, und eine geeignete Konfiguration der Sortierspeicherbereiche spielt für eine gute Abfrageleistung eine wichtige Rolle. Sortieren ist in folgenden Fällen erforderlich:

- Es gibt keinen Index, um eine angeforderte Reihenfolge (z. B. durch eine Anweisung SELECT mit einer Klausel ORDER BY) der Daten herzustellen.
- Es gibt einen Index, aber Sortieren ist effizienter als der Zugriff über den Index.
- Ein Index wird erstellt.
- Ein Index wird gelöscht, wodurch eine Sortierung der Indexseitennummern verursacht wird.

Elemente mit Auswirkung auf das Sortieren

Die folgenden Faktoren wirken sich auf die Sortierleistung aus:

- Die Einstellungen für die folgenden Datenbankkonfigurationsparameter:
  - Die Zwischenspeichergröße für Sortierlisten (*sortheap*) gibt die Größe des Speichers an, der für jede Sortierung verwendet wird.
  - Der Schwellenwert für Sortierspeicher (*sheapthres*) und der Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge (*sheapthres_shr*) steuern die Gesamtgröße des Speichers, der für das Sortieren in der gesamten Instanz für alle Sortiervorgänge verfügbar ist.
- Die Anzahl an Anweisungen in einer Auslastung, für die eine große Menge an Sortierungen erforderlich sind
- Vorhandene oder fehlende Indizes, die zur Vermeidung unnötiger Sortiervorgänge dienen könnten
- Verwendung von Anwendungslogik, die das Sortieraufkommen nicht minimiert
- Paralleles Sortieren, das die Leistung der Sortierungen erhöht, aber nur auftreten kann, wenn die Anweisung partitionsinterne Parallelität verwendet
- Ob die Sortierung *mit* oder *ohne* Überlauf stattfindet. Wenn die sortierten Daten nicht vollständig in den Sortierspeicher passen, das heißt in den Speicherblock, der jedes Mal zugeordnet wird, wenn eine Sortierung durchgeführt wird, laufen die Daten in temporäre Datenbanktabellen über. Sortierungen, die keinen Überlauf verursachen, zeigen stets eine bessere Leistung als solche, bei denen ein Überlauf auftritt.
- Ob die Ergebnisse der Sortierung *über eine Pipe* oder *nicht über eine Pipe* geleitete werden. Wenn sortierte Daten direkt zurückgegeben werden können, ohne dass eine temporäre Tabelle zum Speichern der endgültigen sortierten Liste von Daten erforderlich ist, handelt es sich um einen Sortiervorgang mit Piping (d. h. die Daten werden über eine Pipe zurückgegeben). Wenn die sortierten Daten über eine temporäre Tabelle zurückgegeben werden müssen, wird dies als Sortierung ohne Piping bezeichnet. Eine Sortierung mit Piping ist immer effizienter als eine Sortierung ohne Piping.

  Beachten Sie außerdem, dass bei einer Sortierung mit Piping der Sortierzwischenspeicher nicht freigegeben wird, bevor die Anwendung den dieser Sortierung zugeordneten Cursor schließt. Eine Sortierung mit Piping kann weiter Speicher belegen, bis der Cursor geschlossen wird.

Im Allgemeinen sollte der für die gesamte Instanz verfügbare Sortierspeicher (*sheapthres*) so groß wie möglich sein, ohne übermäßiges Seitenauslagern (Paging) zu

verursachen. Obwohl eine Sortierung vollständig im Sortierspeicher durchgeführt werden kann, kann dies zu einem übermäßigen Auslagern von Seiten führen. In diesem Fall geht der Vorteil eines großen Sortierzwischenspeichers verloren. Aus diesem Grund sollten Sie einen Betriebssystemmonitor verwenden, um alle Änderungen in der Auslagerung von Seiten durch das System zu verfolgen, wenn Sie die Konfigurationsparameter für das Sortieren anpassen.

Techniken zur Verwaltung der Sortierleistung

Ermitteln Sie bestimmte Anwendungen und Anweisungen, bei denen die Sortierung ein wesentliches Leistungsproblem darstellt:

1. Richten Sie Ereignismonitore (Event Monitors) auf Anwendungs- und Anweisungsebene ein, um Unterstützung bei der Ermittlung von Anwendungen mit der längsten Gesamtsortierzeit zu erhalten.
2. Ermitteln Sie innerhalb dieser Anwendungen die Anweisungen mit der längsten *Gesamtsortierzeit*.

   Sie können auch die EXPLAIN-Tabellen durchsuchen, um die Abfragen mit Sortieroperationen zu ermitteln.
3. Verwenden Sie diese Anweisungen als Eingabe für den Designadvisor, der Indizes identifiziert und optional erstellt, um den Sortierbedarf zu reduzieren.

Verwenden Sie den Datenbanksystemmonitor und Vergleichstesttechniken zur Unterstützung bei der Einstellung der Konfigurationsparameter *sortheap* und *sheapthres*. Gehen Sie für die einzelnen Datenbankmanager und Datenbanken folgendermaßen vor:

1. Erstellen Sie eine repräsentative Auslastung und führen Sie sie aus.
2. Sammeln Sie für jede betroffene Datenbank Durchschnittswerte für die folgenden Leistungsvariablen über den Auslastungszeitraum der Vergleichstests:
   - Gesamter verwendeter Sortierspeicher (der Wert des Monitorelements sort_heap_allocated)
   - Aktive Sortiervorgänge und aktive Hash-Joins (die Werte der Monitorelemente active_sorts und active_hash_joins).
3. Setzen Sie den Parameter *sortheap* auf den Durchschnittswert für den *gesamten verwendeten Sortierspeicher* für jede Datenbank.

   Anmerkung: Nachdem die DB2-Binärsortierung mit Teilschlüssel verbessert wurde, sodass auch nicht ganzzahlige Datentypschlüssel unterstützt werden, ist beim Sortieren langer Schlüssel zusätzlicher Speicher erforderlich. Wenn lange Schlüssel für Sortierungen verwendet werden, müssen Sie möglicherweise den Wert für den Konfigurationsparameter *sortheap* erhöhen.
4. Definieren Sie den Wert für *sheapthres*. Gehen Sie wie folgt vor, um eine angemessene Größe zu schätzen:
   a. Stellen Sie fest, welche Datenbank in der Instanz über den größten Wert für *sortheap* verfügt.
   b. Ermitteln Sie die durchschnittliche Größe des Sortierspeichers für diese Datenbank.

      Wenn die Ermittlung des Durchschnittswerts zu aufwendig ist, verwenden Sie als Wert 80% des maximalen Sortierspeichers.
   c. Setzen Sie den Wert für *sheapthres* auf die Durchschnittsanzahl der aktiven Sortiervorgänge multipliziert mit der oben berechneten Durchschnittsgröße des Sortierspeichers.

Dies ist die empfohlene Anfangseinstellung. Anschließend können Sie mithilfe von Vergleichstests diesen Wert optimieren.

Mithilfe der automatischen Speicheroptimierungsfunktion können Sie automatisch und dynamisch für die Sortierung erforderlichen Speicherressourcen zuordnen und die Zuordnung wieder aufheben. Gehen Sie wie folgt vor, um diese Funktion zu verwenden:

- Aktivieren Sie die automatische Speicheroptimierung für die Datenbank, indem Sie den Konfigurationsparameter *self_tuning_mem* auf "ON" setzen.
- Setzen Sie die Konfigurationsparameter *sortheap* und *sheapthres_shr* auf "AUTOMATIC".
- Setzen Sie den Konfigurationsparameter *sheapthres* auf "0".

Aufrechterhalten der Organisation von Tabellen und Indizes

Mit der Zeit können Daten in Ihren Tabellen einer gewissen Fragmentierung unterliegen, die eine Vergrößerung Ihrer Tabellen und Indizes verursacht, da Datensätze über eine zunehmende Anzahl von Datenseiten verteilt werden. Dies kann dazu führen, dass die Anzahl der Seiten ansteigt, die während der Ausführung von Abfragen gelesen werden müssen. Eine Reorganisation Ihrer Tabellen und Indizes sorgt für eine kompaktere Speicherung Ihrer Daten und gibt auf diese Weise verschwendeten Speicherplatz frei und verbessert den Datenzugriff.

Folgende Schritte sind für eine Index- oder Tabellenreorganisation auszuführen:

1. Ermitteln Sie, ob und welche Tabellen oder Indizes reorganisiert werden müssen.
2. Wählen Sie eine Reorganisationsmethode.
3. Führen Sie die Reorganisation für die ermittelten Objekte aus.
4. Überwachen Sie den Fortschritt der Reorganisation.
5. Bei einer Onlinetabellenreorganisation können Sie den Reorganisationsprozess bei Bedarf anhalten und ihn zu einem späteren Zeitpunkt wieder fortsetzen.
6. Werten Sie das Ergebnis der Reorganisation aus, indem Sie feststellen, ob die Operation erfolgreich ausgeführt wurde oder fehlgeschlagen ist. Bei einer Offlinetabellenreorganisation und einer beliebigen Indexreorganisation erfolgt die Operation synchron und das Ergebnis der Reorganisation wird bei Abschluss der Ausführung des Befehls offenbar. Onlinetabellenreorganisationen werden asynchron ausgeführt, sodass Sie zur Bewertung des Ergebnisses auf die Verlaufsdatei zurückgreifen müssen.
7. Wenn Sie eine Onlinetabellenreorganisation ausgeführt haben, können Sie eine Recovery ausführen. Siehe „Recovery einer fehlgeschlagenen Onlinetabellenreorganisation" auf Seite 97.
8. Erfassen Sie Statistikdaten für reorganisierte Objekte.
9. Führen Sie einen Rebind für Anwendungen durch, die auf reorganisierte Objekte zugreifen.

Tabellenreorganisation

Nach zahlreichen Änderungen an Tabellendaten können sich logisch sequenzielle Daten auf physisch nicht sequenziellen Datenseiten befinden, sodass der Datenbankmanager zusätzliche Leseoperationen ausführen muss, um auf die Daten zuzugreifen. Darüber hinaus sind zusätzliche Leseoperationen auch erforderlich, wenn eine beträchtliche Anzahl von Zeilen gelöscht wurde. In einem solchen Fall können Sie eine Reorganisation der Tabelle in Betracht ziehen, um die Tabelle mit

dem Index in Übereinstimmung zu bringen und Speicherplatz wieder verfügbar zu machen. Sie können sowohl die Systemkatalogtabellen als auch Datenbanktabellen reorganisieren.

Anmerkung: Da die Reorganisation einer Tabelle in der Regel mehr Zeit als das Erfassen von Statistikdaten beansprucht, können Sie das Dienstprogramm RUNSTATS ausführen, um die Statistiken für Ihre Daten auf den aktuellen Stand zu bringen, und einen Rebind für Ihre Anwendungen durchzuführen. Wenn aktualisierte Statistiken keine Leistungsverbesserungen erzielen, kann eine Reorganisation eventuell helfen. Detaillierte Informationen zu den Optionen und der Funktionsweise des Dienstprogramms REORG TABLE finden Sie in der zugehörigen Befehlsreferenz.

Beachten Sie die folgenden Faktoren, die Hinweise darauf geben können, dass eine Tabelle reorganisiert werden sollte:

- Es ist ein hohes Aufkommen an INSERT-, UPDATE- und DELETE-Aktivitäten an Tabellen zu verzeichnen, auf die durch Abfragen zugegriffen wird.
- Es treten wesentliche Änderungen in der Leistung von Abfragen auf, die einen Index mit einem hohen Clusterverhältnis verwenden.
- Die Leistung wird durch die Ausführung von RUNSTATS zur Aktualisierung der Statistikdaten nicht besser.
- Der Befehl REORGCHK zeigt den Bedarf einer Reorganisation der Tabelle an.
- Die Kosten einer zunehmenden Verschlechterung der Abfrageleistung und die Kosten einer Reorganisation der Tabelle, zu denen die CPU-Zeit, die zur Ausführung erforderliche Zeit sowie der verringerte Grad des gemeinsamen Zugriffs aufgrund der Sperre, die das Dienstprogramm REORG für die Tabelle bis zum Ende der Reorganisation aktiviert, zu rechnen sind, müssen gegeneinander abgewogen werden.

Verringern der Erforderlichkeit der Reorganisation von Tabellen

Damit die Reorganisation einer Tabelle weniger häufig erforderlich wird, führen Sie nach dem Erstellen der Tabelle folgende Aufgaben aus:

- Ändern Sie die Tabelle (ALTER), um PCTFREE hinzuzufügen.
- Erstellen Sie einen Clusterindex mit PCTFREE für Index.
- Sortieren Sie die Daten.
- Laden Sie die Daten.

Wenn Sie diese Maßnahmen durchgeführt haben, tragen der Clusterindex der Tabelle und die Einstellung von PCTFREE für die Tabelle dazu bei, die ursprüngliche Sortierreihenfolge der Daten beizubehalten. Wenn genügend Speicherplatz auf den Tabellenseiten freigehalten wird, können neue Daten auf den richtigen Seiten eingefügt werden, um ihr Clustering in Bezug auf den Index beizubehalten. Mit wachsender Menge eingefügter Daten und zunehmender Füllung der Seiten der Tabelle werden Datensätze an das Ende der Tabelle angehängt, sodass die Tabelle allmählich ihr Clustering einbüßt.

Wenn Sie das Dienstprogramm REORG TABLE ausführen oder eine Sortierung und ein Laden von Daten nach der Erstellung eines Clusterindex durchführen, versucht der Index eine bestimmte Ordnung der Daten zu erhalten, wodurch sich die Werte für die Statistiken CLUSTERRATIO bzw. CLUSTERFACTOR verbessern, die durch das Dienstprogramm RUNSTATS erfasst werden.

Anmerkung: Die Erstellung von Tabellen mit mehrdimensionalem Clustering (MDC) kann die Erforderlichkeit der Reorganisation von Tabellen senken. Bei MDC-Tabellen bleibt die Clusterbildung für die Spalten erhalten, die Sie als Argumente der Klausel ORGANIZE BY DIMENSIONS in der Anweisung CREATE TABLE angeben. Allerdings kann das Dienstprogramm REORGCHK eine Reorganisation einer MDC-Tabelle empfehlen, wenn es ermittelt, dass zu viele ungenutzte Blöcke vorhanden sind oder dass Blöcke zusammengelegt werden sollten.

Auswählen einer Methode zur Tabellenreorganisation

Es gibt zwei verschiedene Methoden der Tabellenreorganisation: die *klassische* bzw. offline ausgeführte Reorganisation und die online bzw. mit der Option *inplace* ausgeführte Reorganisation.

Die Option INPLACE des Befehls REORG gibt eine Onlinereorganisation an. Ohne Angabe dieser Option wird eine Offlinereorganisation ausgeführt.

Beide Methoden der Reorganisation haben ihre Vor- und Nachteile. Diese werden im Folgenden zusammengefasst. Bei der Auswahl einer Reorganisationsmethode müssen Sie berücksichtigen, welche Methode für Ihre Prioritäten Vorteile bietet. Wenn zum Beispiel die Wiederherstellbarkeit im Fall eines Fehlers wichtiger ist als eine rasche Ausführung der Reorganisation, ist die Onlinereorganisation vielleicht die beste Lösung.

Vorteile der Offlinereorganisation

- Ist die schnellste Methode zur Tabellenreorganisation insbesondere, wenn keine Reorganisation von LOB/LONG-Daten erforderlich ist.
- Tabellen und Indizes sind nach Abschluss perfekt in Clustern angeordnet.
- Indizes werden erneut erstellt, nachdem die Tabelle reorganisiert wurde. Es ist kein separater Schritt zur erneuten Erstellung von Indizes erforderlich.
- Die Spiegelkopie kann in einem Tabellenbereich für temporäre Tabellen erstellt werden. Dadurch wird der Speicherplatzbedarf in dem Tabellenbereich verringert, der die Zieltabelle oder den Zielindex enthält.
- Ermöglicht die Angabe und Verwendung eines anderen Index als den Clusterindex, um die Daten auf neue Weise in Clustern anzuordnen, während die Onlinereorganisation den vorhandenen Clusterindex, sofern vorhanden, verwenden muss.

Nachteile der Offlinereorganisation

- Eingeschränkter Tabellenzugriff. Anwendungen haben lediglich Lesezugriff auf die Tabellen, und dies auch nur während der Sortierungs- und Erstellungsphase der Reorganisation.
- Großer Speicherplatzbedarf, da eine Spiegelkopiemethode verwendet wird.
- Weniger Kontrolle über den Reorganisationsprozess als bei einer Onlinereorganisation: Eine offline ausgeführte Reorganisation kann nicht angehalten und wieder gestartet werden.

Vorteile der Onlinereorganisation

- Anwendungen haben vollständigen Zugriff auf die Tabelle während der Reorganisation, mit Ausnahme der Abschneidephase (TRUNCATE).
- Sie haben mehr Kontrolle über den REORG-Prozess: Der Prozess wird asynchron im Hintergrund ausgeführt und kann angehalten, fortgesetzt und völlig gestoppt werden. Wenn zum Beispiel eine große Anzahl von Aktualisierungen oder Löschungen an einer Tabelle ausgeführt werden, können Sie den REORG-Prozess anhalten.

- Der Prozess ist wiederherstellbarer, wenn Fehler auftreten.
- Erfordert weniger Arbeitsspeicher, da die Tabelle in Teilen nacheinander verarbeitet wird.
- Vorteile der Reorganisation werden sofort mit dem Fortschreiten des REORG-Prozesses nutzbar.

Nachteile der Onlinereorganisation

- Kann je nach Typ der Transaktionen, die während der Reorganisation auf die Tabelle zugreifen, zu einem nicht optimalen Daten- oder Indexclustering führen.
- Tabellenseiten, die zu Beginn der Reorganisation reorganisiert werden, haben möglicherweise mehr Aktualisierungen und sind daher vielleicht fragmentierter als Tabellenseiten, die später in diesem Prozess reorganisiert werden.
- Leistung ist langsamer als bei einer Offlinereorganisation. Für eine normale Reorganisation mit Clustering (nicht nur zur Freigabe von Speicherplatz) kann die Ausführung einer Onlinereorganisation zwischen 10- und 20-mal länger dauern. (Bei einer Tabelle mit gleichzeitig an ihr ausgeführten Anwendungen oder bei einer großen Anzahl definierter Indizes kann sie noch erheblich mehr Zeit in Anspruch nehmen.)
- Die Onlinereorganisation ist ein wiederherstellbarer Prozess. Dies geht jedoch auf Kosten eines erhöhten Protokollierungsbedarfs. Es ist möglich, dass ein beträchtlicher Speicherplatz für die Protokollierung benötigt wird (bis zum Mehrfachen der Größe der Tabelle). Dies ist von der Anzahl der Zeilen, die während der Reorganisation versetzt werden, von der Anzahl der für die Tabelle definierten Indizes und von der Größe der Indizes abhängig.
- Indizes werden gepflegt, nicht erneut erstellt. Das bedeutet, dass eine nachfolgende Indexreorganisation erforderlich werden kann.

Tabelle 2. Vergleich zwischen Online- und Offlinereorganisation

Merkmal	**Offlinereorganisation**	**Onlinereorganisation**
Leistung	Schnell.	Langsam.
Clusterfaktor der Daten bei Abschluss	Gut.	Nicht optimal.
Gemeinsamer Zugriff (auf die Tabelle)	Reicht von 'kein Zugriff' (NO ACCESS) bis 'Lesezugriff' (READ ONLY).	Reicht von 'Lesezugriff' (READ ONLY) bis 'Vollzugriff'.
Datenspeicherplatzbedarf	Beträchtlich.	Nicht erheblich.
Protokollspeicherplatzbedarf	Nicht erheblich.	Kann beträchtlich sein.
Benutzerkontrolle (Möglichkeit zum Anhalten und Neustarten des Prozesses)	Weniger Kontrolle.	Mehr Kontrolle.
Wiederherstellbarkeit	Alles oder nichts: wird erfolgreich ausgeführt oder schlägt fehl.	Wiederherstellbar.
Indexneuerstellung	Wird ausgeführt.	Wird nicht ausgeführt.
Für alle Tabellentypen unterstützt?	Ja	Nein
Angabe eines anderen Index als des Clusterindex möglich?	Ja	Nein

Tabelle 2. Vergleich zwischen Online- und Offlinereorganisation (Forts.)

Merkmal	Offlinereorganisation	Onlinereorganisation
Verwendung eines Tabellenbereichs für temporäre Tabellen möglich?	Ja	Nein

Tabelle 3. Unterstützte Tabellentypen für Online- und Offlinereorganisationen

Tabellentyp	Offlinereorganisation unterstützt	Onlinereorganisation unterstützt
MDC-Tabellen (mit mehrdimensionalem Clustering)	Ja[1]	Nein
Bereichsclustertabelle (RCT-Tabelle)	Nein[2]	Nein
Tabellen im Anfügemodus	Nein	Nein[3]
Tabellen mit LONG/LOB-Daten	Ja[4]	Nein
Tabellen mit Indizes des Typs 1	Ja[5]	Nein
Systemkatalogtabellen: SYSIBM.SYSTABLES, SYSIBM.SYSSEQUENCES, SYSIBM.SYSDBAUTH, SYSIBM.SYSROUTINEAUTH	Ja	Nein

1. Da das Clustering automatisch über MDC-Blockindizes verwaltet wird, dient eine Reorganisation einer MDC-Tabelle lediglich zur Freigabe von Speicherplatz. Es kann kein Index angegeben werden, da der Blockindex verwendet wird.
2. Der Bereich einer RCT-Tabelle (Bereichsclustertabelle) bleibt stets in Clustern angeordnet.
3. Nach Inaktivierung des Anfügemodus kann eine Onlinereorganisation ausgeführt werden.
4. Die Reorganisation von LONG/LOB-Daten kann einen beträchtlichen Zeitraum in Anspruch nehmen. Die Reorganisation von LONG/LOB-Daten führt zu keiner Verbesserung der Abfrageleistung. Sie sollte nur zum Zweck der Speicherplatzfreigabe ausgeführt werden.
5. Nach der Reorganisation werden alle Indizes als Indizes des Typs 2 erneut erstellt.

Detaillierte Informationen zur Ausführung dieser Methoden zur Tabellenreorganisation finden Sie in den Beschreibungen der Syntax des Befehls REORG TABLE.

Überwachen des Fortschritts der Tabellenreorganisation

Informationen über den aktuellen Fortschritt der Tabellenreorganisation werden in die Protokolldatei für Datenbankaktivitäten geschrieben. Die Protokolldatei enthält einen Datensatz für jedes Reorganisationsereignis. Zum Anzeigen dieser Datei führen Sie den Befehl `db2 list history` für die Datenbank aus, in der sich die Tabelle befindet, die reorganisiert wird.

Sie können den Fortschritt der Tabellenreorganisation auch mithilfe von Tabellenmomentaufnahmen überwachen. Überwachungsdaten zur Tabellenreorganisation werden unabhängig von der Einstellung des Datenbankmonitorschalters für Tabellen aufgezeichnet.

Wenn ein Fehler auftritt, wird ein SQLCA-Auszug in die Protokolldatei geschrieben. Bei einer INPLACE-Tabellenreorganisation wird der Status mit PAUSED (angehalten) aufgezeichnet.

Offlinetabellenreorganisation

Bei der Offlinetabellenreorganisation wird ein Spiegelkopieverfahren verwendet, bei dem eine vollständige Kopie der zu reorganisierenden Tabelle erstellt wird.

Die klassische bzw. offline ausgeführte Tabellenreorganisation erfolgt in vier Phasen:

1. *Sortierung (SORT)*
   Wenn ein Index im Befehl REORG TABLE angegeben wird oder ein Clusterindex für die Tabelle definiert ist, werden die Zeilen der Tabelle zunächst nach diesem Index sortiert. Wenn die Option INDEXSCAN angegeben wird, wird eine Indexsuche zur Sortierung der Tabelle verwendet. Anderenfalls wird eine Tabellensuche zur Sortierung verwendet. Diese Phase wird nur für eine Reorganisation mit Clustering ausgeführt. Reorganisationen zur Freigabe von Speicherplatz beginnen mit der Erstellungsphase.
2. *Erstellung (BUILD)*
   In dieser Phase wird eine reorganisierte Kopie der gesamten Tabelle erstellt. Dies geschieht entweder in dem Tabellenbereich, in dem sich die zu reorganisierende Tabelle befindet, oder in einem Tabellenbereich für temporäre Tabellen, der im Befehl REORG angegeben wird.
3. *Ersetzung (REPLACE)*
   In dieser Phase wird das ursprüngliche Tabellenobjekt ersetzt, indem die Tabelle entweder aus dem Tabellenbereich für temporäre Tabellen zurückkopiert wird oder indem auf das neu erstellte Objekt in dem Tabellenbereich der Tabelle, die reorganisiert wird, gezeigt wird.
4. *Erneute Erstellung aller Indizes (RECREATE ALL INDEXES)*
   Alle für die Tabelle definierten Indizes werden erneut erstellt.

Mithilfe von Snapshot Monitor oder Verwaltungssichten mit Momentaufnahmen können Sie den Fortschritt der REORG TABLE-Operation überwachen und feststellen, in welcher Phase sich die Reorganisation zu einem bestimmten Zeitpunkt befindet.

Für offline ausgeführte Tabellenreorganisationen sind die Sperrbedingungen restriktiver als für online ausgeführte Reorganisationen. Während der Erstellung der Kopie ist ein Lesezugriff auf die Tabelle verfügbar. Während der Ersetzung der ursprünglichen Tabelle durch die reorganisierte Kopie sowie während der erneuten Erstellung der Indizes ist jedoch ein exklusiver Zugriff auf die Tabelle erforderlich.

Eine IX-Tabellenbereichssperre ist während des gesamten Reorganisationsprozesses erforderlich. Während der Erstellungsphase (BUILD) wird eine U-Sperre für die Tabelle aktiviert und beibehalten. Eine U-Sperre ermöglicht dem Sperreneigner, die Daten in der Tabelle zu aktualisieren. Jedoch kann keine andere Anwendung Daten aktualisieren. (Lesezugriff wird zugelassen.) Die U-Sperre wird auf eine Z-Sperre hochgestuft, wenn die Ersetzungsphase (REPLACE) gestartet wurde. Während dieser Phase können keine anderen Anwendungen auf die Daten zugreifen. Diese Sperre wird bis zum Abschluss der Reorganisation beibehalten.

Durch den offline ausgeführten Reorganisationsprozess wird eine Reihe von Dateien erstellt. Diese Dateien werden in Ihrem Datenbankverzeichnis gespeichert, wobei den Namen die Tabellenbereichs- und Objekt-IDs als Präfix vorangestellt werden. Zum Beispiel ist 0030002.ROR die Statusdatei für die Reorganisation einer Tabelle mit der Tabellenbereichs-ID 3 und der Tabellen-ID 2.

Die folgenden temporären Dateien werden während der Offlinereorganisation für eine SMS-Tabelle erstellt:

- .DTR: Datendatei der Spiegelkopie
- .LFR: Datei für Langfelddaten
- .LAR: Zuordnungsdatei für Langfelddaten
- .RLB: LOB-Datendatei
- .RBA: LOB-Zuordnungsdatei
- .BMR: Blockobjektdatei (für MDC-Tabellen)

Die folgende temporäre Datei wird während der Indexreorganisation erstellt:

- .IN1: Spiegelkopiedatei

Die folgenden temporären Dateien werden im Tabellenbereich für temporäre Systemtabellen während der Sortierungsphase erstellt:

- .TDA: Datendatei
- .TIX: Indexdatei
- .TLF: Datei für Langfelddaten
- .TLA: Zuordnungsdatei für Langfelddaten
- .TLB: LOB-Datei
- .TBA: LOB-Zuordnungsdatei
- .TBM: Blockobjektdatei

Die dem Reorganisationsprozess zugeordneten Dateien sollten nicht manuell aus dem System entfernt werden.

Ausführen von Offlinetabellenreorganisationen:

Offline ausgeführte Tabellenreorganisationen sind die schnellste Methode zur Defragmentierung von Tabellen. Die Reorganisation verringert den Speicherplatzbedarf für die Tabelle und verbessert den Datenzugriff und die Abfrageleistung.

Zur Reorganisation einer Tabelle benötigen Sie die Berechtigung SYSADM, SYSCTRL, SYSMAINT oder DBADM. Alternativ benötigen Sie das Zugriffsrecht CONTROL für die Tabelle. Sie müssen eine Datenbankverbindung zur Reorganisation einer Tabelle herstellen.

Wenn Sie die Tabellen ermittelt haben, für die eine Reorganisation erforderlich ist, können Sie das Dienstprogramm REORG für diese Tabellen und optional für die Indizes, die für diese Tabellen definiert sind, ausführen.

1. Führen Sie den Befehl REORG TABLE zur Reorganisation einer Tabelle über den Befehlszeilenprozessor (CLP) aus:

   ```
   db2 reorg table test.employee
   ```

   Zur Reorganisation einer Tabelle mit dem Tabellenbereich für temporäre Tabellen mit dem Namen `mytemp` geben Sie folgenden Befehl ein:

   ```
   db2 reorg table test.employee use mytemp
   ```

Zur Reorganisation der Tabelle sowie zur neuen Anordnung der Zeilen nach dem Index myindex geben Sie folgenden Befehl ein:

```
db2 reorg table test.employee index myindex
```

2. Zur Reorganisation einer Tabelle mithilfe einer SQL-Anweisung CALL setzen Sie den Befehl REORG TABLE unter Verwendung der Prozedur ADMIN_CMD ab:

   ```
   call sysproc.admin_cmd ('reorg table employee index myindex')
   ```

3. Zur Reorganisation einer Tabelle mit einer DB2-Administrator-API verwenden Sie die API 'db2REORG'.

Im Anschluss an die Reorganisation einer Tabelle sollten Sie Statistikdaten für die Tabelle erfassen, damit dem Optimierungsprogramm die aktuellsten und präzisesten Daten zur Auswertung von Abfragezugriffsplänen zur Verfügung stehen.

Recovery einer Offlinetabellenreorganisation:

Eine offline ausgeführte Tabellenreorganisation ist bis zu dem Zeitpunkt, zu dem die REPLACE-Phase begonnen hat, ein Alles-oder-nichts-Prozess. Dies bedeutet, dass die REORG TABLE-Operation bei einem Systemabsturz während der Sortierungs- oder Erstellungsphase (SORT- oder BUILD-Phase) rückgängig gemacht wird und nach der Recovery nach dem Systemabsturz nicht wiederholt wird. Stattdessen muss der Befehl REORG TABLE im Anschluss an die Recovery erneut abgesetzt werden.

Wenn Ihr System nach Eintritt in die Ersetzungsphase (REPLACE) abstürzt, muss die REORG TABLE-Operation bis zum Ende ausgeführt und abgeschlossen werden. Dies hängt damit zusammen, dass die gesamte Arbeit der Tabellenreorganisation bereits erledigt wurde und die ursprüngliche Tabelle eventuell nicht mehr verfügbar ist. Während der Recovery nach einem Systemabsturz ist die temporäre Datei für das reorganisierte Objekt erforderlich, nicht jedoch der Tabellenbereich für temporäre Tabellen, der für die Sortierung verwendet wurde. Die Recovery startet die Ersetzungsphase erneut von Beginn an. Daher sind alle Daten im Objekt der Kopie für die Recovery erforderlich. In diesem Fall besteht ein Unterschied zwischen SMS- und DMS-Tabellenbereichen: Das SMS-Objekt muss von einem Objekt in das andere kopiert werden, während in DMS auf das neu reorganisierte Objekt lediglich gezeigt und die ursprüngliche Tabelle gelöscht wird, wenn die Reorganisation im selben Tabellenbereich erfolgte. Indizes werden nicht erneut erstellt. Jedoch werden sie während der Recovery nach dem Systemabsturz als ungültig markiert, und die Datenbank folgt den normalen Regeln zur Bestimmung, wann sie erneut erstellt werden, das heißt, entweder beim Neustart der Datenbank oder beim ersten Zugriff auf die Indizes.

Bei einem Absturz in der Phase der Indexneuerstellung ist das neue Objekt bereits vorhanden, sodass kein Operationsschritt wiederholt wird. (Wie oben erwähnt, werden Indizes nicht erneut erstellt, sondern während der Recovery nach dem Systemabsturz als ungültig markiert. Die Datenbank folgt den normalen Regeln zur Bestimmung, wann sie erneut erstellt werden, das heißt, entweder beim Datenbankneustart oder beim ersten Zugriff auf die Indizes.)

Bei der aktualisierenden Recovery wird die REORG TABLE-Operation wiederholt, wenn die alte Version der Tabelle auf dem Datenträger vorhanden ist. Die aktualisierende Recovery verwendet die Satz-IDs (RIDs), die während der BUILD-Phase protokolliert wurden, um die Abfolge der Operationen, die zur Erstellung der reorganisierten Tabelle ausgeführt wurden, erneut anzuwenden und dadurch die BUILD- und die REPLACE-Phase zu wiederholen. Indizes werden wiederum wie

oben beschrieben als ungültig markiert. Dies bedeutet, dass keine Indexsuche oder Suchsortierung ausgeführt wird und dass lediglich der Tabellenbereich für temporäre Tabellen für die Kopie des reorganisierten Objekts erforderlich ist, wenn ursprünglich ein Tabellenbereich für temporäre Tabellen verwendet wurde. Bei der aktualisierenden Recovery können mehrere REORG TABLE-Operationen aufgrund der Parallelverarbeitung der Recovery gleichzeitig wiederholt werden. In diesem Fall ist die Plattenspeicherbelegung größer als während der Laufzeit.

Verbessern der Leistung von Offlinetabellenreorganisationen:

Die Leistung einer Offlinetabellenreorganisation wird weitgehend durch die Merkmale der Datenbankumgebung bestimmt.

Es gibt keinen nennenswerten Unterschied zwischen der Leistung von REORG TABLE im Modus NO ACCESS und der Leistung von REORG TABLE im Modus ALLOW READ ACCESS. Der einzige Unterschied besteht darin, dass DB2 für den Modus ALLOW READ ACCESS die Sperre für die Tabelle vor dem Ersetzen der Tabelle hochstuft. Daher muss das Dienstprogramm eventuell auf den Abschluss anderer bereits laufender Suchoperationen und die Freigabe der für diese Operationen aktivierten Sperren warten. Die Tabelle ist in beiden Fällen während der Phase der Indexneuerstellung der REORG TABLE-Operation nicht verfügbar.

Tipps zur Leistungsverbesserung

Wenn genügend Speicherplatz in dem Tabellenbereich vorhanden ist, verwenden Sie denselben Tabellenbereich für die ursprüngliche Tabelle und die reorganisierte Kopie der Tabelle, anstatt einen Tabellenbereich für temporäre Tabellen zu verwenden. Dies spart die Zeit, die zum Kopieren der Tabelle aus dem Tabellenbereich für temporäre Tabellen erforderlich ist.

- Ziehen Sie in Betracht, vor der Reorganisation einer Tabelle nicht benötigte Indizes zu löschen, damit während der REORG TABLE-Operation weniger Indizes verwaltet werden müssen.
- Stellen Sie sicher, dass die Vorablesezugriffsgröße (PREFETCHSIZE) der Tabellenbereiche, in denen sich die zu reorganisierende Tabelle befindet, korrekt definiert ist.
- Aktivieren Sie die partitionsinterne Parallelverarbeitung (Parameter INTRA_PARALLEL), sodass die erneute Erstellung von Indizes in Parallelverarbeitung erfolgen kann.
- Optimieren Sie die Datenbankkonfigurationsparameter *sortheap* und *sheapthres*, um den für Sortierungen verwendeten Speicherbereich zu steuern. Da jeder Prozessor eine private Sortieroperation ausführt, sollte der Wert für *sheapthres* mindestens dem Produkt aus *sortheap* x *anzahl_verwendeter_prozessoren* betragen.
- Stellen Sie sicher, dass benutzte Indexseiten möglichst rasch aus dem Pufferpool bereinigt werden, indem Sie die Anzahl der Seitenlöschfunktionen optimieren.

Onlinetabellenreorganisation

Online bzw. mit der Option INPLACE ausgeführte Tabellenreorganisationen (REORG TABLE) bieten dem Benutzer die Möglichkeit, eine Tabelle zu reorganisieren und gleichzeitig einen vollständigen Zugriff auf diese Tabelle zu gewähren. Eine Onlinetabellenreorganisation ermöglicht zwar einen ununterbrochenen Benutzerzugriff auf die Daten, allerdings ist ihre Leistung langsamer als die einer Offlinetabellenreorganisation.

Während einer Onlinetabellenreorganisation wird nicht die gesamte Tabelle gleichzeitig reorganisiert. Vielmehr werden Teile der Tabelle nacheinander reorganisiert.

Die Daten werden nicht in einen Tabellenbereich für temporäre Tabellen herauskopiert: Die Zeilen werden innerhalb des bestehenden Tabellenobjekts versetzt, um das Clustering wiederherzustellen, freie Speicherbereiche wieder freizugeben und Überlaufzeilen zu beseitigen.

Eine Onlinetabellenreorganisation (REORG TABLE) erfolgt in vier Hauptphasen:

1. Auswahl von N Seiten
   Während dieser Phase wählt DB2 N Seiten, wobei N der Größe eines Speicherbereichs (EXTENTSIZE) mit mindestens 32 sequenziell angeordneten Seiten entspricht, zur REORG TABLE-Verarbeitung aus.
2. Freimachen des Bereichs
   Wenn die N Seiten ausgewählt sind, versetzt die Onlinetabellenreorganisation alle Zeilen in diesem Bereich auf freie Seiten in der Tabelle. Jede Zeile, die versetzt wird, hinterlässt einen RP-Datensatz (RP - REORG TABLE Pointer, REORG TABLE-Zeiger), der die Satz-ID (RID) der neuen Position der Zeile enthält. Die Zeile wird auf freien Seiten in der Tabelle als RO-Datensatz (RO - REORG TABLE Overflow, REORG TABLE-Überlauf) eingefügt, der die Daten enthält.
   Wenn die Tabellenreorganisation das Versetzen einer Gruppe von Zeilen beendet hat, wartet sie ab, bis alle laufenden Datenzugriffe, die an der Tabelle ausgeführt werden (z. B. durch gleichzeitig ausgeführte Anwendungen), abgeschlossen sind. Diese laufenden Zugriffe, die als *alte Suchoperationen* bezeichnet werden, verwenden die alten Satz-IDs (RIDs) für den Zugriff auf die Tabellendaten. Alle Zugriffe, die während dieser Wartezeit gestartet werden, werden als *neue Suchoperationen* bezeichnet und verwenden die neuen Satz-IDs für den Zugriff auf die Daten. Wenn alle alten Suchoperationen abgeschlossen sind, bereinigt die Tabellenreorganisation die versetzten Zeilen, indem sie die RP-Datensätze löscht und die RO-Datensätze in normale Datensätze umwandelt.
3. Füllen des Bereichs
   Nachdem alle Zeilen freigemacht wurden, werden die Zeilen in einem reorganisierten Format, nach den verwendeten Indizes sortiert und gemäß den definierten PCTFREE-Einschränkungen zurückgeschrieben. Wenn alle Seiten im Bereich gefüllt sind, werden die nächsten sequenziellen N Seiten in der Tabelle ausgewählt und der Prozess beginnt von vorn.
4. Abschneiden der Tabelle
   Wenn alle Seiten in der Tabelle reorganisiert wurden, wird die Tabelle standardmäßig abgeschnitten, um ungenutzten Speicherplatz wieder freizugeben. Wenn die Option NOTRUNCATE angegeben wird, wird die reorganisierte Tabelle nicht abgeschnitten.

Während einer Onlinetabellenreorganisation erstellte Dateien

Während einer Onlinetabellenreorganisation wird eine Statusdatei (`.OLR`) für jede Datenbankpartition erstellt. Bei dieser Datei handelt es sich um eine Binärdatei mit dem Namen `xxxxyyy.OLR`, wobei `xxxx` die Pool-ID und `yyyy` die Objekt-ID im Hexadezimalformat ist. Diese Datei enthält Informationen, die zur Fortsetzung einer Onlinereorganisation aus einem angehaltenen Status erforderlich sind.

Die Statusdatei enthält die folgenden Informationen:

- Typ der Reorganisation
- Aktive Protokollfolgenummer (LSN) der Tabelle, die gerade reorganisiert wird
- Nächster freizumachender Bereich
- Informationen dazu, ob die Reorganisation die Daten in Clustern ordnet oder nur Speicherplatz freigibt

- ID des Index für das Clustering der Daten

Für die .OLR-Datei wird eine Kontrollsumme gespeichert. Wenn die Datei beschädigt wird, sodass Kontrollsummenfehler auftreten, oder wenn die Protokollfolgenummer (LSN) der Tabelle nicht mit der aktiven Protokollfolgenummer übereinstimmt, muss eine neue Reorganisation eingeleitet werden, sodass eine neue Statusdatei erstellt wird.

Wenn die .OLR-Statusdatei gelöscht wird, kann der REORG TABLE-Prozess nicht fortgesetzt werden und ein Fehler SQL2219 wird zurückgemeldet. In diesem Fall muss ein neuer REORG TABLE-Prozess eingeleitet werden.

Die dem Reorganisationsprozess zugeordneten Dateien sollten nicht manuell aus dem System entfernt werden.

Ausführen von Onlinetabellenreorganisationen:

Eine online bzw. mit der Option INPLACE ausgeführte Tabellenreorganisation ermöglicht Benutzern den Zugriff auf die Tabelle, während diese reorganisiert wird.

Zur Reorganisation einer Tabelle benötigen Sie die Berechtigung SYSADM, SYSCTRL, SYSMAINT oder DBADM. Alternativ benötigen Sie das Zugriffsrecht CONTROL für die Tabelle. Sie müssen eine Datenbankverbindung zur Reorganisation einer Tabelle herstellen.

Sie können eine Onlinetabellenreorganisation mithilfe eines Befehls des Befehlszeilenprozessors (CLP), einer SQL-Anweisung CALL oder über eine DB2-Anwendungsprogrammierschnittstelle (API) ausführen.

1. Wenn Sie eine Tabelle online über den Befehlszeilenprozessor (CLP) reorganisieren wollen, führen Sie den folgenden Befehl mit der Option INPLACE aus:
   ```
   db2 reorg table test.employee inplace
   ```
2. Wenn Sie eine Tabelle online mithilfe einer SQL-Anweisung CALL reorganisieren wollen, führen Sie den Befehl REORG TABLE mithilfe der Prozedur ADMIN_CMD aus:
   ```
   call sysproc.admin_cmd ('reorg table employee inplace')
   ```
3. Zur Reorganisation einer Tabelle mit einer DB2-Administrator-API verwenden Sie die API 'db2REORG'.

Im Anschluss an die Reorganisation einer Tabelle sollten Sie Statistikdaten für die Tabelle erfassen, damit dem Optimierungsprogramm die aktuellsten und präzisesten Daten zur Auswertung von Abfragezugriffsplänen zur Verfügung stehen.

Recovery einer fehlgeschlagenen Onlinetabellenreorganisation:

Das Fehlschlagen einer Onlinetabellenreorganisation ist häufig auf Verarbeitungsfehler zurückzuführen, wie sie zum Beispiel durch einen vollen Datenträger oder durch Protokollierungsfehler verursacht werden. Wenn eine Onlinetabellenreorganisation fehlschlägt, wird eine Nachricht des SQL-Kommunikationsbereichs (SQLCA) in die Verlaufsdatei geschrieben.

Wenn eine oder mehrere Datenbankpartitionen in einer Umgebung mit partitionierten Datenbanken einen Fehler feststellen, wird der SQLCODE-Wert des ersten Knotens, der einen Fehler meldet, zurückgegeben.

Wenn der Fehler während der Laufzeit auftritt, wird die Onlinetabellenreorganisation angehalten und rückgängig gemacht. Wenn das System abstürzt, wird nach dem Neustart die Recovery nach einem Systemabsturz eingeleitet und die Reorganisation angehalten und rückgängig gemacht. Später können Sie die Reorganisation fortsetzen, indem Sie die Option RESUME im Befehl REORG TABLE angeben. Da der Prozess der Onlinetabellenreorganisation vollständig protokolliert wird, ist gewährleistet, dass die Reorganisation wiederherstellbar ist.

Unter bestimmten Umständen kann die Onlinetabellenreorganisation zum Beispiel den Grenzwert *num_log_span* überschreiten. In diesem Fall versetzt DB2 die REORG TABLE-Operation zwangsweise in den Status 'Angehalten' (PAUSE). In der Ausgabe der Momentaufnahme wird der Status des Dienstprogramms REORG TABLE mit 'PAUSED' angegeben.

Das Anhalten der Onlinetabellenreorganisation erfolgt interruptgesteuert. Dies bedeutet, dass die Reorganisation entweder durch den Benutzer (mit der Option PAUSE im Befehl REORG TABLE bzw. mit dem Anwendungsbefehl FORCE) oder durch DB2 unter bestimmten Umständen, zum Beispiel im Fall eines Systemabsturzes, angehalten werden kann.

Anhalten und erneutes Starten einer Onlinetabellenreorganisation:

Eine Onlinetabellenreorganisation, die gerade ausgeführt wird, kann durch den Benutzer angehalten und wieder gestartet werden.

Zum Anhalten bzw. zum erneuten Starten einer Onlinetabellenreorganisation müssen Sie über eine der folgenden Berechtigungen verfügen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL
- Berechtigung SYSMAINT
- Berechtigung DBADM
- Zugriffsrecht CONTROL für die Tabelle

1. Führen Sie zum Anhalten einer Onlinetabellenreorganisation den Befehl REORG TABLE mit der Option PAUSE aus:

   ```
   db2 reorg table homer.employee inplace pause
   ```

2. Führen Sie zum erneuten Starten einer angehaltenen Onlinetabellenreorganisation den Befehl REORG TABLE mit der Option RESUME aus:

   ```
   db2 reorg table homer.employee inplace resume
   ```

   Anmerkung:

   - Wenn eine Onlinereorganisation einer Tabelle angehalten wurde, können Sie keine neue Reorganisation dieser Tabelle starten. Sie müssen die angehaltene Reorganisation fortsetzen oder vollständig stoppen, bevor Sie einen neuen Reorganisationsprozess starten.
   - Wenn eine RESUME-Anforderung abgesetzt wird, berücksichtigt der Reorganisationsprozess die Option TRUNCATE, die im ursprünglichen REORG-Befehl angegeben war, unabhängig davon, welche TRUNCATE-Optionen mit nachfolgenden START-Anforderungen oder zwischendurch abgesetzten RESUME-Anforderungen angegeben werden. Wenn sich eine Reorganisation jedoch in der Abschneidephase (TRUNCATE) befindet und der Benutzer eine RESUME-Anforderungen mit der Option NOTRUNCATE absetzt, wird die Tabelle nicht abgeschnitten und die Reorganisation abgeschlossen.

- Eine Reorganisation kann nicht nach einer RESTORE- und ROLLFORWARD-Operation mit RESUME fortgesetzt werden.

Hinweise zu Sperren und zum gemeinsamem Zugriff bei Onlinetabellenreorganisationen:

Einer der wichtigsten Aspekte einer Onlinetabellenreorganisation ist die Art und Weise, wie das Sperren gesteuert wird, da dies für den gemeinsamen Anwendungszugriff von entscheidender Bedeutung ist.

Zu einem gegebenen Zeitpunkt im Verlauf einer Onlinetabellenreorganisation (REORG TABLE) kann die Operation die folgenden Sperren aktiviert haben:

- Zur Sicherstellung des Schreibzugriffs auf den Tabellenbereich wird eine IX-Sperre für die Tabellenbereiche angefordert, die von der REORG TABLE-Operation betroffen sind.
- Eine Tabellensperre wird angefordert und die gesamte REORG TABLE-Operation hindurch beibehalten. Die Sperrebene hängt vom Zugriffsmodus ab, der für diese Tabelle während der Reorganisation zugelassen wird:
  Wenn ALLOW WRITE ACCESS angegeben ist, wird eine IS-Sperre für die Tabelle aktiviert.
  Wenn ALLOW READ ACCESS angegeben ist, wird eine S-Sperre für die Tabelle aktiviert.
- In der Abschneidephase wird eine S-Sperre für die Tabelle angefordert, während die Reorganisation Zeilen aus dem Abschneidebereich herausholt und neue Zeilen durch alte Suchoperationen (Datenzugriffe, die während der REORG TABLE-Operation bereits ausgeführt werden und auf die alten Satz-IDs für die Datensätze zugreifen) eingefügt werden können. DB2 wartet, bis diese Sperre aktiviert ist, bevor das Abschneiden beginnt. Ganz am Ende der Abschneidephase, wenn die REORG TABLE-Operation das physische Abschneiden der Tabelle ausführt, wird die S-Sperre auf eine spezielle Z-Sperre hochgestuft. Dies bedeutet, dass die REORG TABLE-Operation erst abgeschlossen werden kann, wenn keine Tabellensperren von bereits aktiven Anwendungen mehr aktiviert sind, wozu auch IN-Sperren von einem UR-Suchvorgang zählen.
- Außerdem kann auch eine Zeilensperre abhängig vom Typ der Tabellensperre angefordert werden:
  Wenn eine S-Sperre für die Tabelle aktiviert ist, sind keine einzelnen S-Sperren auf Zeilenebene erforderlich, sodass keine weiteren Sperren benötigt werden.
  Wenn für eine Tabelle eine IS-Sperre aktiviert ist, wird eine S-Sperre auf Zeilenebene angefordert, bevor die Zeile versetzt wird, und wieder freigegeben, wenn die Versetzung ausgeführt ist.
- Interne Sperren werden möglicherweise zur Steuerung des Zugriffs auf das Objekt einer Onlinetabellenreorganisation und andere DB2-Dienstprogramme, wie zum Beispiel ein Online-Backup, benötigt.

Das Sperren hat Auswirkungen auf die Leistung sowohl für die REORG TABLE-Operation als auch für gleichzeitig ausgeführte Benutzeranwendungen. Es wird ausdrücklich empfohlen, Momentaufnahmedaten zu Sperren zu untersuchen, um die Sperraktivitäten während der Ausführung von Onlinetabellenreorganisationen zu verstehen.

Überwachen einer Tabellenreorganisation

Mithilfe des Befehls GET SNAPSHOT, der Verwaltungssicht SNAPTAB_REORG oder der Tabellenfunktion SNAP_GET_TAB_REORG können Sie Informationen über den Status Ihrer Operationen zur Tabellenreorganisation abrufen.

Sie müssen mit der Datenbank verbunden sein und die folgende Berechtigung besitzen:

- Berechtigung SYSMON
- Zugriffsrecht SELECT oder CONTROL für die Verwaltungssicht SNAPTAB_REORG oder Zugriffsrecht EXECUTE für die Tabellenfunktion SNAP_GET_TAB_REORG
- Für den Zugriff auf Informationen zu Reorganisationsoperationen mithilfe von SQL verwenden Sie die Verwaltungssicht SNAPTAB_REORG. Die folgende SELECT-Anweisung gibt zum Beispiel Details zu Tabellenreorganisationsoperationen für alle Datenbankpartitionen in der zurzeit verbundenen Datenbank zurück:

  ```
  SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSCHEMA, 1, 15)
     AS TAB_SCHEMA, REORG_PHASE, SUBSTR(REORG_TYPE, 1, 20) AS REORG_TYPE,
     REORG_STATUS, REORG_COMPLETION, DBPARTITIONNUM
     FROM SYSIBMADM.SNAPTAB_REORG ORDER BY DBPARTITIONNUM
  ```

  Wenn keine Tabellen reorganisiert wurden, werden 0 Zeilen zurückgegeben.
- Für den Zugriff auf Informationen zu Reorganisationsoperationen mithilfe von Snapshot Monitor setzen Sie den Befehl GET SNAPSHOT FOR TABLES ON *datenbank* ab und untersuchen die Werte der Monitorelemente für Tabellenreorganisationen (mit dem Präfix "reorg_").

Da die Offlinetabellenreorganisation (REORG TABLE) synchron erfolgt, werden alle Fehler in einer Offlinetabellenreorganisation an das aufrufende Programm des Dienstprogramms (d. h. entweder an die Anwendung oder an die Befehlszeile) zurückgemeldet.

Die Fehlerbehandlung in einer Onlinetabellenreorganisation unterscheidet sich geringfügig von der Fehlerbehandlung in einer Offlinetabellenreorganisation. Da eine Onlinetabellenreorganisation asynchron ausgeführt wird, werden keine SQL-Nachrichten an den Befehlszeilenprozessor (CLP) geschrieben. Zum Anzeigen von SQL-Fehlern, die durch eine Onlinetabellenreorganisation zurückgemeldet werden, geben Sie den Befehl LIST HISTORY REORG ein.

Eine Onlinetabellenreorganisation wird im Hintergrund als Prozess mit dem Namen `db2Reorg` ausgeführt. Dies heißt, dass eine erfolgreiche Rückkehr des REORG TABLE-Prozesses an die aufrufende Anwendung nicht bedeutet, dass die Reorganisation erfolgreich abgeschlossen wurde. Auch wenn die aufrufende Anwendung ihre Datenbankverbindung beendet, wird der Prozess `db2Reorg` fortgesetzt.

Indexreorganisation

Wenn Tabellen durch Löschungen und Einfügungen aktualisiert werden, verschlechtert sich die Indexleistung auf folgende Weisen:

- Aufteilung von Blattseiten

  Wenn Blattseiten aufgeteilt (fragmentiert) sind, steigen die E/A-Aufwände, weil mehr Blattseiten gelesen werden müssen, um Tabellenseiten abzurufen.
- Die physische Indexseitenreihenfolge stimmt nicht mehr mit der Reihenfolge der Schlüssel auf diesen Seiten überein. Dies wird als Index mit *schlechter Clusterbildung* bezeichnet.

  Wenn Blattseiten schlechte Clusterbildung aufweisen, ist ein sequenzieller Vorablesezugriff nicht effizient, was zu längeren E/A-Wartezeiten führt.
- Der Index bildet mehr als die maximal effiziente Anzahl von Stufen.

In einem solchen Fall sollte der Index reorganisiert werden.

Wenn Sie den Parameter MINPCTUSED bei der Erstellung eines Index definieren, führt der Datenbankserver automatisch Indexblattseiten zusammen, wenn ein Schlüssel gelöscht wird und der freie Speicherplatz weniger als der angegebene Prozentsatz ist. Dieser Vorgang wird als *Online-Indexdefragmentierung* bezeichnet. Um die Indexclusterbildung wiederherzustellen, Speicherplatz freizugeben und die Anzahl von Blattseitenstufen zu verringern, können Sie eine der folgenden Methoden anwenden:

- Löschen und erneutes Erstellen des Index
- Reorganisieren von Indizes online mithilfe des Befehls REORG INDEXES

  Sie können diese Methode in einer Produktionsumgebung wählen, da sie Benutzern ermöglicht, die Tabelle zu lesen und Daten in sie zu schreiben, während ihre Indizes neu erstellt werden.
- Verwenden des Befehls REORG TABLE mit Optionen, die eine Offlinereorganisation der Tabelle und ihrer Indizes ermöglichen

Online-Indexreorganisation

Wenn Sie den Befehl REORG INDEXES mit der Option ALLOW WRITE ACCESS verwenden, werden alle Indizes für die angegebene Tabelle erneut erstellt, während der Lese- und Schreibzugriff auf die Tabelle möglich ist. Alle Änderungen an der zugrunde liegenden Tabelle, die während des Reorganisationsprozesses einen Einfluss auf die Indizes haben würden, werden in den DB2-Protokollen aufgezeichnet. Außerdem werden dieselben Änderungen in den internen Speicherpufferbereich gestellt, sofern ein solcher Speicherbereich zur Verfügung steht. Bei der Reorganisation werden die protokollierten Änderungen während des erneuten Erstellens der Indizes zwecks Abgleichs mit der aktuellen Schreibaktivität verarbeitet. Der interne Speicherpufferbereich ist ein ausgewiesener Speicherbereich, der bei Bedarf vom Dienstprogrammzwischenspeicher zugeordnet wird, um die Änderungen an den erstellten bzw. reorganisierten Indizes zu speichern. Durch Verwendung dieses Speicherpufferbereichs können die Änderungen bei der Indexreorganisation verarbeitet werden, indem sie zunächst direkt aus dem Speicher gelesen werden. Gegebenenfalls werden anschließend auch die Protokolle gelesen, jedoch erst zu einem viel späteren Zeitpunkt. Der zugeordnete Speicher wird nach Beendigung der Reorganisation wieder freigegeben. Nach Beendigung der Reorganisation hat der erneut erstellte Index möglicherweise keine perfekte Clusterbildung. Wenn PCTFREE für einen Index angegeben wird, wird der angegebene Prozentsatz an Speicherplatz auf jeder Seite bei der Reorganisation freigehalten.

Für partitionierte Tabellen wird die Online-Indexreorganisation und die Bereinigung einzelner Indizes unterstützt. Zur Reorganisation einzelner Indizes geben Sie den Indexnamen an: `REORG INDEX indexname for TABLE tabellenname`

Die Online-Indexreorganisation im ALLOW WRITE-Modus wird für räumliche Indizes und MDC-Tabellen (MDC - Multi-Dimensional Clustering) nicht unterstützt.

Anmerkung: Die Option CLEANUP ONLY des Befehls REORG INDEXES/INDEX führt keine vollständige Reorganisation der Indizes durch. Die Option CLEANUP ONLY ALL entfernt die Schlüssel, die als gelöscht markiert sind und deren Löschung bekanntermaßen festgeschrieben ist. Sie gibt außerdem Seiten frei, auf denen alle Schlüssel als gelöscht markiert sind und diese Löschung bekanntermaßen festgeschrieben wurde. Bei der Freigabe von Seiten werden benachbarte Blattseiten zusammengefügt, wenn dadurch mindestens der Wert von PCTFREE an

freiem Speicherplatz auf der zusammengefügten Seite übrig behalten werden kann. Der Wert für PCTFREE ist der Prozentsatz an freiem Speicherbereich, der für den Index bei seiner Erstellung definiert wird. Die Option CLEANUP ONLY PAGES löscht nur Seiten, auf denen alle Schlüssel als gelöscht markiert sind und diese Löschung bekanntermaßen festgeschrieben ist.

Bei der Reorganisation von Indizes für partitionierte Tabellen mit der Option CLEANUP ONLY werden alle Zugriffsebenen unterstützt. Wenn die Option CLEANUP ONLY nicht angegeben wird, ist die Standardzugriffsebene ALLOW NO ACCESS die einzig unterstützte Zugriffsebene.

Für die Indexreorganisation gelten die folgenden Vorraussetzungen:

- Die Berechtigung SYSADM, SYSMAINT, SYSCTRL oder DBADM oder das Zugriffsrecht CONTROL für die Indizes und die Tabelle
- Eine Menge an freiem Speicherplatz in dem Tabellenbereich, in dem die Indizes gespeichert werden, die der aktuellen Größe des Index entspricht

  Ziehen Sie in Betracht, die Indizes, die einer Reorganisation unterliegen, in einem großen Tabellenbereich anzulegen, wenn Sie die Anweisung CREATE TABLE ausführen.
- Zusätzlicher Protokollspeicherbereich

  Die Indexreorganisation protokolliert ihre Aktivitäten. Infolgedessen kann die Reorganisation fehlschlagen, insbesondere wenn das System ausgelastet ist und andere gleichzeitige Aktivitäten protokolliert werden.

  Anmerkung: Wenn der Befehl REORG INDEXES ALL mit der Option ALLOW NO ACCESS fehlschlägt, werden die Indizes als ungültig markiert, und die Operation wird nicht rückgängig gemacht. Wenn jedoch ein REORG-Befehl mit der Option ALLOW READ ACCESS oder ein REORG-Befehl mit der Option ALLOW WRITE ACCESS fehlschlägt, wird das ursprüngliche Indexobjekt wiederhergestellt.

Ermitteln des Zeitpunkts für die Reorganisation von Tabellen und Indizes

Nach zahlreichen Änderungen an Tabellendaten können sich logisch sequenzielle Daten auf physisch nicht sequenziellen Datenseiten befinden, vor allem dann, wenn durch eine große Anzahl von Aktualisierungsoperationen Überlaufsätze entstanden sind. Wenn die Daten auf diese Weise organisiert sind, muss der Datenbankmanager zusätzliche Leseoperationen durchführen, um auf sequenzielle Daten zuzugreifen. Darüber hinaus sind zusätzliche Leseoperationen auch erforderlich, wenn eine beträchtliche Anzahl von Zeilen gelöscht wurde.

Die Tabellenreorganisation defragmentiert die Daten, wodurch die unnötige Belegung von Speicherplatz eliminiert wird, und ordnet die Zeilen neu, sodass Überlaufsätze berücksichtigt werden und somit der Datenzugriff und letztendlich die Abfrageleistung verbessert werden. Sie können auch angeben, dass die Daten anhand eines bestimmten Indexes neu geordnet werden sollen, sodass Abfragen mit einem Minimum an Leseoperationen auf die Daten zugreifen können.

Zahlreiche Änderungen an Tabellendaten haben Aktualisierungen an den Indizes zur Folge, und die Indexleistung kann abnehmen. Auf Indexblattseiten kann es zu Fragmentierung und schlechterem Clustering kommen, und der Index könnte mehr Stufen als für eine optimale Leistung erforderlich entwickeln. Alle diese Aspekte verursachen ein höheres E/A-Aufkommen und wirken sich negativ auf die Leistung aus.

Jeder der nachfolgend aufgeführten Faktoren kann darauf hinweisen, dass eine Tabelle oder ein Index reorganisiert werden sollte:

- Es ist ein hohes Aufkommen an Einfüge-, Aktualisierungs- und Löschaktivitäten für eine Tabelle angefallen, seit die Tabelle zuletzt reorganisiert wurde.
- Es treten wesentliche Änderungen in der Leistung von Abfragen auf, die einen Index mit einem hohen Clusterverhältnis verwenden.
- Die Leistung wird durch die Ausführung von RUNSTATS zur Aktualisierung der Statistikdaten nicht besser.
- Der Befehl REORGCHK gibt an, ob eine Tabelle oder ein Index reorganisiert werden muss. (Hinweis: In manchen Fällen empfiehlt REORGCHK stets eine Tabellenreorganisation, selbst nachdem eine Tabellenreorganisation ausgeführt wurde.) Beispiel: Wenn eine Seitengröße von 32 KB mit einer durchschnittlichen Satzlänge von 15 Byte und maximal 253 Sätzen pro Seite verwendet wird, bedeutet dies, dass für jede Seite 32700 - (15 x 253) = 28905 nicht verwendbare Byte anfallen. Etwa 88% der Seite besteht also aus freiem Speicherbereich. Die Benutzer sollten die Empfehlungen von REORGCHK analysieren und den Nutzen und Aufwand einer Reorganisation gegeneinander abwägen.
- Der Diagnoseanzeiger 'db.tb_reorg_req' (Reorganisation erforderlich) befindet sich im Status ATTENTION. In den Erfassungsdetails dieses Diagnoseanzeigers wird die Liste der Tabellen und Indizes beschrieben, die von einer Reorganisation profitieren könnten.

Der Befehl REORGCHK gibt Statistikdaten zur Datenorganisation aus und kann Empfehlungen dazu liefern, ob bestimmte Tabellen oder Indizes reorganisiert werden müssen. Allerdings kann die Ausführung spezieller Abfragen auf die Katalogstatistiktabellen in regelmäßigen Intervallen ein Leistungsdatenprotokoll liefern, das es Ihnen ermöglicht, Trends auszumachen, die eine größere Tragweite für die Leistung haben.

Fragen Sie die Katalogstatistiktabellen ab, und überwachen Sie die folgenden Statistikdaten, um zu bestimmen, ob Sie Tabellen oder Indizes reorganisieren müssen:

1. Überlauf von Zeilen

   Fragen Sie die Spalte OVERFLOW in der Sicht SYSSTAT.TABLES ab, um den OVERFLOW-Wert zu überwachen. Die Werte in dieser Spalte stellen die Anzahl von Zeilen dar, die nicht auf ihre ursprünglichen Seiten passen. Zeilendaten können überlaufen, wenn Spalten mit variabler Länge in der Tabelle dazu führen, dass die Datensatzlänge in einer Weise zunimmt, dass die Daten nicht mehr an ihre ursprünglich zugewiesene Position auf der Datenseite passt. Längenänderungen können auch auftreten, wenn der Tabellendefinition eine Spalte hinzugefügt wird und später durch Aktualisieren der Zeile Daten gespeichert werden. In diesen Fällen wird an der ursprünglichen Stelle der Zeile ein Zeiger hinterlegt, und der tatsächliche Wert an einer anderen, von dem Zeiger angegebenen Stelle gespeichert. Dadurch kann die Leistung beeinträchtigt werden, da der Datenbankmanager dem Zeiger folgen muss, um den Inhalt der Zeile zu finden. Durch diese beiden Schritte verlängert sich die Verarbeitungszeit und erhöht sich möglicherweise auch die Anzahl der erforderlichen E/A-Operationen.

   Eine Reorganisation der Tabellendaten beseitigt die Zeilenüberläufe. Daher erhöht sich mit steigender Anzahl von Überlaufzeilen auch der potenzielle Nutzen einer Reorganisation der Tabellendaten.

2. Abrufstatistiken (Fetch)

   Fragen Sie die drei folgenden Spalten in den Katalogstatistiktabellen SYSCAT.INDEXES und SYSSTAT.INDEXES ab, um die Effektivität der Vorablesefunktionen beim Zugriff auf die Tabellen in der Indexreihenfolge zu ermitteln. Diese Statistiken vermitteln einen Eindruck von der durchschnittlichen Leistung der Vorablesefunktionen für die zugrunde liegende Tabelle.

   - In der Spalte AVERAGE_SEQUENCE_FETCH_PAGES wird die durchschnittliche Anzahl von Seiten gespeichert, auf die in der Reihenfolge in der Tabelle zugegriffen werden kann. Die Seiten, auf die in der Reihenfolge zugegriffen werden kann, kommen für den Vorablesezugriff in Betracht. Ein kleiner Wert gibt an, dass die Vorablesefunktionen nicht so effizient sind, wie sie sein könnten, weil sie nicht die gesamte Anzahl von Seiten, die durch die Einstellung PREFETCHSIZE für den Tabellenbereich definiert sind, einlesen können. Ein hoher Wert zeigt an, dass die Vorablesefunktionen effektiv arbeiten. Für eine Tabelle mit Clusterindex sollte dieser Wert dem Wert für NPAGES nahe kommen, der die Anzahl von Seiten angibt, die Zeilen enthalten.
   - In der Spalte AVERAGE_RANDOM_FETCH_PAGES wird die durchschnittliche Anzahl von Tabellenseiten gespeichert, auf die beim Abrufen von Tabellenzeilen über den Index zwischen sequenziellen Seitenzugriffen ein wahlfreier Zugriff erfolgt. Die Vorablesefunktionen ignorieren kleine Anzahlen wahlfreier Seiten, wenn die Mehrzahl der Seiten in der Reihenfolge vorliegt, und setzen den Vorablesezugriff bis zur konfigurierten Vorablesegröße fort. Mit zunehmender Unordnung in der Reihenfolge der Tabelle steigt die Anzahl von Seiten, auf die ein wahlfreier Zugriff erfolgen muss. Eine solche zunehmende Unordnung wird in der Regel durch Einfügungen außerhalb der Reihenfolge, das heißt, entweder am Ende der Tabelle oder in Überlaufseiten, verursacht. Dies führt zu Abrufen, die die Abfrageleistung verlangsamen, wenn der Index für den Zugriff auf einen Bereich von Werten verwendet wird.
   - In der Spalte AVERAGE_SEQUENCE_FETCH_GAP wird die durchschnittliche Lücke zwischen Tabellenseiten in der Reihenfolge beim Abrufen über den Index gespeichert. Erkannt beim Durchsuchen von Indexseiten stellt jede Lücke die durchschnittliche Anzahl von Seiten dar, die jeweils zwischen Seiten in der Reihenfolge wahlfrei abgerufen werden müssen. Solche Lücken treten auf, wenn auf viele Seiten wahlfrei zugegriffen wird, wodurch die Vorablesefunktionen angehalten werden. Ein hoher Wert zeigt an, dass eine Tabelle nicht gut organisiert oder bezüglich des Index eine niedrige Clusterbildung aufweist.

3. Anzahl von Indexblattseiten, die als gelöscht markierte, jedoch noch nicht gelöschte Satz-IDs (RIDs) enthalten

   In Indizes des Typs 2 werden Satz-IDs in der Regel nicht physisch gelöscht, wenn die Satz-ID als gelöscht markiert wird. Dies bedeutet, dass nützlicher Speicherplatz von diesen logisch gelöschten Satz-IDs belegt sein könnte. Zum Abrufen der Anzahl von Blattseiten, auf denen jede Satz-ID als gelöscht markiert ist, fragen Sie die Spalte NUM_EMPTY_LEAFS der Statistiktabellen SYSCAT.INDEXES und SYSSTAT.INDEXES ab. Für Blattseiten, in denen nicht alle Satz-IDs als gelöscht markiert sind, wird die Gesamtzahl logisch gelöschter Satz-IDs in der Spalte NUMRIDS_DELETED gespeichert.

   Verwenden Sie diese Informationen zur Abschätzung, wie viel Speicherplatz durch eine Ausführung des Befehls REORG INDEXES mit der Option CLEANUP ALL zurückgewonnen werden könnte. Um nur den Speicherplatz in Seiten wieder verfügbar zu machen, in denen alle Satz-IDs als gelöscht markiert sind, führen Sie den Befehl REORG INDEXES mit der Option CLEANUP ONLY PAGES aus.

4. Statistiken zum Clusterverhältnis und zum Clusterfaktor für Indizes

   In der Spalte CLUSTERRATIO der Katalogtabelle SYSTCAT.INDEXES wird ein Statistikwert für das Clusterverhältnis gespeichert. Dieser Wert (zwischen 0 und 100) stellt den Grad der Datenclusterbildung bezüglich des Index dar. Wenn Sie detaillierte Indexstatistikdaten (DETAILED) erfassen, wird ein feinerer Statistikwert zwischen 0 und 1 für den Clusterfaktor in der Spalte CLUSTERFACTOR gespeichert und der Wert der Spalte CLUSTERRATIO ist -1. Nur einer dieser beiden Werte zur Clusterbildung wird im Katalog SYSCAT.INDEXES aufgezeichnet. Zum Vergleich der CLUSTERFACTOR-Werte mit den CLUSTERRATIO-Werten müssen Sie den CLUSTERFACTOR-Wert mit 100 multiplizieren, um einen Prozentwert zu erhalten.

   Anmerkung: Im Allgemeinen kann nur einer der Indizes in einer Tabelle einen hohen Grad an Clusterbildung aufweisen.

   Indexsuchen, die nicht mit einem reinen Indexzugriff durchgeführt werden, erzielen bei höheren Clusterverhältnissen wahrscheinlich eine bessere Leistung. Ein niedriges Clusterverhältnis führt zu vermehrten Ein-/Ausgabeoperationen für diese Art der Suche, da nach dem ersten Zugriff auf eine Datenseite die Wahrscheinlichkeit geringer ist, dass sich diese Seite immer noch im Pufferpool befindet, wenn der nächste Zugriff auf sie erfolgt. Die Leistung für einen Index ohne Clusterbildung kann möglicherweise durch Erhöhen der Puffergröße verbessert werden.

   Wenn Tabellendaten anfangs bezüglich eines bestimmten Index Clusterbildung aufwiesen und die Statistikdaten zur Clusterbildung nun anzeigen, dass in Bezug auf denselben Index nur eine geringe Clusterbildung vorhanden ist, kann es sinnvoll sein, die Tabelle zu reorganisieren, um die Daten bezüglich dieses Index wieder in Clustern anzuordnen.

5. Anzahl der Blattseiten (Leaf pages)

   Fragen Sie die Spalte NLEAF in der Tabelle SYSCAT.INDEXES ab, um die Anzahl von Blattseiten zu ermitteln, die von einem Index belegt werden. Diese Anzahl gibt Auskunft darüber, wie viele E/A-Operationen für Indexseiten für ein komplettes Durchsuchen eines Index erforderlich sind.

   Ein Index sollte idealerweise möglichst wenig Speicherplatz belegen, um die E/A-Operationen für eine Indexsuche zu reduzieren. Wahlfreie Aktualisierungen können dazu führen, dass Seiten geteilt werden und sich der Index dadurch vergrößert. Wenn Indizes während der Reorganisation einer Tabelle neu erstellt werden, kann jeder Index mit dem minimal erforderlichen Speicherbereich erstellt werden.

   Anmerkung: Bei der Erstellung von Indizes werden standardmäßig zehn Prozent des freien Speicherbereichs auf jeder Indexseite nicht belegt. Zur Erhöhung des Betrages an freiem Speicherbereich geben Sie den Parameter PCTFREE bei der Erstellung des Index an. Der Wert für PCTFREE wird bei jeder Reorganisation des Index verwendet. Ein freier Speicherbereich von mehr als zehn Prozent kann die Häufigkeit der Indexreorganisation verringern, weil in dem zusätzlichen Speicherbereich weitere Indexeinfügungen untergebracht werden können.

6. Anzahl der leeren Datenseiten

   Zur Berechnung der Anzahl leerer Seiten in einer Tabelle fragen Sie die Spalten FPAGES und NPAGES der Tabelle SYSCAT.TABLES ab und subtrahieren den NPAGES-Wert vom FPAGES-Wert. In der Spalte FPAGES wird die Gesamtanzahl der verwendeten Seiten, in der Spalte NPAGES die Anzahl von Seiten gespeichert, die Zeilen enthalten. Leere Seiten können auftreten, wenn ganze Bereiche von Zeilen gelöscht werden.

Mit dem Ansteigen der Anzahl leerer Seiten wächst die Notwendigkeit einer Reorganisation für eine Tabelle. Bei der Reorganisation einer Tabelle werden die leeren Seiten aus dem Speicherbereich für eine Tabelle entfernt und der von der Tabelle belegte Speicherplatz verringert. Da leere Seiten bei einer Tabellensuche auch in den Pufferpool gelesen werden, kann durch das Entfernen ungenutzter Seiten die Leistung einer Tabellensuche verbessert werden.

Wenn die Gesamtanzahl Seiten (FPAGES) in einer Tabelle keiner oder gleich NPARTITIONS *1 EXTENTSIZE ist, wird keine Tabellenreorganisation empfohlen. NPARTITIONS ist die Anzahl von Datenpartitionen, wenn es sich um eine partitionierte Tabelle handelt. Ansonsten hat NPARTITIONS den Wert 1. Wird in einer Umgebung mit partitionierten Datenbanken die Anzahl Datenbankpartitionen in einer Datenbankpartitionsgruppe der Tabelle mit eingerechnet, ändert sich die Bedingung in FPAGES <= Anzahl der Datenbankpartitionen in einer Datenbankpartitionsgruppe einer Tabelle * NPARTITIONS * 1 EXTENTSIZE.

Bedenken Sie vor der Reorganisation Folgendes: Die Kosten einer zunehmenden Verschlechterung der Abfrageleistung und die Kosten einer Reorganisation der Tabelle bzw. der Indizes, zu denen die CPU-Zeit, die zur Ausführung erforderliche Zeit sowie der verringerte Grad des gemeinsamen Zugriffs aufgrund der Sperre, die das Dienstprogramm REORG für die Tabelle bis zum Ende der Reorganisation aktiviert, zu rechnen sind, müssen gegeneinander abgewogen werden.

Reorganisationsaufwand für Tabellen und Indizes

Bei der Ausführung einer Reorganisation einer Tabelle oder eines Index entsteht ein gewisser Systemaufwand, der bei der Entscheidung, ob ein Objekt reorganisiert werden soll, zu berücksichtigen ist.

Im Hinblick auf den Aufwand, der bei der Reorganisation von Tabellen und Indizes entsteht, sind folgende Aspekte zu beachten:

- CPU-Belegung des ausführenden Dienstprogramms.
- Beschränkung des gemeinsamen Zugriffs während der Ausführung des Dienstprogramms REORG. Der gemeinsame Zugriff wird durch die Sperranforderungen der REORG-Operation beschränkt.
- Zusätzlicher Speicherbedarf. (Die Offlinetabellenreorganisation erfordert zusätzlichen Speicherplatz zur Aufnahme einer Spiegelkopie der Tabelle. Für online bzw. mit der Option INPLACE ausgeführte Tabellenreorganisationen ist zusätzlicher Speicherplatz für die Protokollierung erforderlich. Die Online-Indexreorganisation erfordert zusätzlichen Speicherplatz zur Aufnahme einer Spiegelkopie des Index bzw. der Indizes sowie zusätzlichen Speicherplatz für die Protokollierung. Die Offline-Indexreorganisation benötigt weniger Protokollspeicherbereich und arbeitet ohne Spiegelkopie.)

In einigen Fällen kann eine reorganisierte Tabelle größer als die ursprüngliche Tabelle sein, sodass der Speicherplatzbedarf entsprechend steigt. Eine Tabelle kann in folgenden Situationen nach der Reorganisation größer sein:

- Wenn bei einer REORG TABLE-Operation mit Clustering, bei der ein Index zur Festlegung der Reihenfolge der Zeilen verwendet wird, die Tabellendatensätze eine variable Länge haben (z. B. durch die Verwendung von VARCHAR-Datentypen), kann mehr Speicherplatz belegt werden, da einige Seiten vielleicht weniger Zeilen als in der ursprünglichen Tabelle enthalten.
- Wenn der Tabelle Spalten vor der Reorganisation, jedoch nach der Erstellung der Tabelle, hinzugefügt wurden, treten die zusätzlichen Spalten in einigen Zeilen nach der Reorganisation vielleicht zum ersten Mal in Erscheinung.

- Wenn der Wert für die Größe des auf der Seite belassenen freien Speicherbereichs (Attribut PCTFREE) seit der letzten Reorganisation erhöht wurde.
- Wenn die Tabelle ein großes Objekt (LOB) enthält, da dies die Möglichkeit mit sich bringt, dass die LOB-Daten mehr Speicherplatz als zuvor belegen.

Speicherplatzbedarf für eine Offlinetabellenreorganisation

Da bei der Offlinereorganisation ein Spiegelkopieverfahren verwendet wird, müssen Sie über ausreichend zusätzlichen Speicherplatz verfügen, um eine weitere Kopie der Tabelle unterbringen zu können. Die Spiegelkopie wird entweder in dem Tabellenbereich, in dem sich die Originaltabelle befindet, oder in einem vom Benutzer angegebenen Tabellenbereich für temporäre Tabellen erstellt.

Zusätzlicher Speicherplatz in einem Tabellenbereich für temporäre Tabellen kann für die Verarbeitung von Sortierungen erforderlich werden, wenn eine Tabellensuchsortierung ausgeführt wird. Der erforderliche zusätzliche Speicherplatz kann bis zur Größe der Tabelle betragen, die reorganisiert wird. Wenn der Clusterindex ein SMS-Typ oder ein eindeutiger DMS-Typ ist, erfordert die erneute Erstellung dieses Index keine Sortierung. Stattdessen wird dieser Index durch ein Durchsuchen der neu reorganisierten Daten erstellt. Alle anderen Indizes, für die eine Neuerstellung notwendig ist, erfordern eine Sortierung, bei der potenziell ein Speicherplatzbedarf im Tabellenbereich für temporäre Tabellen anfällt, der bis zur Größe der Tabelle, die reorganisiert wird, betragen kann.

Eine offline ausgeführte Tabellenreorganisation generiert relativ wenige Steuerprotokolleinträge und nimmt daher relativ wenig Protokollspeicherplatz in Anspruch. Wenn die Reorganisation keinen Index verwendet, sind die einzigen Protokollsätze, die aufgezeichnet werden, die Protokollsätze für Tabellendaten. Wenn ein Index angegeben wird oder wenn ein Clusterindex für die Tabelle vorhanden ist, werden die Satz-IDs (RIDs) der Zeilen in der Reihenfolge protokolliert, in der sie in die neue Version der Tabelle eingefügt werden. Jeder Protokollsatz für die Satz-IDs enthält maximal 8000 Satz-IDs, wobei jede Satz-ID 4 Byte belegt. Dies kann einer der Faktoren sein, die zu einem Mangel an Protokollspeicher während einer Offlinetabellenreorganisation beitragen. Beachten Sie, dass Satz-IDs nur protokolliert werden, wenn die Datenbank wiederherstellbar (`LOGRETAIN=ON`) ist.

Protokollspeicherbedarf für eine Onlinetabellenreorganisation

Der für eine Onlinetabellenreorganisation erforderliche Protokollspeicherbedarf ist in der Regel höher als der für eine Offlinetabellenreorganisation. Die Größe des erforderlichen Speicherplatzes wird durch die Anzahl der Zeilen, die reorganisiert werden, die Anzahl von Indizes, die Größe der Indexschlüssel sowie durch den anfangs bestehenden Grad an Fragmentierung der Tabelle bestimmt. Es empfiehlt sich daher, einen typischen Vergleichspunkt (Benchmark) für die Protokollspeichernutzung für Ihre Tabelle zu ermitteln.

Jede Zeile in der Tabelle wird aller Wahrscheinlichkeit nach zweimal während einer Onlinetabellenreorganisation versetzt. Wenn ein Index vorhanden ist, muss der Indexschlüssel für jede Zeile zunächst mit der neuen Position aktualisiert werden und nach Abschluss aller Zugriffe auf die alte Position erneut aktualisiert werden, um den Verweis auf die alte Satz-ID zu entfernen. Wenn die Zeile zurückversetzt wird, werden diese Aktualisierungen analog erneut ausgeführt. Alle diese Aktivitäten werden protokolliert, um die Onlinetabellenreorganisation vollständig wiederherstellbar zu machen. Daher werden mindestens zwei Datenprotokollsätze (jede Instanz mit den Zeilendaten) und vier Indexprotokollsätze (jede Instanz mit den Schlüsseldaten) aufgezeichnet. Der Clusterindex neigt besonders dazu, die

Indexseiten zu füllen, sodass Teilungen und Zusammenführungen des Index auftreten, die ebenfalls protokolliert werden müssen.

Da eine Onlinetabellenreorganisation häufig interne COMMIT-Befehle ausführt, sind die aktiven Protokolle in der Regel nicht besonders zahlreich. Wenn es überhaupt einen Zeitraum gibt, in dem eine Onlinereorganisation eine große Anzahl aktiver Protokolle hat, dann während der Abschneidephase (TRUNCATE), während deren eine S-Tabellensperre aktiviert wird. Wenn die Tabellenreorganisation die Sperre nicht aktivieren kann, wartet sie ab und behält das Protokoll bei, sodass andere Transaktionen die Protokolle eventuell rasch füllen.

Senken des Bedarfs an Tabellen- und Indexreorganisationen

Sie können verschiedene Strategien anwenden, um die Häufigkeit zu verringern, mit der Sie Ihre Tabellen und Indizes reorganisieren müssen. Auf diese Weise lassen sich die Kosten einer unnötigen Ausführung von Reorganisationsoperationen vermeiden.

Senken des Bedarfs an Tabellenreorganisationen

- Verwenden Sie Mehrpartitionstabellen. Je kleiner die Tabelle ist, desto geringer ist die Wahrscheinlichkeit, dass sie reorganisiert werden muss.
- Erstellen Sie Tabellen mit mehrdimensionalem Clustering (MDC-Tabellen), bei denen automatisch für das Clustering der Spalten gesorgt wird, die in der Klausel ORGANIZE BY DIMENSION der Anweisung CREATE TABLE angegeben werden.
- Aktivieren Sie den Anfügemodus (APPEND) für Tabellen. Wenn die Indexschlüsselwerte neuer Zeilen zum Beispiel stets neue HIGHKEY-Werte (d. h. neue Höchstwerte) sind, versucht das Clustering-Attribut der Tabelle, diese Zeilen an das Ende der Tabelle zu setzen. In diesem Fall kann das Versetzen der Tabelle in den Anfügemodus eine bessere Wahl als ein Clusterindex sein.
- Gehen Sie nach Erstellung einer Tabelle wie folgt vor:
  - Ändern Sie die Tabelle (ALTER), um PCTFREE hinzuzufügen.
  - Erstellen Sie einen Clusterindex mit angegebenem PCTFREE für den Index.
  - Sortieren Sie die Daten, bevor Sie sie in die Tabelle laden.

  Ein Clustering mit einer geeigneten PCTFREE-Einstellung für eine Tabelle hilft bei der Beibehaltung der ursprünglichen, sortierten Reihenfolge. Wenn genügend Speicherplatz auf den Tabellenseiten freigehalten wird, können neue Daten auf den richtigen Seiten eingefügt werden, um ihr Clustering in Bezug auf den Index beizubehalten. Mit wachsender Menge eingefügter Daten und zunehmender Füllung der Seiten der Tabelle werden Datensätze an das Ende der Tabelle angehängt, sodass die Tabelle allmählich ihr Clustering einbüßt.

  Wenn Sie das Dienstprogramm REORG TABLE ausführen oder eine Sortierung und ein Laden von Daten nach der Erstellung eines Clusterindex durchführen, versucht der Index eine bestimmte Ordnung der Daten zu erhalten, wodurch sich die Werte für die Statistiken CLUSTERRATIO bzw. CLUSTERFACTOR verbessern, die durch das Dienstprogramm RUNSTATS erfasst werden.

Senken des Bedarfs an Indexreorganisationen

- Erstellen Sie Clusterindizes mit Einstellungen für PCTFREE bzw. LEVEL2 PCTFREE für Indexseiten. Der mögliche Bereich reicht von 0 bis 99 %, wobei der Standardwert 10 % ist.
- Erstellen Sie Indizes mit angegebenem MINPCTUSED. Der mögliche Bereich reicht von 0 bis 99 %, wobei der empfohlene Wert 50 % ist. Ziehen Sie dennoch die Verwendung der Option CLEANUP ONLY ALL des Befehls REORG INDEXES in Betracht, um stattdessen Blattseiten (engl. leaf pages) zusammenzufügen.

Automatische Reorganisation

Nach zahlreichen Änderungen an Tabellendaten können die Tabelle und die zugehörigen Indizes fragmentiert werden. Logisch sequenzielle Daten können sich auf nicht sequenziellen physischen Seiten befinden, sodass der Datenbankmanager für den Datenzugriff zusätzliche Leseoperationen ausführen muss.

Die vom Dienstprogramm RUNSTATS gesammelten Statistikinformationen zeigen unter anderem die Datenverteilung innerhalb einer Tabelle. Insbesondere kann durch eine Analyse dieser Statistiken ermittelt werden, wann eine Reorganisation und welche Art der Reorganisation erforderlich ist. Die automatische Reorganisation bestimmt die Erforderlichkeit einer Reorganisation für Tabellen und Indizes mithilfe der REORGCHK-Formeln. Sie bewertet in regelmäßigen Abständen Tabellen und Indizes, deren Statistiken aktualisiert wurden, um zu prüfen, ob eine Reorganisation erforderlich ist. Wenn dies der Fall ist, terminiert sie intern eine Indexreorganisation oder eine klassische Tabellenreorganisation für die Tabelle. Dies setzt voraus, dass Ihre Anwendungen ohne Schreibzugriff auf die Tabellen arbeiten, die reorganisiert werden.

Die Funktion zur automatischen Reorganisation kann mithilfe der Datenbankkonfigurationsparameter AUTO_REORG, AUTO_TBL_MAINT und AUTO_MAINT aktiviert bzw. inaktiviert werden.

In einer Umgebung mit partitionierten Datenbanken erfolgt die Festlegung, eine automatische Reorganisation auszuführen, und die Einleitung der automatischen Reorganisation in der Katalogtabellenpartition. Die Datenbankkonfigurationsparameter müssen lediglich in der Katalogtabellenpartition aktiviert werden. Die Reorganisation wird in allen Datenbankpartitionen ausgeführt, in denen sich die Zieltabellen befinden.

Wenn Sie sich nicht sicher sind, wann und wie Ihre Tabellen und Indizes zu reorganisieren sind, können Sie die automatische Reorganisation in Ihren Gesamtplan zur Datenbankverwaltung aufnehmen.

Tabellen und Indizes, die für die automatische Reorganisation in Betracht kommen, können über den Assistenten zur automatischen Verwaltung in der Steuerzentrale oder in der Diagnosezentrale konfiguriert werden.

Aktivieren der automatischen Reorganisation von Tabellen und Indizes

Über gut organisierte Tabellen- und Indexdaten zu verfügen, ist von kritischer Bedeutung für einen effizienten Datenzugriff und eine optimale Auslastungsleistung. Nach zahlreichen Änderungen an Tabellendaten können sich logisch sequenzielle Daten auf physisch nicht sequenziellen Datenseiten befinden, sodass der Datenbankmanager zusätzliche Leseoperationen ausführen muss, um auf die Daten zuzugreifen. Darüber hinaus sind zusätzliche Leseoperationen auch erforderlich, wenn eine beträchtliche Anzahl von Zeilen gelöscht wurde. Verwenden Sie die automatische Reorganisation von Tabellen, um DB2 zu veranlassen, die Offline-Reorganisation von Tabellen und Indizes zu verwalten, sodass Sie sich nicht darum zu kümmern brauchen, wann und wie die Daten reorganisiert werden. Sie können DB2 sowohl Systemkatalogtabellen als auch Datenbanktabellen reorganisieren lassen.

Sie können diese Funktion entweder über die Tools der grafischen Benutzerschnittstelle oder über die Befehlszeilenschnittstelle aktivieren.

- Gehen Sie wie folgt vor, um Ihre Datenbank für die automatische Reorganisation über die Tools der grafischen Benutzerschnittstelle zu konfigurieren:
  1. Öffnen Sie den Assistenten **Automatische Verwaltung konfigurieren** entweder über die Steuerzentrale, indem Sie ein Datenbankobjekt mit der rechten Maustaste anklicken, oder über die Diagnosezentrale, indem Sie die Datenbankinstanz, die Sie zur automatischen Reorganisation konfigurieren wollen, mit der rechten Maustaste anklicken. Wählen Sie **Automatische Verwaltung konfigurieren** im Kontextmenü aus.
  2. In diesem Assistenten können Sie die automatische Reorganisation zur Defragmentierung von Daten aktivieren, die Tabellen angeben, die automatisch reorganisiert werden sollen und ein Verwaltungsfenster zur Ausführung des REORG-Dienstprogramms angeben.
- Zur Konfiguration der automatischen Reorganisation Ihrer Datenbank über die Befehlszeilenschnittstelle setzen Sie die folgenden Konfigurationsparameter auf den Wert ON:
  – AUTO_MAINT
  – AUTO_TBL_MAINT
  – AUTO_REORG

Verwenden von relationalen Indizes zur Leistungsverbesserung

Indizes können zur Leistungsverbesserung beim Zugreifen auf Tabellendaten verwendet werden. Relationale Indizes werden zum Arbeiten mit relationale Daten verwendet. Beim Zugriff auf XML-Daten werden Indizes für XML-Daten verwendet.

Obwohl das Optimierungsprogramm entscheidet, ob ein relationaler Index für den Zugriff auf Daten relationaler Tabellen verwendet werden soll, müssen Sie, abgesehen von den folgenden Ausnahmen, ermitteln, welche Indizes die Leistung verbessern könnten, und diese Indizes erstellen. Ausnahmen sind die Dimensionsblockindizes und die zusammengesetzten Blockindizes, die für jede Dimension automatisch erstellt werden, die Sie bei der Erstellung einer MDC-Tabelle (MDC - Multi-Dimensional Clustering) angeben.

Sie müssen außerdem in folgenden Situationen das Dienstprogramm RUNSTATS ausführen, um neue Statistikdaten zu den relationalen Indizes zu erfassen:

- Nachdem Sie einen relationalen Index erstellt haben.
- Nachdem Sie den Wert für PREFETCHSIZE geändert haben.

Darüber hinaus sollten Sie das Dienstprogramm RUNSTATS in regelmäßigen Abständen ausführen, um die Statistikdaten auf aktuellem Stand zu halten. Ohne aktuelle Statistiken zu Indizes ist das Optimierungsprogramm nicht in der Lage, den besten Datenzugriffsplan für Abfragen zu ermitteln.

Anmerkung: Um festzustellen, ob ein relationaler Index in einem bestimmten Paket verwendet wird, verwenden Sie die EXPLAIN-Einrichtung. Zum Planen von relationalen Indizes können Sie mit dem Designadvisor in der Steuerzentrale oder mit dem Tool `db2advis` Hinweise zu relationalen Indizes abrufen, die von einer oder mehreren SQL-Anweisungen verwendet werden können.

Vorteile eines relationalen Index gegenüber keinem Index

Wenn für eine Tabelle kein Index vorhanden ist, muss eine Tabellensuche für jede Tabelle durchgeführt werden, auf die in einer SQL-Abfrage verwiesen wird. Je

umfangreicher die Tabelle ist, desto länger dauert die Tabellensuche, weil eine Tabellensuche erfordert, dass auf jede Tabellenzeile sequenziell zugegriffen wird. Obwohl eine Tabellensuche für eine komplexe Abfrage, die die meisten der Zeilen in einer Tabelle abruft, effizienter sein kann, ist für eine Abfrage, die nur einige Tabellenzeilen zurückgibt, eine Indexsuche für den Zugriff auf Tabellenzeilen effizienter.

Das Optimierungsprogramm wählt eine Indexsuche aus, wenn in der SELECT-Anweisung auf die relationalen Indexspalten verwiesen wird und wenn aufgrund der Schätzungen zu erwarten ist, dass eine Indexsuche schneller als eine Tabellensuche durchgeführt werden kann. Indexdateien sind im Allgemeinen kleiner und erfordern weniger Zeit zum Lesen als eine ganze Tabelle, besonders wenn die Tabellen umfangreicher werden. Darüber hinaus braucht eventuell nicht der gesamte Index durchsucht zu werden. Die Vergleichselemente, die auf einen Index angewandt werden, verringern die Anzahl der Zeilen, die aus den Datenseiten gelesen werden müssen.

Wenn eine angeforderte Sortierreihenfolge der Ausgabe einer Indexspalte entspricht, können die Zeilen durch eine Suche in diesem Index in der Spaltenreihenfolge gleich in der richtigen Reihenfolge ohne Sortierung abgerufen werden.

Jeder relationale Indexeintrag enthält einen Suchschlüsselwert und einen Zeiger auf die Zeile, die den entsprechenden Wert enthält. Wenn Sie den Parameter ALLOW REVERSE SCANS in der Anweisung CREATE INDEX angeben, können die Werte sowohl in aufsteigender als auch in absteigender Reihenfolge gesucht werden. Es ist daher möglich, die Suche unter Angabe geeigneter Vergleichselemente zu begrenzen. Ein relationaler Index kann auch dazu verwendet werden, Zeilen in einer geordneten Reihenfolge abzurufen und somit zu vermeiden, dass der Datenbankmanager die Zeilen nach dem Lesen aus der Tabelle in einem weiteren Arbeitsgang sortieren muss.

Neben dem Suchschlüsselwert und dem Zeilenzeiger kann ein relationaler Index auch INCLUDE-Spalten enthalten, bei denen es sich um nicht indexierte Spalten in der indexierten Zeile handelt. Solche Spalten machen es dem Optimierungsprogramm möglich, die angeforderten Informationen aus dem Index abzurufen, ohne auf die Tabelle selbst zugreifen zu müssen.

Anmerkung: Das Vorhandensein eines relationalen Index für die Tabelle, die abgefragt wird, garantiert jedoch keine sortierte Ergebnismenge. Nur durch die Angabe der Klausel ORDER BY lässt sich die Reihenfolge der Ergebnismenge sicherstellen.

Obwohl Indizes die Zugriffszeiten erheblich verringern können, können sie auch nachteilige Auswirkungen auf die Leistung haben. Vor der Erstellung von Indizes sind daher die Auswirkungen mehrerer Indizes auf den Plattenspeicherplatz und die Verarbeitungszeit zu bedenken:

- Jeder Index erfordert Speicher oder Plattenspeicherplatz. Die exakte Menge ist von der Größe der Tabelle und der Größe und Anzahl von Spalten in dem relationalen Index abhängig.
- Jede für eine Tabelle ausgeführte Operation INSERT oder DELETE erfordert eine zusätzliche Aktualisierung aller Indizes für diese Tabelle. Dies gilt auch für jede Operation UPDATE, mit der der Wert eines Indexschlüssels geändert wird.
- Das Dienstprogramm LOAD erstellt alle vorhandenen relationalen Indizes neu bzw. hängt Daten an sie an.

Der Parameter `indexfreespace` MODIFIED BY kann für den Befehl LOAD angegeben werden, um den Wert für PCTREE des Index außer Kraft zu setzen, der bei der Erstellung des relationalen Index verwendet wurde.

- Jeder relationale Index fügt potenziell einen alternativen Zugriffspfad für eine SQL-Abfrage hinzu, den das Optimierungsprogramm berücksichtigen soll. Dadurch verlängert sich die Kompilierungszeit.

Wählen Sie Indizes sorgfältig, um den Anforderungen des Anwendungsprogramms gerecht zu werden.

Tipps zur Planung von relationalen Indizes

Die relationalen Indizes, die Sie erstellen, sollten von den relationalen Daten und den Abfragen abhängig sein, die auf sie zugreifen.

Verwenden Sie den Designadvisor in der Steuerzentrale oder das Tool `db2advis`, um die besten Indizes für eine bestimmte Abfrage oder für eine Reihe von Abfragen, die eine Auslastung definieren, zu finden. Dieses Tool empfiehlt relationale Indizes mit leistungssteigernden Merkmalen wie zum Beispiel INCLUDE-Spalten, übernommenen eindeutigen Indizes und mit ALLOW REVERSE SCANS definierten Indizes.

Die folgenden Richtlinien geben Anhaltspunkte zur Erstellung nützlicher relationaler Indizes für verschiedene Zwecke:

- Definieren Sie zur Vermeidung einiger Sortieroperationen an allen möglichen Stellen Primärschlüssel und eindeutige Schlüssel, indem Sie die Anweisung CREATE UNIQUE INDEX verwenden.
- Fügen Sie eindeutigen Indizes zur Verbesserung des Datenabrufs INCLUDE-Spalten hinzu. Zu diesem Zweck bieten sich Spalten mit folgenden Eigenschaften an:
  - Auf sie wird häufig zugegriffen, sodass die Leistung durch einen reinen Indexzugriff verbessert werden kann.
  - Sie sind zur Begrenzung des Bereichs einer Indexsuche nicht erforderlich.
  - Sie haben keinen Einfluss auf die Reihenfolge oder die Eindeutigkeit des Indexschlüssels.
- Verwenden Sie für einen effizienten Zugriff auf kleine Tabellen relationale Indizes, um häufige Abfragen auf Tabellen mit einer größeren Anzahl von Datenseiten, wie in der Spalte NPAGES in der Katalogsicht SYSCAT.TABLES eingetragen, zu optimieren. Gehen Sie wie folgt vor:
  - Erstellen Sie einen Index für jede Spalte, die beim Join von Tabellen verwendet werden soll.
  - Erstellen Sie einen Index für jede Spalte, die regelmäßig nach bestimmten Werten durchsucht werden soll.
- Treffen Sie eine Entscheidung zwischen aufsteigender und absteigender Reihenfolge von Schlüsseln, je nachdem, welche Reihenfolge am häufigsten benötigt wird, um die Sucheffizienz zu erhöhen. Obwohl die Werte in umgekehrter Richtung gesucht werden können, wenn Sie den Parameter ALLOW REVERSE SCANS in der Anweisung CREATE INDEX angeben, zeigen Suchoperationen in der angegebenen Indexreihenfolge eine etwas bessere Leistung als umgekehrte Suchen.
- Beachten Sie folgende Punkte, um den Aufwand der Indexverwaltung und den Indexspeicherbedarf möglichst gering zu halten:
  - Vermeiden Sie die Erstellung von relationalen Indizes, bei denen es sich um Teilschlüssel der anderen Indexschlüssel für die Spalten handelt. Wenn es

zum Beispiel einen Index für die Spalten a, b und c gibt, ist es im Allgemeinen nicht sinnvoll, einen zweiten Index für die Spalten a und b zu erstellen.

– Erstellen Sie nicht willkürlich für alle Spalten relationale Indizes. Unnötige Indizes belegen nicht nur Speicherplatz, sondern führen auch zu langen Vorbereitungszeiten (PREPARE). Dies spielt besonders für komplexe Abfragen eine wichtige Rolle, wenn eine Optimierungsklasse mit dynamisch programmierter Joinaufzählung (Dynamic Programming Join Enumeration) verwendet wird.

  Beachten Sie die folgende allgemeine Regel für die typische Anzahl von relationalen Indizes, die Sie für eine Tabelle definieren. Diese Anzahl richtet sich nach der primären Verwendung der Datenbank:

  - Erstellen Sie für Umgebungen zur Onlinetransaktionsverarbeitung (OLTP-Umgebungen) nur einen oder zwei Indizes.
  - Für Umgebungen mit reinen Leseabfragen können Sie mehr als fünf Indizes erstellen.
  - Für gemischte Abfrage- und OLTP-Umgebungen können Sie zwischen zwei und fünf Indizes erstellen.

- Erstellen Sie relationale Indizes für Fremdschlüssel, um die Leistung von DELETE- und UPDATE-Operationen an der übergeordneten Tabelle zu verbessern.
- Erstellen Sie zur Verbesserung von DELETE- und UPDATE-Operationen mit MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle), die mit IMMEDIATE und INCREMENTAL definiert sind, eindeutige relationale Indizes für den implizierten eindeutigen Schlüssel der MQT, das heißt, für die Spalten in der GROUP BY-Klausel der MQT-Definition.
- Erstellen Sie relationale Indizes für Spalten, die häufig zum Sortieren der relationalen Daten verwendet werden, um schnelle Sortieroperationen zu ermöglichen.
- Wenn Sie bei der Erstellung eines mehrspaltigen relationalen Index zur Verbesserung der Joinleistung mehrere Auswahlmöglichkeiten für die erste Schlüsselspalte haben, verwenden Sie die Spalte, die am häufigsten mit dem Vergleichselement equijoin („=") angegeben wird, oder die Spalte mit der größten Anzahl unterschiedlicher Werte als ersten Schlüssel.
- Definieren Sie einen Clusterindex, um neu eingefügte Zeilen nach diesem Index in Cluster einzureihen und Seitentrennungen zu vermeiden. Ein Clusterindex sollte die Notwendigkeit, die Tabelle zu reorganisieren, erheblich herabsetzen.

  Verwenden Sie bei der Definition einer Tabelle das Schlüsselwort PCTFREE, um anzugeben, wie viel freier Speicherplatz auf der Seite beibehalten werden soll, damit eingefügte Zeilen auf Seiten geeignet gespeichert werden können. Sie können auch die Klausel `pagefreespace` MODIFIED BY des Befehls LOAD angeben.
- Zur Aktivierung der Online-Indexdefragmentierung verwenden Sie bei der Erstellung von relationalen Indizes die Option MINPCTUSED. Mit der Option MINPCTUSED wird die Schwelle für die Mindestgröße des genutzten Speicherbereichs auf einer äußeren Indexseite (Blattseite) angegeben und die Online-Indexdefragmentierung aktiviert. Dadurch kann sich die Notwendigkeit für eine Reorganisation verringern. Dies geht jedoch auf Kosten einer Leistungseinbuße beim Löschen von Schlüsseln, wenn durch dieses Löschen Schlüssel physisch von der Indexseite entfernt werden.

Ziehen Sie die Erstellung eines relationalen Index unter folgenden Umständen in Betracht:

- Erstellen Sie einen relationalen Index für Spalten, die in WHERE-Klauseln der am häufigsten verarbeiteten Abfragen und Transaktionen verwendet werden.

  Betrachten Sie zum Beispiel die folgende Klausel WHERE:

```
WHERE WORKDEPT='A01' OR WORKDEPT='E21'
```

Diese Klausel wird im Allgemeinen von einem Index für die Spalte WORKDEPT profitieren, sofern die Spalte WORKDEPT nicht viele doppelte Werte enthält.

- Erstellen Sie einen relationalen Index für eine Spalte bzw. Spalten, um die Zeilen in der für die Abfrage erforderlichen Reihenfolge zu sortieren. Das Ordnen ist nicht nur für die Klausel ORDER BY erforderlich, sondern auch für andere Klauseln wie DISTINCT und GROUP BY.

  Im folgenden Beispiel wird die Klausel DISTINCT verwendet:

  ```
  SELECT DISTINCT WORKDEPT
         FROM EMPLOYEE
  ```

  Der Datenbankmanager kann einen Index verwenden, der in aufsteigender oder absteigender Reihenfolge für die Spalte WORKDEPT definiert ist, um doppelte Werte zu eliminieren. Derselbe Index könnte auch verwendet werden, um Werte wie in folgendem Beispiel mit einer Klausel GROUP BY zu gruppieren:

  ```
  SELECT WORKDEPT, AVERAGE(SALARY)
    FROM EMPLOYEE
  GROUP BY WORKDEPT
  ```

- Erstellen Sie einen relationalen Index mit einem zusammengesetzten Schlüssel, der jede Spalte nennt, auf die in einer Anweisung verwiesen wird. Wenn ein Index in dieser Weise angegeben wird, können relationale Daten nur aus dem Index abgerufen werden, was effizienter als der Zugriff auf eine Tabelle ist.

  Betrachten Sie zum Beispiel die folgende SQL-Anweisung:

  ```
  SELECT LASTNAME
    FROM EMPLOYEE
    WHERE WORKDEPT IN ('A00','D11','D21')
  ```

  Wenn ein relationaler Index für die Spalten WORKDEPT und LASTNAME der Tabelle EMPLOYEE definiert ist, kann die Anweisung möglicherweise effizienter verarbeitet werden, indem nur der Index und nicht die gesamte Tabelle durchsucht wird. Beachten Sie, dass hier die Spalte WORKDEPT die erste Spalte des relationalen Index sein sollte, da sich das Vergleichselement auf diese Spalte bezieht.

- Erstellen Sie einen relationalen Index mit INCLUDE-Spalten, um die Verwendung von Indizes für Tabellen zu verbessern. Für das vorige Beispiel ließe sich ein eindeutiger relationaler Index folgendermaßen definieren:

  ```
  CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
  ```

  Die Angabe von lastname als INCLUDE-Spalte und nicht als Teil des Indexschlüssels bedeutet, dass lastname nur auf den äußeren Seiten (Blattseiten) des Index gespeichert wird.

Tipps zur Leistung von relationalen Indizes

Beachten Sie die folgenden Vorschläge zur Verwendung und Verwaltung relationaler Indizes:

- **Angeben eines großen Dienstprogrammzwischenspeichers**

  Sowohl für die Indexerstellung (CREATE INDEX) als auch die Reorganisation von Indizes (REORG INDEXES) wird Schreibzugriff anderer Benutzer bzw. Anwendungen auf die zugrunde liegende Tabelle unterstützt. Wenn Sie ein hohes Aktualisierungsaufkommen an der zugrunde liegenden Tabelle für den relationalen Index erwarten, der erstellt oder reorganisiert wird, sollten Sie die Konfiguration eines großen Dienstprogrammzwischenspeichers in Betracht ziehen. Ein großer Dienstprogrammzwischenspeicher beschleunigt die Erstellung bzw. Reorganisation der Indizes während der Übernahmephase. Alle Schreibvorgänge für die erstellten bzw. reorganisierten Indizes werden in den DB2-Proto-

kollen und im internen Speicherpufferbereich aufgezeichnet. Der interne Speicherpufferbereich ist ein ausgewiesener Speicherbereich, der bei Bedarf vom Dienstprogrammzwischenspeicher zugeordnet wird, um die Änderungen an den erstellten bzw. reorganisierten Indizes zu speichern. Die Verwendung dieses Speichers beschleunigt die Übernahmephase. Der zugeordnete Speicher wird nach Beendigung der Indexerstellung bzw. -reorganisation wieder freigegeben. Die Leistung während der Übernahmephase lässt sich stark verbessern, wenn sichergestellt wird, dass ein ausreichender Dienstprogrammzwischenspeicher zur Verfügung steht, um alle oder zumindest die meisten Änderungen an den erstellten bzw. reorganisierten Indizes aufzunehmen.

- **Erhöhen des Konfigurationsparameters sheapthres bei Verwendung einer SMP-Maschine**

  Jeder Subagent erhält die im Konfigurationsparameter **sortheap** angegebene Speicherkapazität zum Durchsuchen der Tabelle, um Sortierspeicherüberläufe zu vermeiden. Sie sollten die Anzahl der Sortierspeicherüberläufe überwachen und den Wert des Parameters **sheapthres** entsprechend erhöhen.

- **Angeben separater Tabellenbereiche für relationale Indizes**

  Indizes können in einem anderen Tabellenbereich als die Tabellendaten gespeichert werden. Dadurch kann Plattenspeicher eventuell effizienter genutzt werden, indem die Bewegungen der Schreib-/Leseköpfe beim Indexzugriff reduziert werden. Sie können auch Indextabellenbereiche auf schnelleren physischen Einheiten erstellen. Darüber hinaus können Sie den Tabellenbereich für Indizes einem anderen Pufferpool zuordnen, wodurch die Indexseiten vielleicht länger im Puffer bleiben, weil sie nicht mit den Tabellendatenseiten konkurrieren.

  Wenn Sie Indizes nicht in getrennten Tabellenbereichen speichern, verwenden sowohl die Datenseiten als auch die Indexseiten dieselben Werte für EXTENTSIZE und PREFETCHSIZE. Wenn Sie für Indizes einen anderen Tabellenbereich verwenden, können Sie unterschiedliche Werte für alle Merkmale eines Tabellenbereichs auswählen. Da Indizes in der Regel kleiner als Tabellen sind und sich über weniger Container erstrecken, haben Indizes in der Regel auch kleinere Werte für EXTENTSIZE (z. B. 8 und 16 Seiten). Das Abfrageoptimierungsprogramm beachtet die Geschwindigkeit der Einheit für einen Tabellenbereich bei der Auswahl eines Zugriffsplans.

- **Sicherstellen des Grads der Clusterbildung**

  Wenn für Ihre SQL-Anweisung eine Reihenfolge erforderlich ist (wenn sie zum Beispiel die SQL-Klauseln ORDER BY, GROUP BY und DISTINCT enthält), wählt das Optimierungsprogramm den Index möglicherweise nicht aus, selbst wenn die Reihenfolge in folgenden Fällen erfüllt wird:

  – Wenn der Grad der Index-Clusterbildung gering ist. Untersuchen Sie die Spalten CLUSTERRATIO und CLUSTERFACTOR des Katalogs SYSCAT.INDEXES, um Informationen zu erhalten.

  – Wenn die Tabelle so klein ist, dass es weniger aufwendig ist, die Tabelle zu durchsuchen und die Ergebnismenge im Hauptspeicher zu sortieren.

  – Wenn für den Zugriff auf die Tabelle konkurrierende Indizes gibt

  Führen Sie nach der Erstellung eines Clusterindex einen Befehl REORG TABLE im klassischen Modus aus, wodurch ein perfekt organisierter Index erstellt wird. Zur erneuten Herstellung der Clusterbildung der Tabelle müssen Sie eventuell eine Sortieroperation oder eine LOAD-Operation durchführen. Im Allgemeinen ist die Clusterbildung in einer Tabelle nur anhand eines Index möglich. Erstellen Sie weitere Indizes, nachdem Sie den Clusterindex erstellt haben. Ein Clusterindex versucht, eine bestimmte Reihenfolge der Daten zu erhalten, wodurch die vom Dienstprogramm RUNSTATS gesammelten statistischen Werte für CLUSTERRATIO bzw. CLUSTERFACTOR verbessert werden.

Zur Erhaltung des Clusterverhältnisses können Sie einen geeigneten Wert für PCTFREE beim Ändern einer Tabelle angeben, bevor Sie diese Tabelle mit Daten füllen (LOAD) oder reorganisieren. Der durch PCTFREE angegebene freie Speicherplatz auf jeder Seite reserviert Platz für Einfügungen, sodass diese Einfügungen der Clusterbildung entsprechend vorgenommen werden können. Wenn Sie PCTFREE für die Tabelle nicht angeben, wird der gesamte freie Speicherplatz durch die Reorganisation beseitigt.

Anmerkung: Die Clusterbildung wird zurzeit bei Aktualisierungen (UPDATE) nicht beibehalten, sofern es sich nicht um Bereichsclustertabellen handelt. Das heißt, wenn Sie einen Datensatz aktualisieren, sodass sich der Schlüsselwert im Clusterindex ändert, wird dieser Datensatz nicht unbedingt auf eine entsprechend andere Seite versetzt, um die Clusterreihenfolge beizubehalten. Verwenden Sie zur Beibehaltung der Clusterbildung anstelle der Anweisung UPDATE die Anweisung DELETE und anschließend INSERT.

- **Aktualisieren der Tabellen- und Indexstatistiken**

  Führen Sie nach der Erstellung eines neuen relationalen Index das Dienstprogramm RUNSTATS aus, um Indexstatistikdaten zu sammeln. Anhand dieser Statistikdaten kann das Optimierungsprogramm bestimmen, ob durch die Verwendung des Index die Zugriffsleistung verbessert werden kann.

- **Aktivieren der Online-Indexdefragmentierung**

  Die Online-Indexdefragmentierung wird aktiviert, wenn die Klausel MINPCTUSED auf einen Wert größer null für den relationalen Index gesetzt wird. Die Online-Indexdefragmentierung ermöglicht eine Komprimierung von Indizes durch Zusammenfügen von Blattseiten, wenn der freie Speicherplatz auf einer Seite den angegebenen Wert erreicht oder unterschreitet, wobei der Index jedoch verfügbar bleibt.

- **Reorganisieren von relationalen Indizes nach Bedarf**

  Um optimale Leistungsvorteile aus den Indizes zu ziehen, sollten Sie eine regelmäßige Reorganisation der Indizes in Betracht ziehen, da Aktualisierungen an Tabellen dazu führen, dass der Vorablesezugriff auf Indexseiten an Effizienz einbüßt.

  Zur Reorganisation eines Index können Sie ihn entweder löschen und neu erstellen oder das Dienstprogramm REORG verwenden.

  Geben Sie zur Verringerung der Notwendigkeit für häufige Reorganisationen bei der Erstellung eines relationalen Index einen geeigneten Wert für PCTFREE an, um einen Prozentsatz an freiem Speicherplatz auf jeder Indexblattseite, wenn sie erstellt wird, freizuhalten. So ist bei zukünftigen Aktivitäten, durch die Datensätze in den Index eingefügt werden, die Wahrscheinlichkeit geringer, dass Indexseitentrennungen verursacht werden. Seitentrennungen bewirken, dass Indexseiten nicht mehr direkt aufeinander folgen oder sequenziell sind, was wiederum zu einer verminderten Effizienz des Vorablesezugriffs auf Indexseiten führt.

  Anmerkung: Der bei der Erstellung des relationalen Index angegebene PCTFREE wird beibehalten, wenn der Index reorganisiert wird.

  Durch Löschen und erneutes Erstellen bzw. Reorganisieren des relationalen Index wird auch eine neue Gruppe von Seiten erstellt, die weitgehend zusammenhängend und sequenziell sind. Dies verbessert den Vorablesezugriff auf Indexseiten. Obwohl das Dienstprogramm REORG TABLE mehr Aufwand an Zeit und Ressourcen erfordert, stellt es sicher, dass die Datenseiten in Clustern angeordnet werden. Die Clusterbildung bietet größere Vorteile für Indexsuchen, durch die auf eine bedeutende Anzahl von Datenseiten zugegriffen wird.

- **Analysieren von EXPLAIN-Informationen zur Nutzung des relationalen Index**

  Führen Sie in regelmäßigen Abständen EXPLAIN für Ihre am häufigsten verwendeten Abfragen aus, und überprüfen Sie, ob jeder Ihrer relationalen Indizes wenigstens einmal verwendet wird. Wenn ein Index in keiner Abfrage verwendet wird, empfiehlt es sich, diesen Index zu löschen.

  EXPLAIN-Informationen geben Ihnen auch Einblick, ob Tabellensuchen in großen Tabellen als innere Tabellen bei Joins mit Verschachtelungsschleife verarbeitet werden. Dies würde darauf hinweisen, dass ein Index für die Spalte des Joinvergleichselements entweder fehlt oder als ineffektiv für die Anwendung des Joinvergleichselements betrachtet wird.
- **Verwenden flüchtiger Tabellen für Tabellen mit stark variierender Größe**

  Eine *flüchtige* Tabelle ist eine Tabelle, deren zur Ausführungszeit zwischen leer und sehr groß schwanken kann. Für diese Art von Tabelle, bei der die Kardinalität stark variiert, kann das Optimierungsprogramm einen Zugriffsplan generieren, der eine Tabellensuche einer Index vorzieht.

  Durch die Deklaration einer Tabelle als „flüchtig" in der Anweisung ALTER TABLE...VOLATILE ist es dem Optimierungsprogramm möglich, eine Indexsuche auf der flüchtigen Tabelle auszuführen. In den folgenden Fällen verwendet das Optimierungsprogramm unabhängig von den statistischen Daten eine Indexsuche anstelle einer Tabellensuche:

  – Wenn alle Spalten, auf die verwiesen wird, im Index vorhanden sind.

  – Wenn der Index bei der Indexsuche ein Vergleichselement anwenden kann.

  Wenn es sich bei der Tabelle um eine typisierte Tabelle handelt, wird die Verwendung der Anweisung ALTER TABLE...VOLATILE nur für die Stammtabelle der Hierarchie der typisierten Tabelle unterstützt.

Indexbereinigung und Indexpflege

Wenn Sie Indizes erstellen, sinkt die Leistung, sofern Sie nicht dafür sorgen, dass die Indizes kompakt und organisiert bleiben. Berücksichtigen Sie die folgenden Empfehlungen, um Indizes so klein und effizient wie möglich zu halten:

- Aktivieren Sie die Online-Indexdefragmentierung.

  Erstellen Sie Indizes mit der Klausel MINPCTUSED. Löschen Sie vorhandene Indizes und erstellen Sie sie erneut, falls erforderlich.
- Führen Sie häufige Commits durch oder aktivieren Sie X-Sperren für Tabellen, entweder explizit oder durch Sperreneskalation, falls häufige Commits nicht möglich sind.

  Indexschlüssel, die als gelöscht markiert sind, können nach dem Commit aus der Tabelle entfernt werden. X-Sperren für Tabellen ermöglichen es, den gelöschten Schlüssel physisch zu entfernen, wenn er als gelöscht markiert ist, wie weiter unten erläutert wird.
- Verwenden Sie REORGCHK, um festzustellen, wann Indizes oder Tabellen (oder beides) zu reorganisieren sind und wann REORG INDEXES mit der Option CLEANUP ONLY zu verwenden ist.

  Um einen Schreib- und Lesezugriff auf einen Index bei der Reorganisation zuzulassen, führen Sie den Befehl REORG INDEXES mit der Option ALLOW WRITE ACCESS aus.

Anmerkung: In DB2 Version 8.1 und späteren Versionen werden alle Indizes als Indizes des Typs 2 erstellt. Eine Ausnahme besteht dann, wenn Sie einer Tabelle einen Index hinzufügen, die bereits Indizes des Typs 1 hat. In diesem Fall wird auch der neue Index als Index des Typs 1 erstellt. Zur Ermittlung des Typs von

Index, der für eine Tabelle vorhanden ist, führen Sie den Befehl INSPECT aus. Zur Umwandlung von Indizes des Typs 1 in Indizes des Typs 2 führen Sie den Befehl REORG INDEXES aus.

Die Hauptvorteile von Indizes des Typs 2 lassen sich folgendermaßen beschreiben:

- Ein Index kann für Spalten erstellt werden, deren Länge 255 Byte überschreitet.
- Das Sperren der nächsten Schlüssel wird auf ein Minimum reduziert, wodurch sich der gemeinsame Zugriff verbessert. Der größte Teil des Sperrens der nächsten Schlüssel wird vermieden, weil ein Schlüssel nur als gelöscht markiert, jedoch nicht physisch von der Indexseite entfernt wird. Informationen über das Sperren von Schlüsseln finden Sie in den Abschnitten über die Auswirkungen von Sperren auf die Leistung.

Indexschlüssel, die als gelöscht markiert sind, werden bei folgenden Aktionen bereinigt:

- Während nachfolgender Einfüge-, Aktualisierungs- oder Löschaktivitäten

  Während des Einfügens von Schlüsseln werden Schlüssel, die als gelöscht markiert und bekanntermaßen festgeschrieben sind, bereinigt, wenn eine solche Bereinigung die Durchführung einer Seitentrennung vermeiden hilft und verhindert, dass der Index größer wird.

  Während des Löschens von Schlüsseln wird, wenn sämtliche Schlüssel auf einer Seite als gelöscht markiert wurden, ein Versuch unternommen, eine andere Indexseite zu finden, auf der alle Schlüssel als gelöscht markiert sind und alle diese Löschungen mit COMMIT festgeschrieben wurden. Wenn eine solche Seite gefunden wird, wird sie aus der Indexstruktur gelöscht.

  Wenn beim Löschen eines Schlüssels eine X-Sperre für die Tabelle aktiv ist, wird der Schlüssel physisch gelöscht und nicht nur als gelöscht markiert. Während dieser physischen Löschung werden alle gelöschten Schlüssel auf derselben Seite ebenfalls entfernt, wenn sie als gelöscht markiert sind und bekannt ist, dass diese Löschung festgeschrieben ist.
- Bei der Ausführung des Befehls REORG INDEXES mit CLEANUP-Optionen

  Die Option CLEANUP ONLY PAGES sucht nach Indexseiten und gibt diese frei, wenn auf ihnen alle Schlüssel als gelöscht markiert sind und diese Löschung mit COMMIT festgeschrieben wurde.

  Die Option CLEANUP ONLY ALL gibt nicht nur Indexseiten frei, auf denen alle Schlüssel als gelöscht markiert und festgeschrieben sind, sondern entfernt zudem auch Satz-IDs (RIDs), die als gelöscht markiert sind und deren Löschung festgeschrieben wurde, auf Seiten, die auch einige nicht gelöschte Satz-IDs enthalten.

  Diese Option versucht außerdem, benachbarte Blattseiten zusammenzufügen, wenn dies zu einer zusammengefügten Blattseite führt, die mindestens den durch PCTFREE angegebenen freien Speicherbereich enthält. Der Wert für PCTFREE ist der Prozentsatz an freiem Speicherbereich, der für den Index bei seiner Erstellung definiert ist. Der Standardwert für PCTFREE ist 10 %. Wenn zwei Seiten zusammengefügt werden können, wird eine der Seiten freigegeben.

  Bei partitionierten Tabellen sollten Sie RUNSTATS nach Abschluss einer asynchronen Indexbereinigung ausführen, um genaue Indexstatistiken mit vorhandenen Datenpartitionen, deren Zuordnung aufgehoben wurde, zu generieren. Um festzustellen, ob in der Tabelle Datenpartitionen vorhanden sind, deren Zuordnung aufgehoben wurde, können Sie das Statusfeld in der Tabelle SYSDATAPARTITIONS überprüfen und nach dem Wert ″I″ (Indexbereinigung) oder ″D″ (Zuordnung zu abhängiger MQT aufgehoben) suchen.

- Bei einer Neuerstellung eines Index

  Zu den Dienstprogrammen, die Indizes neu erstellen, gehören folgende:
  - Der Befehl REORG INDEXES, wenn er nicht mit einer der CLEANUP-Optionen verwendet wird
  - Der Befehl REORG TABLE, wenn er nicht mit der Option INPLACE verwendet wird
  - Der Befehl IMPORT mit der Option REPLACE
  - Der Befehl LOAD mit der Option INDEXING MODE REBUILD

Indexverhalten bei partitionierten Tabellen

Indizes für partitionierte Tabellen sind Indizes für normale Tabellen insofern ähnlich, als dass jeder Index Zeiger auf Zeilen in allen Datenpartitionen der Tabelle enthält. Ein wichtiger Unterschied besteht jedoch darin, dass jeder Index für eine partitionierte Tabelle ein unabhängiges Objekt ist. In Umgebungen mit partitionierten Datenbanken wird der Index über die Datenbankpartitionen in der gleichen Weise verteilt wie die Tabelle. Da ein Index für eine partitionierte Tabelle unabhängig von anderen Indizes einsetzbar ist, sind einige besondere Aspekte im Hinblick darauf zu beachten, welcher Tabellenbereich beim Erstellen eines Indexes für eine partitionierte Tabelle verwendet wird.

Ein Index für eine partitionierte Tabelle wird in einem einzigen Tabellenbereich erstellt, auch wenn sich die Datenpartitionen der Tabelle über mehrere Tabellenbereiche erstrecken. Sowohl DMS- als auch SMS-Tabellenbereiche unterstützen die Verwendung von Indizes, die sich an einer anderen Position befinden als die Tabelle. Alle angegebenen Tabellenbereiche müssen sich in derselben Datenbankpartitionsgruppe befinden. Jeder Index kann in einem eigenen Tabellenbereich, auch in großen Tabellenbereichen, abgelegt werden. Jeder Tabellenbereich für einen Index muss denselben Speichermechanismus wie die Datenpartitionen verwenden (entweder DMS oder SMS). Indizes in großen Tabellenbereichen können bis zu 2^29 Seiten enthalten.

Ein Index für eine partitionierte Tabelle bietet folgende zusätzliche Vorteile:

- Beim Löschen und bei der Online-Erstellung von Indizes wird eine bessere Leistung erzielt.
- Es besteht die Möglichkeit, verschiedene Werte für die Tabellenbereichsmerkmale für verschiedene Indizes der Tabelle zu verwenden (z. B. können verschiedene Seitengrößen für die einzelnen Indizes zu einer besseren Speicherplatznutzung beitragen).
- Weniger E/A-Konkurrenzsituationen lassen einen effizienteren gemeinsamen Zugriff auf die Indexdaten für die Tabelle zu.
- Wenn einzelne Indizes gelöscht werden, wird der Speicherplatz sofort für das System verfügbar, ohne dass eine Indexreorganisation erforderlich ist.
- Wenn eine Indexreorganisation ausgeführt werden soll, kann ein einzelner Index reorganisiert werden.

Abb. 16 auf Seite 120 zeigt einen nicht partitionierten Index für eine partitionierte Tabelle, die sich in einem einzigen Tabellenbereich befindet.

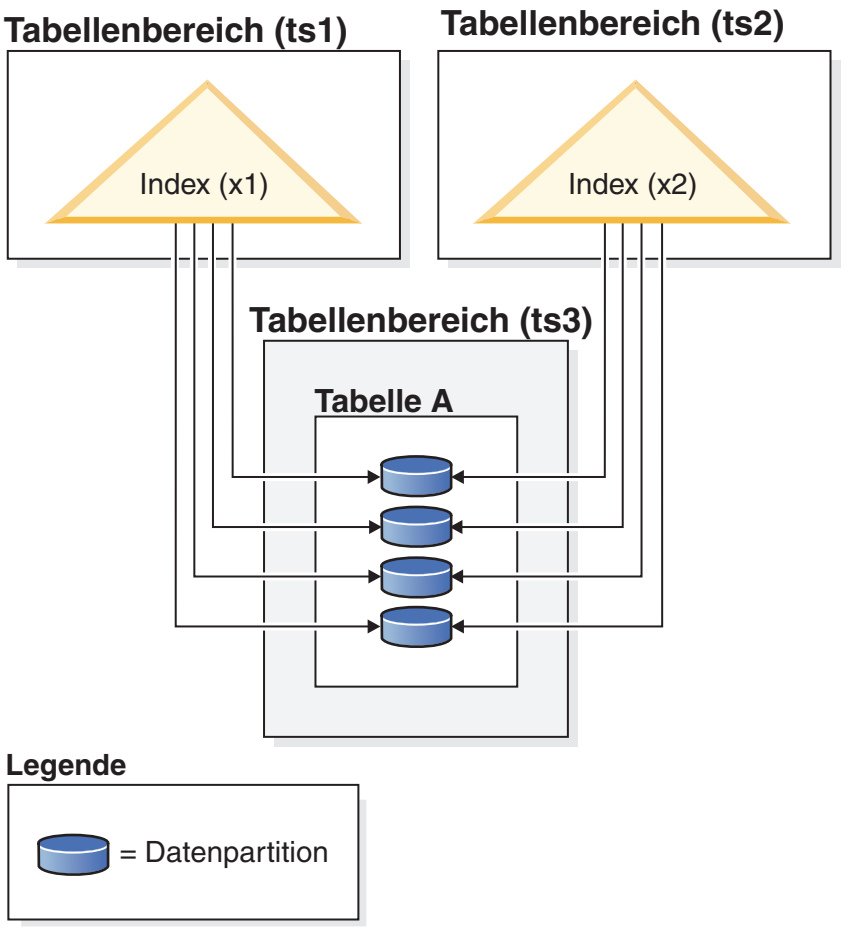


Abbildung 16. Indexverhalten bei einer partitionierten Tabelle

Abb. 17 auf Seite 121 zeigt das Indexverhalten bei einer partitionierten Tabelle, die gleichzeitig über mehrere Datenbankpartitionen verteilt ist.

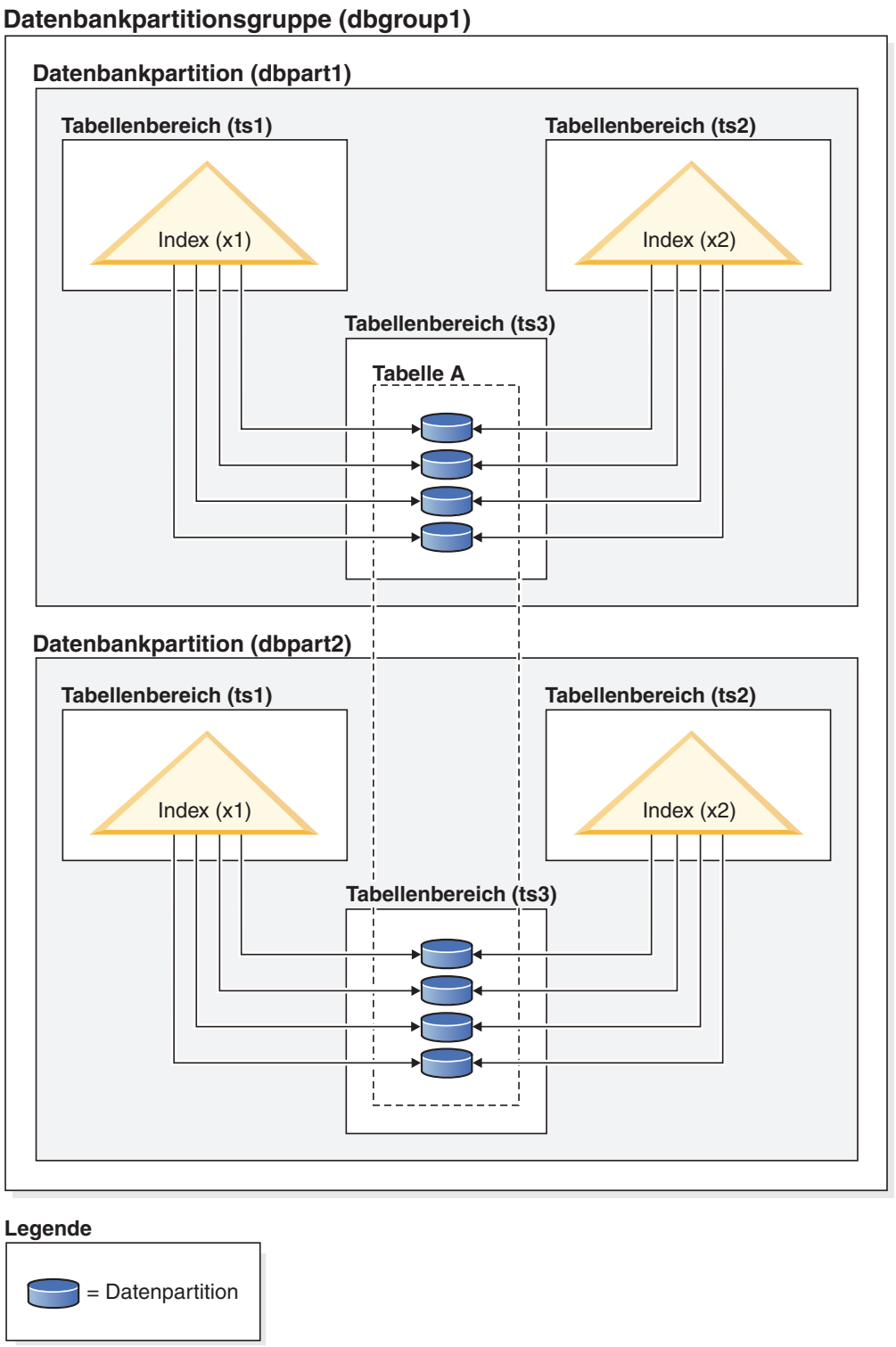


Abbildung 17. Indexverhalten bei einer Tabelle, die sowohl verteilt als auch partitioniert ist.

Sie können einen Tabellenbereich für Indizes für eine partitionierte Tabelle in der Anweisung CREATE INDEX ...IN <tb_bereich1> angeben, der ein anderer ist, als der Tabellenbereich für Indizes, der in der Anweisung CREATE TABLE ... INDEX IN <tb_bereich2> angegeben wurde.

Nur für partitionierte Tabellen könne Sie die Indexspeicherposition durch die IN-Klausel der Anweisung CREATE INDEX überschreiben, sodass Sie eine Tabellenbereichsposition für den Index angegeben können. Durch diese Methode können Sie verschiedene Indizes für eine partitionierte Tabelle je nach Bedarf in einem anderen Tabellenbereich speichern. Wenn Sie eine partitionierte Tabelle erstellen, ohne die Speicherposition für die zugehörigen nicht partitionierten Indizes anzugeben, und einen Index mit der Anweisung CREATE INDEX erstellen, ohne einen bestimmten Tabellenbereich anzugeben, wird der Index im Tabellenbereich der ersten zugeordneten bzw. sichtbaren Datenpartition erstellt. Zur Bestimmung, wo der Index zu erstellen ist, werden die drei folgenden möglichen Fälle in der angegebenen Reihenfolge (beginnend mit Fall 1) ausgewertet. Die Auswertung wird beendet, wenn einer der Fälle zutrifft:

```
Fall 1:
Wenn ein Tabellenbereich für den Index in der Anweisung CREATE INDEX ... IN
<tb_bereich1> angegeben wird, wird der in <tb_bereich1> angegebene
Tabellenbereich für diesen Index verwendet.
Fall 2:
Wenn ein Tabellenbereich für den Index in der Anweisung CREATE TABLE .. INDEX IN
<tb_bereich2> angegeben wird, wird der in <tb_bereich2> angegebene
Tabellenbereich für diesen Index verwendet.
Fall 3:
Wenn kein Tabellenbereich definiert ist, wird der Tabellenbereich verwendet, der
von der ersten zugeordneten oder sichtbaren Datenpartition verwendet wird.
```

Wo der Index erstellt wird, hängt davon ab, was bei der Ausführung der Anweisung CREATE TABLE geschieht. Wenn Sie für nicht partitionierte Tabellen keine Klausel INDEX IN angeben, fügt die Datenbank diese Information für Sie ein und gibt denselben Tabellenbereich wie für die Datentabelle an. Wenn Sie für partitionierte Tabelle diese Information weglassen, wird sie nicht nachgetragen und es gilt Fall 3.

Beispiel 1: Dieses Beispiel geht von der Annahme aus, dass eine partitionierte Tabelle 'sales' mit den Spalten (a int, b int, c int) vorhanden ist. Es wird ein eindeutiger Index 'a_idx' im Tabellenbereich 'ts1' erstellt.

```
CREATE UNIQUE INDEX a_idx ON sales ( a ) IN ts1
```

Beispiel 2: Dieses Beispiel geht von der Annahme aus, dass eine partitionierte Tabelle 'sales' mit den Spalten (a int, b int, c int) vorhanden ist. Es wird ein Index 'b_idx' im Tabellenbereich 'ts2' erstellt.

```
CREATE INDEX b_idx ONsales ( b ) IN ts2
```

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.

Online-Indexdefragmentierung

Eine Onlinedefragmentierung wird durch den benutzerdefinierbaren Schwellenwert für die minimale Größe des verwendeten Speicherbereichs auf einer Indexseite (Blattseite) ermöglicht. Wenn ein Indexschlüssel aus einer Blattseite gelöscht und dabei der Schwellenwert überschritten wird, werden die benachbarten Indexseiten daraufhin überprüft, ob zwei Blattseiten zusammengefügt werden können. Wenn auf einer Seite ausreichend Platz zum Zusammenfügen zweier benachbarter Seiten vorhanden ist, erfolgt die Zusammenfügung unverzüglich im Hintergrund.

Eine Onlinedefragmentierung des Index ist nur möglich für Indizes, die in Version 6 und späteren Versionen erstellt wurden. Wenn vorhandene Indizes die Möglichkeit der Onlinezusammenfügung benötigen, müssen sie gelöscht und mit der Klausel MINPCTUSED erneut erstellt werden. Setzen Sie MINPCTUSED auf einen Wert kleiner als 100. Der empfohlene Wert für MINPCTUSED ist kleiner als 50, da der Zweck darin besteht, zwei benachbarte Indexblattseiten zusammenzufügen. Durch den Wert 0 für MINPCTUSED, der gleichzeitig der Standardwert ist, wird die Onlinedefragmentierung inaktiviert.

Seiten im Index werden freigegeben, wenn der letzte Indexschlüssel auf der betreffenden Seite entfernt wird. Eine Ausnahme zu dieser Regel ist der Fall, wenn Sie die Klausel MINPCTUSED in der Anweisung CREATE INDEX angeben. Mit der Klausel MINPCTUSED wird ein Prozentwert des Speicherplatzes auf einer Indexblattseite angegeben. Wenn ein Indexschlüssel gelöscht wird und der Prozentsatz des gefüllten Speicherplatzes auf der Seite bei oder unter dem angegebenen Wert liegt, versucht der Datenbankmanager die verbleibenden Schlüssel mit Schlüsseln einer benachbarten Seite auf eine Seite zusammenzufügen. Wenn genügend Platz auf einer benachbarten Seite vorhanden ist, wird die Zusammenfügung ausgeführt und eine Indexblattseite gelöscht.

Seiten von Indizes, die keine Blattseiten sind, werden bei einer Online-Indexdefragmentierung nicht zusammengefügt. Jedoch werden leere Nichtblattseiten gelöscht und zur Wiederverwendung durch andere Indizes für die gleiche Tabelle verfügbar gemacht. Um diese Nichtblattseiten für andere Objekte in einem DMS-Speichermodell freizugeben oder um Plattenspeicherplatz in einem SMS-Speichermodell freizugeben, führen Sie eine vollständige Reorganisation der Tabelle oder der Indizes aus. Eine vollständige Reorganisation der Tabelle und der Indizes kann den Index auf minimale Größe verkleinern. Nichtblattseiten von Indizes werden bei einer Online-Indexdefragmentierung nicht zusammengefügt, jedoch gelöscht und zur Wiederverwendung freigegeben, wenn sie leer werden. Die Anzahl der Stufen im Index und die Anzahl der Blatt- und Nichtblattseiten können sich verringern.

Für Indizes des Typs 2 werden Schlüssel von einer Seite beim Löschen von Schlüsseln nur dann entfernt, wenn eine exklusive Sperre (Modus X) für die Tabelle aktiviert ist. Bei einer solchen Operation ist eine Online-Indexdefragmentierung effektiv. Wenn beim Löschen von Schlüsseln hingegen keine X-Sperre für die Tabelle aktiviert ist, werden Schlüssel als gelöscht markiert, jedoch nicht physisch von der Indexseite entfernt. Infolgedessen wird auch kein Versuch einer Defragmentierung unternommen.

Zur Defragmentierung von Indizes des Typs 2, in denen Schlüssel zwar als gelöscht markiert werden, jedoch in der physischen Indexseite verbleiben, führen Sie den Befehl REORG INDEXES mit der Option CLEANUP ONLY ALL aus. Die Option CLEANUP ONLY ALL defragmentiert den Index ohne Rücksicht auf den Wert von MINPCTUSED. Wenn Sie den Befehl REORG INDEXES mit der Option CLEANUP ONLY ALL ausführen, werden zwei benachbarte Blattseiten zusammengefügt, wenn eine solche Zusammenfügung mindestens den Wert von PCTFREE an freiem Speicherplatz auf der zusammengefügten Seite übrig lassen kann. PCTFREE wird bei der Erstellung eines Index angegeben und hat den Standardwert 10 Prozent.

Clusterindexverhalten bei partitionierten Tabellen

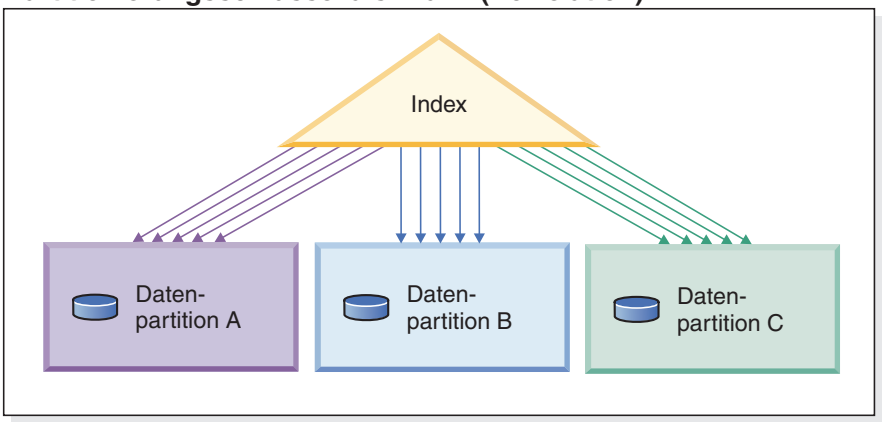
Clusterindizes bieten für partitionierte Tabellen die gleichen Vorteile wie für reguläre Tabellen. Allerdings ist bei der Auswahl eines Clusterindex im Hinblick auf die Definitionen von Tabellenpartitionierungsschlüsseln Sorgfalt geboten.

Sie können Clusterindizes für eine partitionierte Tabelle mit einem beliebigen Clusterschlüssel erstellen. Der Datenbankserver versucht den Clusterindex zu verwenden, um die Daten lokal in jeder Datenpartition in Clustern zusammenzufassen. Bei einer Einfügung über einen Clusterindex wird der Index durchsucht, um eine passende Zeilenkennung (Satz-ID, RID) zu ermitteln. Diese RID wird als Ausgangspunkt für die Suche nach einem Platz in der Tabelle zum Einfügen des Datensatzes verwendet. Zur Erzielung einer gut gepflegten Clusterbildung mit guter Leistung sollte es eine Korrelation zwischen den Indexspalten und den Spalten des Tabellenpartitionierungsschlüssels geben. Eine Methode, eine solche Korrelation sicherzustellen, besteht darin, den Indexspalten die Spalten des Tabellenpartitionierungsschlüssels als Präfix voranzustellen, wie im folgenden Beispiel gezeigt:

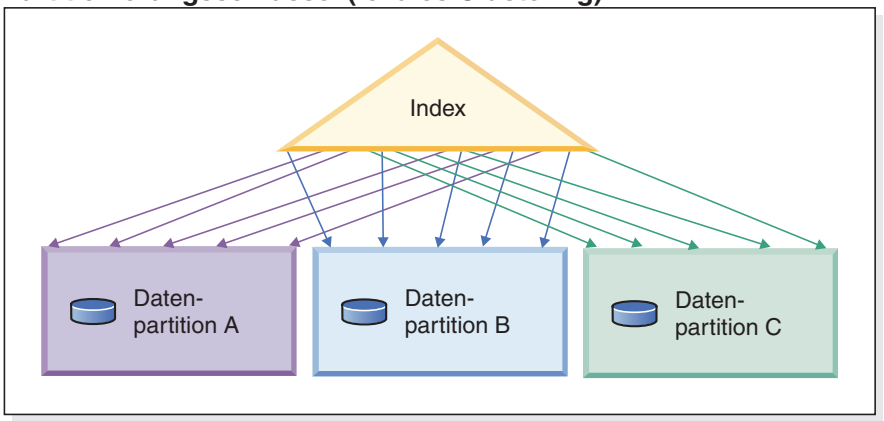
```
PARTITION BY RANGE (Month, Region)
CREATE INDEX ...(Month, Region, Department) CLUSTER
```

Obwohl der Datenbankserver diese Korrelation nicht zwingend umsetzt, ist zu erwarten, dass alle Schlüssel im Index nach Partitions-IDs zusammen gruppiert werden, um eine gute Clusterbildung zu erreichen. Wenn zum Beispiel eine Tabelle nach Quartal partitioniert ist und ein Clusterindex für Datum definiert wird, weil zwischen Quartal und Datum eine Beziehung besteht, kann eine optimale Clusterbildung der Daten mit guter Leistung erzielt werden, weil alle Schlüssel einer Datenpartition im Index zusammen gruppiert werden.

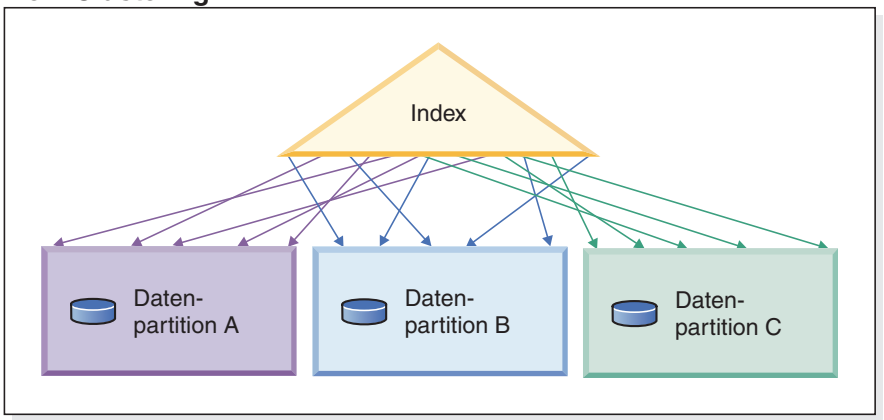
Clustering mit dem Partitionierungsschlüssel als Präfix (Korrelation)



Clustering entspricht nicht dem Partitionierungsschlüssel (lokales Clustering)



Kein Clustering



Legende

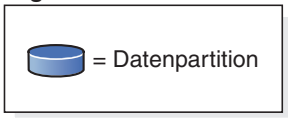


Abbildung 18. Die möglichen Effekte eines Clusterindex für eine partitionierte Tabelle. In der ersten Abbildung werden die Daten sowohl global als auch lokal zu Clustern zusammengefasst.

Wie in Abb. 18 auf Seite 127 gezeigt, wird bei dem gegebenen Layout des Indexes und der Daten in den einzelnen Beispielen eine optimale Suchleistung erzielt, wenn die Clusterbildung mit dem Tabellenpartitionierungsschlüssel korreliert ist. Wenn die Clusterbildung nicht mit dem Tabellenpartitionierungsschlüssel korreliert ist, ist es unwahrscheinlich, dass die Werte im Index lokal zu Clustern zusammengefasst werden. Da eine Korrelation zwischen den Tabellenpartitionierungsspalten und den Indexspalten erwartet wird, ist ein Szenario mit perfektem lokalen Clustering sehr unwahrscheinlich.

Das Clustering bietet folgende Vorteile:

- Innerhalb jeder Datenpartition sind die Zeilen in der Reihenfolge des Schlüssels angeordnet.
- Clusterindizes verbessern die Leistung von Suchoperationen, welche die Tabelle in der Reihenfolge der Schlüssel durchqueren. Dies liegt daran, dass die Suche die erste Zeile der ersten Seite abruft, dann jede Zeile auf eben dieser Seite, bis alle Zeilen für diese Seite abgerufen sind. Anschließend fährt die Suche mit der nächsten Seite fort. Dies bedeutet, dass sich nur eine Seite der Tabelle zu einem gegebenen Zeitpunkt im Pufferpool befinden muss. Wenn die Tabelle hingegen nicht zu Clustern zusammengefasst ist, wird jede Zeile mit hoher Wahrscheinlichkeit von einer anderen Seite abgerufen. Sofern im Pufferpool nicht genügend Platz zur Aufnahme der gesamten Tabelle ist, führt dies dazu, dass jede Seite mehr als einmal abgerufen wird und sich die Suche erheblich verlangsamt.

  Für partitionierte Tabellen kann der Idealfall, dass jede Seite bei der Suche jeweils nur einmal abgerufen wird, nur dann garantiert werden, wenn der Tabellenpartitionierungsschlüssel ein Präfix des Clusterschlüssels (siehe erstes Beispiel in Abb. 18 auf Seite 127 ist. Wenn der Clusterschlüssel jedoch nicht wie zuvor beschrieben mit dem Tabellenpartitionierungsschlüssel korreliert ist und die Daten lokal zu Clustern zusammengefasst sind, können Sie immer noch den vollen Vorteil des Clusterindex ausschöpfen, wenn im Pufferpool genügend Platz ist, um eine Seite jeder Datenpartition aufzunehmen. Der Grund hierfür ist, dass jede Zeile, die für eine gegebene Datenpartition abgerufen wird, in der Nähe einer vorigen Zeile liegt, die für dieselbe Datenpartition abgerufen wurde (siehe zweites Beispiel in Abb. 18 auf Seite 127). Wie zuvor erwähnt, wird die Clusterbildung möglicherweise nicht gut aufrechterhalten, wenn der Clusterschlüssel nicht mit dem Tabellenpartitionierungsschlüssel korreliert ist. Wenn Sie jedoch kein hohes Volumen an Einfüge-, Aktualisierungs- und Löschoperationen für Ihre Tabelle erwarten, sollte diese Lösung von Vorteil sein.

  Selbst wenn nicht genügend Platz für eine Seite jeder Datenpartition im Pufferpool vorhanden ist, lassen sich dennoch einige Vorteile aus der Definition eines Clusterindex gewinnen.

Datenbankagenten

Für jede Datenbank, auf die eine Anwendung zugreift, werden verschiedene Prozesse oder Threads gestartet, um die verschiedenen Anwendungstasks durchzuführen. Zu diesen Tasks gehören das Protokollieren, die Kommunikation und der Vorablesezugriff.

Datenbankagenten sind Threads innerhalb des Datenbankmanagers, die zur Ausführung von Anwendungsanforderungen verwendet werden. In Version 9.5 werden Agenten auf allen Plattformen als Threads ausgeführt.

Die maximale Anzahl von Anwendungsverbindungen wird durch den Konfigurationsparameter *max_connections* des Datenbankmanagers gesteuert. Die Arbeit jeder Anwendungsverbindung wird durch einen einzigen Verarbeitungsagenten koordiniert.

Ein *Verarbeitungsagent* führt Anwendungsanforderungen aus, ist aber nicht dauerhaft mit einer bestimmten Anwendung verbunden. Koordinatoragenten sind am längsten einer Anwendung zugeordnet, da sie mit ihr verbunden bleiben, bis die Anwendung die Verbindung trennt. Die einzige Ausnahme von dieser Regel ist der Fall, wenn der Konzentrator der Steuerkomponente aktiviert ist, sodass ein Koordinatoragent diese Zuordnung an Transaktionsgrenzen (COMMIT, ROLLBACK der Transaktion) beenden kann.

Es gibt vier Typen von Verarbeitungsagenten:

- Nicht zugeordnete Agenten
- Aktive Koordinatoragenten
- Subagenten

Nicht zugeordnete Agenten

Dies ist die einfachste Form eines Verarbeitungsagenten. Er hat keine abgehende Verbindung und keine lokale Datenbankverbindung oder Instanzverbindung.

Aktive Koordinatoragenten

Jede Datenbankverbindung einer Clientanwendung hat einen einzelnen aktiven Agenten, der die Arbeit der Anwendung an der Datenbank koordiniert. Wenn der Koordinatoragent erstellt ist, führt er alle Datenbankanforderungen für die zugehörige Anwendung aus und kommuniziert mit anderen Agenten per Interprozesskommunikation (IPC) oder über Protokolle zur Fernverbindung. Jeder Agent arbeitet mit seinem eigenen privaten Speicher und benutzt Ressourcen des Datenbankmanagers und globale Datenbankressourcen, wie z. B. den Pufferpool, gemeinsam mit anderen Agenten. Wenn eine Transaktion abgeschlossen wird, kann der aktive Koordinatoragent zu einem inaktiven Agenten werden.

Wenn ein Client die Verbindung zu einer Datenbank oder zu einer Instanz trennt, geschieht mit dem koordinierenden Agenten Folgendes:

- Er wird ein aktiver Agent. Wenn andere Verbindungen warten, wird der Verarbeitungsagent ein aktiver Koordinatoragent.
- Er wird freigegeben und als nicht zugeordnet markiert, wenn keine Verbindungen warten und die maximale Zahl von Poolagenten automatisch verwaltet wird oder noch nicht erreicht wurde.
- Er wird beendet und der von ihm verwendete Speicher wird freigegeben, wenn keine Verbindungen warten und die maximale Zahl von Poolagenten erreicht wurde.

Subagenten

Der Koordinatoragent verteilt Datenbankanforderungen an Subagenten, die ihrerseits die Anforderungen für die Anwendung ausführen. Wenn der Koordinatoragent erstellt ist, verarbeitet er alle Datenbankanforderungen für seine Anwendung, indem er die Subagenten koordiniert, die die Anforderungen in der Datenbank ausführen. In DB2 Version 9.5 können Subagenten auch in nicht partitionierten Umgebungen sowie in Umgebungen, in denen die abfrageinterne Parallelität nicht aktiviert ist, vorhanden sein.

Agenten, die keine Arbeit für Anwendungen ausführen und darauf warten, zugeordnet zu werden, gelten als nicht zugeordnete Agenten im Bereitschaftsmodus und befinden sich in einem *Agentenpool*. Diese Agenten sind für Anforderungen von Koordinatoragenten, die für Clientprogramme aktiv sind, oder für Subagenten, die für vorhandene Koordinatoragenten aktiv sind, verfügbar. Die Anzahl der verfügbaren Agenten ist vom Wert des Konfigurationsparameters *num_poolagents* des Datenbankmanagers abhängig.

Falls kein Agent im Bereitschaftsmodus vorhanden ist, wenn ein Agent angefordert wird, wird ein neuer Agent dynamisch erstellt. Da die Erstellung eines neuen Agenten einen bestimmten Systemaufwand erfordert, ist die CONNECT- und ATTACH-Leistung besser, wenn ein nicht zugeordneter Agent im Bereitschaftsmodus für einen Client aktiviert werden kann.

Wenn ein Subagent für eine Anwendung aktiv ist, ist er dieser Anwendung *zugeordnet*. Nach Beendigung der angeforderten Operationen kann er in den Agentenpool versetzt werden, bleibt jedoch der ursprünglichen Anwendung zugeordnet. Wenn die Anwendung eine weitere Operation anfordert, überprüft der Datenbankmanager zunächst den Agentenpool nach zugeordneten inaktiven Agenten, bevor er einen neuen Agenten erstellt.

Verwaltung von Datenbankagenten

Die meisten Anwendungen richten eine Eins-zu-eins-Beziehung zwischen der Zahl der verbundenen Anwendungen und der Zahl der Anwendungsanforderungen ein, die von der Datenbank verarbeitet werden können. Es kann jedoch sein, dass Sie aufgrund Ihrer Arbeitsumgebung eine Viele-zu-eins-Beziehung zwischen der Zahl der verbundenen Anwendungen und der Zahl der Anwendungsanforderungen benötigen, die verarbeitet werden können.

Die Möglichkeit, diese Faktoren separat zu steuern, wird durch Konfigurationsparameter des Datenbankmanagers bereitgestellt:

- Der Parameter *max_connections* gibt die zulässige Anzahl verbundener Anwendungen an.
- Der Parameter *max_coordagents* gibt die Anzahl von Anwendungsanforderungen an, die gleichzeitig verarbeitet werden können.

Der Verbindungskonzentrator ist aktiviert, wenn der Wert des Parameters *max_connections* größer als der Wert des Parameters *max_coordagents* ist.

Da jeder aktive Koordinatoragent globalen Ressourcenaufwand erfordert, steigt mit wachsender Zahl dieser Agenten auch die Wahrscheinlichkeit, dass die Obergrenze an verfügbaren globalen Datenbankressourcen erreicht wird. Um das Erreichen der Obergrenze der verfügbaren globalen Datenbankressourcen zu vermeiden, können Sie den Parameter *max_connections* auf einen höheren Wert als den Parameter *max_coordagents* setzen.

Beim Einstellen der Werte der Parameter *max_connections* und *max_coordagents* können Sie auch den Wert AUTOMATIC angeben. Es gibt zwei bestimmte Szenarios, in denen die Einstellung AUTOMATIC Vorteile bietet:

- Wenn Sie sicher sind, dass Ihr System alle Verbindungen verarbeiten kann, die möglicherweise benötigt werden, jedoch der Aufwand der genutzten globalen Ressourcen begrenzt werden soll (durch Begrenzen der Anzahl koordinierender Agenten), sollten Sie den Wert AUTOMATIC nur für den Parameter *max_connections* angeben. Wenn der Parameter *max_connections* auf AUTOMATIC mit einem Wert größer als *max_coordagents* gesetzt wird, bedeutet dies, dass der Verbin-

dungskonzentrator aktiviert wird, wobei eine beliebige Anzahl von Verbindungen zugelassen wird (solange ausreichend Systemressourcen verfügbar sind), die maximale Anzahl koordinierender Agenten jedoch begrenzt bleibt. Dies kann zur Steuerung von Hauptspeicher- und Plattenbeschränkungen durch Begrenzen der Anzahl gleichzeitig ausgeführter Anwendungen verwendet werden.

- Wenn Sie keine künstlichen Begrenzungen für Ihr System festlegen wollen, durch die Sie die maximale Anzahl von Verbindungen und koordinierenden Agenten beschränken, jedoch wissen, dass Ihr System eine Viele-zu-eins-Beziehung (zwischen verbundenen Anwendungen und verarbeiteten Anwendungsanforderungen) erfordert oder von einer solchen Beziehung profitieren würde, sollten Sie den Verbindungskonzentrator aktivieren und beide Parameter auf AUTOMATIC setzen. Wenn beide Parameter auf AUTOMATIC gesetzt sind, verwendet der Datenbankmanager die Werte, die Sie angeben, als Verhältnis, das die ideale Anzahl von koordinierenden Agenten und Verbindungen darstellt.

Beispiel

Betrachten Sie die folgende Parameterkonfiguration:

- Der Parameter *max_connections* ist mit einem Wert von 200 auf AUTOMATIC gesetzt.
- Der Parameter *max_coordagents* ist mit einem Wert von 100 auf AUTOMATIC gesetzt.

Das Verhältnis von *max_connections* zu *max_coordagents* beträgt 300 zu 100. Der Datenbankmanager zieht neue koordinierende Agenten bei Eintreffen von Verbindungen in Betracht, das heißt, die Konzentratorfunktion wird nur bei Bedarf angewendet. Die obigen Einstellungen werden wie folgt umgesetzt:

- Für die Verbindungen 1 bis 100 werden neue koordinierende Agenten erstellt.
- Für die Verbindungen 101 bis 300 werden keine neuen koordinierenden Agenten erstellt, sondern sie nutzten die 100 bereits erstellten Agenten gemeinsam.
- Für die Verbindungen 301 bis 400 werden neue koordinierende Agenten erstellt.
- Für die Verbindungen 401 bis 600 werden keine neuen koordinierenden Agenten erstellt, sondern sie nutzten die 200 bereits erstellten Agenten gemeinsam.
- Und so weiter...

Anmerkung: Dieses Beispiel geht von der Annahme aus, dass die verbundenen Anwendungen genügend Arbeit verursachen, um die Erstellung neuer koordinierender Agenten in jedem Schritt zu rechtfertigen. Wenn die verbundenen Anwendungen nach einer gewissen Zeit keine Arbeit mehr erledigen, werden die koordinierenden Agenten inaktiv und werden vielleicht beendet.

Wenn sich die Anzahl der Verbindungen reduziert, der Umfang der Arbeit, der von den verbleibenden Verbindungen verursacht wird, jedoch hoch ist, wird die Anzahl der koordinierenden Agenten möglicherweise nicht sofort verringert. Die Parameter *max_connections* und *max_coordagents* wirken sich nicht direkt auf den Zusammenschluss (Pooling) von Agenten oder die Beendigung von Agenten aus. Die normalen Regeln für die Beendigung von Agenten sind weiterhin gültig. Das heißt, dass die Anzahl von Verbindungen zu koordinierenden Agenten möglicherweise nicht exakt das von Ihnen angegebene Verhältnis darstellt. Agenten werden möglicherweise zur Wiederverwendung an den Agentenpool zurückgegeben, bevor sie beendet werden.

Wenn ein feinerer Grad an Steuerung erforderlich ist, wird ein einfacheres Verhältnis empfohlen. Zum Beispiel kann das oben gezeigte Verhältnis von 300 zu 100 auf

ein Verhältnis von 3 zu 1 reduziert werden. Wenn der Parameter *max_connections* auf 3 (AUTOMATIC) und der Parameter *max_coordagents* auf 1 (AUTOMATIC) gesetzt wird, wird für je drei Verbindungen die Erstellung eines koordinierenden Agenten zugelassen.

Verbesserungen des Verbindungskonzentrators für Clientverbindungen

Für Internetanwendungen mit vielen relativ kurzfristigen Verbindungen oder ähnliche Arten von Anwendungen verbessert der Verbindungskonzentrator die Leistung, indem er es ermöglicht, wesentlich mehr Clientverbindungen effizient zu verarbeiten. Darüber hinaus verringert er die Speicherauslastung für jede Verbindung und senkt die Anzahl von Kontextumschaltungen.

Anmerkung: Der Verbindungskonzentrator ist aktiviert, wenn der Wert des Parameters *max_connections* größer als der Wert des Parameters *max_coordagents* ist.

In einer Umgebung, die viele gleichzeitige Benutzerverbindungen erfordert, können Sie den Verbindungskonzentrator aktivieren, um die Systemressourcen effizienter zu nutzen. Diese Funktion bietet Vorteile, die bisher nur im Verbindungspooling von DB2 Connect zu finden waren. Sowohl das Verbindungspooling als auch der Verbindungskonzentrator werden im DB2 Connect Benutzerhandbuch beschrieben. Nach Herstellung der ersten Verbindung verringert der Verbindungskonzentrator die Zeit für die Verbindung zu einem Host. Wenn die Trennung einer Verbindung zu einem Host angefordert wird, wird die eingehende Verbindung gelöscht, während die abgehende Verbindung zum Host in einem Pool erhalten bleibt. Wenn eine neue Anforderung zur Herstellung einer Verbindung zu dem Host erfolgt, versucht der Datenbankmanager eine vorhandene abgehende Verbindung aus dem Pool wiederzuverwenden.

Anmerkung: Wenn Anwendungen mit einem Verbindungspooling oder mit dem Verbindungskonzentrator arbeiten, erreichen Sie die beste Leistung, indem Sie die Parameter optimieren, die die Größe des Datenblocks steuern, der im Cache zwischengespeichert wird. Weitere Informationen finden Sie im DB2 Connect Benutzerhandbuch.

Bei Nutzung des Verbindungspoolings ist DB2 Connect auf eingehende und abgehende TCP/IP-Verbindungen beschränkt.

Verwendungsbeispiele

Beispiel 1:

Betrachten Sie eine ESE-Umgebung mit einer einzelnen Datenbankpartition, in der im Durchschnitt 1000 Benutzer mit der Datenbank verbunden sind. Manchmal kann die Anzahl der verbundenen Benutzer höher liegen, daher sollte keine Einschränkung auf 1000 festgelegt werden. Die Anzahl der gleichzeitig ausgeführten Transaktionen kann ca. 200, jedoch nie über 250 betragen. Transaktionen sind kurz.

Für diese Auslastung könnten Sie die folgenden Konfigurationsparameter des Datenbankmanagers definieren:

- Der Parameter *max_coordagents* wird auf 250 gesetzt, um die maximale Anzahl gleichzeitiger Transaktionen zu unterstützen.
- Der Parameter *max_connections* wird auf AUTOMATIC mit dem Wert 1000 gesetzt, um die Unterstützung für eine beliebige Anzahl von Verbindungen sicherzustellen (in diesem Beispiel reicht jede Anzahl über 250 aus, um den Verbindungskonzentrator zu aktivieren).

- Der Parameter *num_poolagents* wird auf den Standardwert gesetzt, wodurch sichergestellt werden sollte, dass Datenbankagenten unter geringem Aufwand zur Erstellung neuer verfügbar sind, um eingehende Clientanforderungen zu bedienen.

Beispiel 2:

In einem System, das dem aus Beispiel 1 ähnlich ist, in dem anscheinend eine unbegrenzte Anzahl von Verbindungen zugelassen wird, ist es möglich, die Anzahl der koordinierenden Agenten abhängig von der Anzahl Verbindungen ebenfalls anwachsen zu lassen. Nehmen Sie für dieses Beispiel an, dass im Durchschnitt 1000 Benutzer verbunden sind, wobei etwa 250 Transaktionen gleichzeitig ausgeführt werden. Der Spitzenwert für die Anzahl von Benutzern ist jedoch manchmal unvorhersehbar. Gelegentlich könnten 2000 Benutzer verbunden sein, wobei zu erwarten ist, dass wahrscheinlich von diesen im Durchschnitt 500 Arbeit ausführen. Es sollten zumeist keine 500 koordinierenden Agenten zugelassen werden, da normalerweise nur 1000 Benutzer verbunden sind und 250 koordinierende Agenten für diese normalerweise ausreichen.

Für diese Auslastung könnten Sie den Konfiguration des Datenbankmanagers wie folgt aktualisieren:

```
db2 update dbm cfg using MAX_COORDAGENTS 250 AUTOMATIC
db2 update dbm cfg using MAX_CONNECTIONS 1000 AUTOMATIC
```

Dies bedeutet, dass, wenn die Anzahl von Verbindungen über 1000 steigt, zusätzliche koordinierende Agenten nach Bedarf erstellt werden, wobei die maximale Anzahl durch die Gesamtanzahl der Verbindungen festgelegt wird. Da beide Parameter einen Wert angeben und beide auf AUTOMATIC gesetzt sind, behält der Datenbankmanager bei steigender Auslastung das Verhältnis von koordinierenden Agenten zu Verbindungen annähernd bei.

Beispiel 3:

In einem System, für das Sie den Verbindungskonzentrator nicht aktivieren, jedoch die Anzahl verbundener Benutzer, zum Beispiel auf 250 gleichzeitig verbundene Benutzer, begrenzen wollen, stellen Sie die Konfigurationsparameter des Datenbankmanagers wie folgt ein:

- *max_connections* auf 250
- *max_coordagents* auf 250

Beispiel 4:

In einem System, für das Sie den Verbindungskonzentrator nicht aktivieren wollen und auch die Anzahl der verbundenen Benutzer nicht begrenzen wollen, stellen Sie die Konfigurationsparameter des Datenbankmanagers wie folgt ein:

```
db2 update dbm cfg using MAX_COORDAGENTS AUTOMATIC
db2 update dbm cfg using MAX_CONNECTIONS AUTOMATIC
```

Agenten in einer partitionierten Datenbank

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität verfügt jede Datenbankpartition (d. h. jeder Datenbankserver oder Knoten) über einen eigenen Pool von Agenten, aus dem Subagenten entnommen werden. Durch die Verwendung dieses Pools müssen Subagenten nicht jedes Mal erstellt und wieder gelöscht werden, wenn ein Subagent benötigt wird oder seine Arbeit beendet hat. Die Subagenten können als zugeord-

nete Agenten im Pool bleiben und vom Datenbankmanager für neue Anforderungen von der Anwendung, der sie zugeordnet sind, oder von neuen Anwendungen verwendet werden.

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität besteht eine enge Beziehung zwischen dem Einfluss auf die Leistung und den Speicherbedarf innerhalb des Systems und der Optimierung des Agentenpools:

- Der Konfigurationsparameter des Datenbankmanagers für die Größe des Agentenpools (*num_poolagents*) betrifft die Gesamtanzahl von Agenten und Subagenten, die Anwendungen in einer Datenbankpartition (auch als Knoten bezeichnet) zugeordnet bleiben können. Wenn die Poolgröße zu klein ist und der Pool voll ist, wird ein Subagent aus der Zuordnung mit der Anwendung, für die er aktiv ist, gelöst und beendet. Da Subagenten ständig erstellt und Anwendungen erneut zugeordnet werden müssen, sinkt die Leistung.

  Standardmäßig ist der Konfigurationsparameter *num_poolagents* auf AUTOMATIC mit dem Wert 100 gesetzt. Wenn dieser Parameter auf AUTOMATIC gesetzt ist, verwaltet der Datenbankmanager automatisch die Anzahl inaktiver Agenten, die in einem Pool zusammengefasst werden sollen.

  Außerdem könnte bereits eine Anwendung den Pool mit zugeordneten Subagenten füllen, wenn der Wert für den Parameter *num_poolagents* zu klein ist. Wenn nun eine andere Anwendung einen neuen Subagenten benötigt und im zugeordneten Pool über keine Agenten verfügt, stoppt sie Subagenten aus den Agentenpools anderer Anwendungen und startet sie erneut. Dieses Verhalten stellt sicher, dass Ressourcen vollständig belegt werden.
- Wägen Sie die Vor- und Nachteile zu weniger Agenten gegen den Ressourcenbedarf zu vieler Agenten ab, die zu einem gegebenen Zeitpunkt aktiv sein können.

  Wenn der Wert des Parameters *num_poolagents* zum Beispiel zu groß ist, werden zugeordnete Subagenten lange Zeit ungenutzt im Pool behalten und belegen Datenbankmanagerressourcen, die für andere Tasks nicht verfügbar sind.

  Anmerkung: Wenn der Verbindungskonzentrator aktiviert ist, spiegelt die Anzahl von Agenten, die durch den Parameter *num_poolagents* angegeben wird, nicht unbedingt die exakte Anzahl von Agenten wider, die sich zu einem bestimmten Zeitpunkt inaktiv im Pool befinden können. Die Anzahl von Agenten kann zeitweise überschritten werden, um Perioden mit höherer Auslastung aufzufangen.

Andere asynchrone Prozesse und Threads

Neben den Datenbankagenten werden auch noch andere asynchrone Aktivitäten des Datenbankmanagers als eigene Prozesse bzw. Threads ausgeführt. Dazu gehören:

- E/A-Server oder E/A-Vorablesefunktionen der Datenbank
- Asynchrone Seitenlöschfunktionen der Datenbank
- Protokollfunktionen der Datenbank
- Deadlock-Detektoren für Datenbanken
- Übertragungs- und IPC-Empfangsprozesse (Listeners)
- Funktionen zum Neuausgleich von Daten in Tabellenbereichscontainern

Informationen des Datenbanksystemmonitors

Der DB2-Datenbankmanager pflegt Daten über den Betrieb, die Leistung und die Anwendungen, die ihn verwenden. Diese Daten werden während des Betriebs des Datenbankmanagers verwaltet und können wichtige Informationen über die Leistung und zur Fehlerbehebung liefern. Zum Beispiel erhalten Sie Anhaltspunkte über:

- Die Anzahl von Anwendungen, die mit einer Datenbank verbunden sind, ihren Status und welche SQL- und XQuery-Anweisungen (falls überhaupt) von jeder Anwendung ausgeführt werden.
- Informationen, die zeigen, wie gut der Datenbankmanager und die Datenbank konfiguriert sind, und bei der Optimierung helfen.
- Aufgetretene Deadlocks für eine angegebene Datenbank, beteiligte Anwendungen und welche Sperren in der Konkurrenzsituation waren.
- Die Liste von Sperren, die von einer Anwendungen oder Datenbank aktiviert wurden. Wenn die Anwendung die Verarbeitung nicht fortsetzen kann, weil sie auf eine Sperre wartet, werden weitere Informationen zur Sperre, einschließlich der Anwendung, die sie aktiviert hat, bereitgestellt.

Da die Erfassung einiger dieser Daten mit Systemaufwand für den Betrieb von DB2 verbunden ist, sind **Monitorschalter** verfügbar, mit denen gesteuert werden kann, welche Informationen erfasst werden. Zum expliziten Einstellen der Monitorschalter wird der Befehl UPDATE MONITOR SWITCHES oder die API sqlmon() verwendet. (Sie benötigen hierzu die Berechtigung SYSADM, SYSCTRL, SYSMAINT oder SYSMON.)

Sie können auf die vom Datenbankmanager verwalteten Daten zugreifen, indem Sie entweder eine Momentaufnahme (Snapshot) erstellen oder einen Ereignismonitor verwenden.

Erstellen einer Momentaufnahme

Mit einer der drei folgenden Methoden können Sie eine Momentaufnahme erstellen:

- Verwenden des Befehls GET SNAPSHOT über die Befehlszeile
- Schreiben einer eigenen Anwendung mit dem API-Aufruf db2GetSnapshot()
- Verwenden der Tabellenfunktionen zur Erstellung von Momentaufnahmen, um Monitordaten über einen bestimmten Bereich des Datenbanksystems zurückzugeben

Verwenden eines Ereignismonitors

Ein Ereignismonitor (Event Monitor) erfasst Systemmonitordaten nach Eintreten bestimmter Ereignisse, wie zum Beispiel das Ende einer Transaktion, das Ende einer Anweisung oder die Erkennung eines Deadlocks. Diese Informationen können in Dateien oder eine benannte Pipe geschrieben werden.

Ein Ereignismonitor wird wie folgt verwendet:

1. Erstellen Sie die Definition des Ereignismonitors mithilfe der Steuerzentrale oder der SQL-Anweisung CREATE EVENT MONITOR. Diese Anweisung speichert die Definition in den Systemkatalogen der Datenbank.
2. Aktivieren Sie den Ereignismonitor über die Steuerzentrale oder mit der folgenden SQL-Anweisung:

```
SET EVENT MONITOR ergname STATE 1
```

Wenn die Ergebnisse in eine benannte Pipe geschrieben werden, starten Sie die Anwendung, die die Daten aus der benannten Pipe liest, bevor Sie den Ereignismonitor aktivieren. Sie können entweder eine eigene Anwendung zum Lesen schreiben oder den Befehl db2evmon verwenden. Sobald der Ereignismonitor aktiv ist und mit dem Schreiben von Ereignissen in die Pipe beginnt, liest db2evmon die Ereignisse, wie sie generiert werden, und schreibt Sie in die Standardausgabe.

3. Lesen Sie die Tracedaten. Wenn Sie einen Ereignismonitor verwenden, der in eine Datei schreibt, können Sie die binären Tracedaten, die er erstellt, mit einer der folgenden Methoden anzeigen:
   - Verwenden Sie das Tool db2evmon zum Formatieren der Tracedaten für die Standardausgabe.
   - Klicken Sie das Symbol für **Event Analyzer** in der Steuerzentrale unter einem Windows-basierten Betriebssystem an, um eine grafische Schnittstelle zum Anzeigen der Tracedaten, zum Suchen nach Schlüsselwörtern und zum Herausfiltern nicht erwünschter Daten aufzurufen.

     Anmerkung: Wenn das Datenbanksystem, das Sie überwachen, nicht auf derselben Maschine wie die Steuerzentrale ausgeführt wird, müssen Sie die Ereignismonitordatei auf dieselbe Maschine wie die Steuerzentrale kopieren, damit die Tracedaten angezeigt werden können. Sie können die Datei alternativ auch in ein gemeinsam benutztes Dateisystem stellen, auf das beide Maschinen zugreifen können.

Leistungseinfluss bei der Verwendung eines Deadlock-Ereignismonitors

Wenn ein Deadlock-Ereignismonitor mit der Option HISTORY aktiv ist, wird die allgemeine Leistung eines DB2-Datenbanksystems wie folgt beeinflusst:

- Der Speicher, der im Paketcache für in den Cache gestellte dynamische SQL- und XQuery-Anweisungen verwendet wird, die im Anweisungsprotokoll aufgelistet werden, wird erst freigegeben, wenn das bestimmte Anweisungsprotokoll nicht mehr benötigt wird (d. h., die aktuelle UOW (Unit of Work, Arbeitseinheit) wird beendet). Dadurch kann die Größe des Paketcaches aufgrund der erhöhten Speicherbelegung im Cache, der nicht freigegeben werden kann, zunehmen.
- Das Kopieren von Anweisungsinformationen in die Liste des Anweisungsprotokolls hat einen geringfügigen Einfluss auf die Systemleistung.
- In jeder Datenbankpartition wird der DB2-Systemmonitorzwischenspeicher vermehrt genutzt, um eine Protokoll-Liste der Anweisungen für jede aktive Anwendung in der Datenbankpartition zu speichern. Die Zunahme hängt von der Anzahl der Anweisungen ab, die in jeder UOW von jeder Anwendung ausgeführt werden. Im Folgenden sehen Sie eine Berechnung für den Monitorzwischenspeicher:

```
Wenn ein Ereignismonitor den Typ DEADLOCK aufweist
und die Option WITH DETAILS HISTORY aktiviert ist,
fügen Sie X*100 Byte mal die maximale Anzahl an gleichzeitig
ablaufenden Anwendungen hinzu, die aktiv sein sollen;
dabei ist X die erwartete maximale Anzahl an Anweisungen in
der UOW Ihrer Anwendung.
Wenn der Ereignismonitor den Typ DEADLOCK aufweist
und die Option WITH DETAILS HISTORY VALUES aktiviert ist,
fügen Sie ebenfalls X*Y Byte mal die maximale Anzahl an gleichzeitig
ablaufenden Anwendungen hinzu, die aktiv sein sollen;
dabei ist Y die Summe der erwarteten maximalen Größe der
in Ihre SQL- und XQuery-Anweisungen gebundenen Parameterwerte.
```

Wenn ein Deadlock-Ereignismonitor mit der Option VALUES aktiv ist, wird die allgemeine Leistung eines DB2-Datenbanksystems wie folgt beeinflusst (samt der zuvor aufgelisteten Beeinflussungsmöglichkeiten für die Option HISTORY):

- Das Kopieren von Anweisungsinformationen in die Liste des Anweisungsprotokolls hat einen äußerst geringfügigen Einfluss auf die Systemleistung.
- In jeder Datenbankpartition wird der DB2-Systemmonitorzwischenspeicher vermehrt genutzt, damit eine Protokoll-Liste zu den Anweisungen für jede aktive Anwendung in der Datenbankpartition aufbewahrt wird. Die Zunahme hängt von der Anzahl der pro Anweisung verwendeten Datenwerte sowie der Anzahl der Anweisungen ab, die in jeder UOW von jeder Anwendung ausgeführt werden.
- Der Datenbankmanager verwaltet eine Zusatzkopie der Datenwerte, die, in Abhängigkeit von der Größe und Anzahl der Variablen, die Leistung beeinflussen können.

Die Speicherauswirkungen auf den DB2-Systemmonitorzwischenspeicher kann dann von nennenswerter Bedeutung sein, wenn die Optionen HISTORY und VALUES für den Deadlock-Ereignismonitor angegeben werden. Um die Auswirkungen so gering wie möglich zu halten, verwenden Sie diese Optionen nur, wenn sie wirklich erforderlich sind. Eine andere Möglichkeit, die Auswirkungen zu verringern, besteht darin, die konfigurierte Größe des DB2-Systemmonitorzwischenspeichers vor der Aktivierung des Ereignismonitors in allen Datenbankpartitionen zu vergrößern.

Wenn ein tatsächlicher Deadlock auftritt und ein Deadlock-Ereignismonitor aktiv ist, wird die Systemleistung durch die Generierung der Ereignismonitordatensätze beeinflusst. Der Grad und die Dauer der Beeinflussung hängt von der Anzahl der Anwendungen und Datenbankpartitionen, die am Deadlock beteiligt sind, sowie von der Anzahl der Anweisungen und Datenwerte in den relevanten Anweisungsprotokoll-Listen ab.

Effiziente SELECT-Anweisungen

Da SQL eine flexible Abfragesprache höherer Ebene ist, haben Sie die Möglichkeit, mehrere verschiedene SELECT-Anweisungen zum Abrufen der gleichen Daten zu schreiben. Die Leistung kann jedoch für die verschiedenen Formen der Anweisung sowie für die verschiedenen Optimierungsklassen unterschiedlich sein.

Berücksichtigen die folgenden Richtlinien für SELECT-Anweisungen:

- Geben Sie nur Spalten an, die Sie benötigen. Obwohl es einfacher ist, alle Spalten mit einem Stern (*) anzugeben, führt dies zu überflüssigem Verarbeitungsaufwand und zur Rückgabe unnötiger Spalten.
- Verwenden Sie Vergleichselemente, die die Antwortmenge auf nur die Zeilen einschränken, die Sie benötigen.
- Wenn die Anzahl der Zeilen, die Sie benötigen, bedeutend kleiner ist als die Gesamtanzahl der Zeilen, die abgerufen werden könnten, geben Sie die Klausel OPTIMIZE FOR an. Diese Klausel wirkt sich sowohl auf die Auswahl der Zugriffspläne als auch auf die Anzahl der im Kommunikationspuffer geblockten Zeilen aus.
- Wenn die Anzahl der abzurufenden Zeilen klein ist, geben Sie nur die Klausel OPTIMIZE FOR *k* ROWS an. Die Klausel FETCH FIRST n ROWS ONLY ist nicht erforderlich. Wenn *n* jedoch groß ist und Sie die ersten *k* Zeilen schnell abrufen wollen und eine mögliche Verzögerung für die nachfolgenden *k* Zeilen in Kauf

nehmen, geben Sie beide Klauseln an. Die Größe der Kommunikationspuffer wird in Abhängigkeit vom kleineren der Werte für *n* und *k* festgelegt. Das folgende Beispiel zeigt beide Klauseln:

```
SELECT EMPNAME, SALARY FROM EMPLOYEE
   ORDER BY SALARY DESC
   FETCH FIRST 100 ROWS ONLY
   OPTIMIZE FOR 20 ROWS
```

- Zur Verwendung der Zeilenblockung geben Sie die Klausel FOR READ ONLY oder FOR FETCH ONLY an, um die Leistung zu verbessern. Dadurch wird außerdem der gleichzeitige Zugriff verbessert, weil für die abgerufenen Spalten zu keiner Zeit exklusive Sperren aktiviert sind. Zusätzliches Umschreiben der Abfrage kann ebenfalls stattfinden. Durch die Angabe der Klausel FOR READ ONLY oder FOR FETCH ONLY sowie durch die BIND-Option BLOCKING ALL lässt sich die Leistung von Abfragen für Kurznamen in einem föderierten System in ähnlicher Weise verbessern.
- Geben Sie für Cursor, die durch positionierte Aktualisierungen aktualisiert werden, die Klausel FOR UPDATE OF an, um dem Datenbankmanager zu ermöglichen, gleich zu Anfang angemessenere Sperrstufen zu wählen und potenzielle Deadlocks zu vermeiden. Beachten Sie, dass FOR UPDATE-Cursor die Zeilenblockung nicht nutzen können.
- Für Cursor, die durch Aktualisierungen mit Suche aktualisiert werden, können Sie Deadlocks vermeiden und trotzdem die Zeilenblockung ermöglichen, indem Sie Sperren des Modus U durch die Klauseln FOR READ ONLY und USE AND KEEP UPDATE LOCKS für die betroffenen Zeilen erzwingen.
- Vermeiden Sie nach Möglichkeit Umsetzungen numerischer Datentypen. Beim Vergleichen von Werten ist es in der Regel effizienter, zwei Werte mit demselben Datentyp miteinander zu vergleichen. Wenn Umwandlungen erforderlich sind, können Ungenauigkeiten aufgrund begrenzter Präzision und Leistungseinbußen aufgrund von Umwandlungen, die zur Laufzeit ausgeführt werden müssen, die Folge sein.

  Verwenden Sie, falls möglich, die folgenden Datentypen:

  – Zeichen (CHAR) anstatt Zeichen variierender Länge (VARCHAR) für kurze Spalten
  – Ganze Zahlen (INTEGER) anstatt Gleitkommazahlen (FLOAT) oder Dezimalzahlen (DECIMAL)
  – Datum/Uhrzeit (Datetime) anstatt Zeichen
  – Numerische Typen anstatt Zeichen
- Lassen Sie zur Verringerung der Wahrscheinlichkeit von Sortieroperationen Klauseln oder Operationen wie DISTINCT oder ORDER BY weg, wenn solche Operationen nicht erforderlich sind.
- Wählen Sie eine einzige Zeile aus, wenn Sie das Vorhandensein von Zeilen prüfen wollen. Öffnen Sie dazu einen Cursor und rufen Sie eine Zeile ab (FETCH) oder führen Sie eine Auswahl für eine Zeile (SELECT INTO) durch. Vergessen Sie nicht, auf den SQLCODE-Wert -811 zu prüfen, wenn mehr als eine Zeile gefunden wird.

  Sofern Sie nicht wissen, dass die Tabelle sehr klein ist, vermeiden Sie die folgende Anweisung zur Prüfung auf einen Nichtnullwert:

  ```
  SELECT COUNT(*) FROM  TABLENAME
  ```

  Das Zählen aller Zeilen wirkt sich bei großen Tabellen negativ auf die Leistung aus.

- Wenn das Aktualisierungsaufkommen gering ist und die Tabellen umfangreich sind, sollten Sie Indizes für Spalten definieren, die häufig in Vergleichselementen verwendet werden.
- Ziehen Sie die Verwendung einer IN-Liste in Betracht, wenn dieselbe Spalte in mehreren PREDICATE-Klauseln erscheint. Bei großen IN-Listen, die mit Hostvariablen verwendet werden, können Verarbeitungsschleifen für Untergruppen der Hostvariablen die Leistung verbessern.

Die folgenden Vorschläge gelten insbesondere für SELECT-Anweisungen, die auf mehrere Tabellen zugreifen.

- Verwenden Sie Joinvergleichselemente für den Join von Tabellen. Ein Joinvergleichselement ist ein Vergleich zwischen zwei Spalten verschiedener Tabellen in einem Join.
- Definieren Sie Indizes für die Spalten in dem Joinvergleichselement, um eine effizientere Verarbeitung des Joins zu ermöglichen. Indizes sind außerdem für UPDATE- und DELETE-Anweisungen förderlich, die SELECT-Anweisungen enthalten, die auf mehrere Tabellen zugreifen.
- Vermeiden Sie nach Möglichkeit Ausdrücke oder Klauseln OR mit Joinvergleichselementen, weil der Datenbankmanager einige Jointechniken nicht nutzen kann. Dies kann dazu führen, dass nicht die effizienteste Joinmethode gewählt wird.
- In einer Umgebung mit partitionierten Datenbanken stellen Sie nach Möglichkeit sicher, dass beide Tabellen, die verknüpft werden, über die Joinspalte partitioniert sind.

Kapitel 15. Das Dienstprogramm Governor

Das Dienstprogramm Governor kann das Verhalten von Anwendungen überwachen, die an der Datenbank ausgeführt werden, und abhängig von den Regeln, die Sie in der Konfigurationsdatei von Governor angeben, bestimmte Funktionsweisen ändern.

Eine Governor-Instanz besteht aus einem Front-End-Dienstprogramm und mindestens einem Dämon. Jede Governor-Instanz, die Sie starten, ist für eine Instanz des Datenbankmanagers spezifisch. Wenn Sie das Dienstprogramm Governor starten, wird standardmäßig ein Governor-Dämon in jeder Datenbankpartition einer partitionierten Datenbank gestartet. Sie können jedoch angeben, dass ein Dämon in einer einzelnen Datenbankpartition gestartet werden soll, die Sie überwachen wollen.

Anmerkung: Wenn Governor aktiv ist, können die Momentaufnahmenanforderungen des Dienstprogramms die Leistung des Datenbankmanagers beeinträchtigen. Zur Verbesserung der Leistung können Sie das Aktivierungsintervall (wake-up) von Governor vergrößern, um den CPU-Bedarf des Dienstprogramms zu verringern.

Jeder Governor-Dämon sammelt Informationen über die Anwendungen, die an einer Datenbank ausgeführt werden. Anschließend prüft er diese Informationen anhand der Regeln, die Sie in der Konfigurationsdatei von Governor für diese Datenbank angeben.

Governor verwaltet Anwendungstransaktionen, wie es durch die Regeln in der Konfigurationsdatei angegeben ist. Zum Beispiel könnte die Anwendung einer Regel darauf hinweisen, dass eine Anwendung eine bestimmte Ressource übermäßig beansprucht. Eine Regel gibt die durchzuführende Aktion an, wie zum Beispiel das Ändern der Priorität der Anwendung oder das zwangsweise Trennen der Anwendung von der Datenbank.

Wenn die einer Regel zugeordnete Aktion die Priorität der Anwendung ändert, ändert Governor die Priorität von Agenten in der Datenbankpartition, in der die Ressourcenverletzung aufgetreten ist. Wenn in einer partitionierten Datenbank die Anwendung zwangsweise von der Datenbank getrennt wird, findet die Aktion auch dann statt, wenn der Dämon, der die Verletzung festgestellt hat, auf dem Koordinatorknoten der Anwendung ausgeführt wird.

Governor protokolliert alle von ihm durchgeführten Aktionen. Zur Prüfung der Aktionen können Sie die Protokolldateien abfragen.

Starten und Stoppen von Governor

Das Dienstprogramm Governor überwacht Anwendungen, die eine Verbindung zu einer Datenbank herstellen, und ändert ihre Funktionsweise gemäß den Regeln, die Sie in einer Konfigurationsdatei von Governor für diese Datenbank angeben.

Bevor Sie Governor starten, müssen Sie die Konfigurationsdatei erstellen.

Zum Starten oder Stoppen von Governor benötigen Sie die Berechtigung *sysadm* oder *sysctrl*.

Gehen Sie wie folgt vor, um Governor zu starten oder zu stoppen:

1. Führen Sie zum Starten von Governor den Befehl *db2gov* in der DB2-Befehlszeile aus. Geben Sie die folgenden erforderlichen Parameter an:
   - START *datenbankname*

     Der von Ihnen angegebene Datenbankname muss mit dem Namen der Datenbank in der Konfigurationsdatei übereinstimmen, die Sie angeben. Stimmen die Namen nicht überein, wird ein Fehler zurückgegeben. Beachten Sie, dass bei der Ausführung von Governor für mehrere Datenbanken für jede Datenbank Dämonen gestartet werden.
   - *konfig_dateiname*

     Der Name der Konfigurationsdatei von Governor für diese Datenbank. Wenn sich die Datei nicht an der Standardposition befindet, d. h. im Verzeichnis `sqllib`, müssen Sie außer dem Dateinamen auch den Pfad mit angeben.
   - *protokolldateiname*

     Der Basisname der Protokolldatei für diesen Governor. In einer partitionierten Datenbank wird die Nummer der Datenbankpartition für jede Datenbankpartition hinzugefügt, in der ein Dämon für diese Instanz von Governor ausgeführt wird.

   Zum Starten von Governor in einer einzelnen Datenbankpartition für eine partitionierte Datenbank fügen Sie die Option *nodenum* hinzu.

   Um zum Beispiel Governor für eine Datenbank mit dem Namen *sales* nur auf Knoten 3 einer partitionierten Datenbank mit einer Konfigurationsdatei mit dem Namen *salescfg* und einer Protokolldatei mit dem Namen *saleslog* zu starten, geben Sie den folgenden Befehl ein:

   ```
   db2gov START sales nodenum 3 salescfg saleslog
   ```

   Wenn Governor in allen Datenbankpartitionen der Datenbank *sales* gestartet werden soll, geben Sie den folgenden Befehl ein:

   ```
   db2gov START sales  salescfg saleslog
   ```
2. Zum Stoppen von Governor geben Sie den Befehl *db2gov* mit der Option STOP ein.

   Wenn Governor zum Beispiel in allen Datenbankpartitionen der Datenbank *sales* gestoppt werden soll, geben Sie den folgenden Befehl ein:

   ```
   db2gov STOP sales
   ```

   Soll Governor nur in Datenbankpartition 3 gestoppt werden, geben Sie den folgenden Befehl ein:

   ```
   db2gov START sales nodenum 3
   ```

Der Governor-Dämon

Wenn der Governor-Dämon gestartet wird, d. h. wenn entweder Sie das Dienstprogramm `db2gov` ausführen oder Governor aktiv wird (wake-up), führt er die folgende wiederkehrende Abfolge von Operationen aus.

1. Er prüft, ob seine Governor-Konfigurationsdatei geändert bzw. noch nicht gelesen wurde. Trifft eine der beiden Bedingungen zu, liest der Dämon die Regeln in der Datei. Dadurch können Sie das Verhalten des Governor-Dämons ändern, während er aktiv ist.
2. Er fordert Momentaufnahmeninformationen zur Ressourcennutzungsstatistik für jede Anwendung und jeden Agenten an, die bzw. der in der Datenbank arbeitet.

Anmerkung: Auf einigen Plattformen sind die CPU-Statistikdaten auf dem DB2-Monitor nicht verfügbar. In diesem Fall stehen die Abrechnungsregel (account) und die CPU-Begrenzung (cpu) nicht zur Verfügung.

3. Er überprüft die Statistik für jede einzelne Anwendung anhand der Regeln in der Governor-Konfigurationsdatei. Wenn eine Regel auf eine Anwendung zutrifft, führt Governor die angegebene Aktion aus.

   Anmerkung: Der Governor vergleicht aufgelaufene Informationen mit den in der Konfigurationsdatei definierten Werten. Dies bedeutet: Falls die Konfigurationsdatei mit neuen Werten aktualisiert wird, die eine Anwendung möglicherweise bereits überschritten hat, werden die Governor-Regeln, die für die betreffende Überschreitung gelten, im nächsten Governor-Intervall unverzüglich auf die Anwendung angewandt.
4. Er schreibt für jede ausgeführte Aktion einen Datensatz in die Governor-Protokolldatei.

Anmerkung: Governor kann nicht zur Anpassung der Agentenprioritäten verwendet werden, wenn der Konfigurationsparameter *agentpri* des Datenbankmanagers einen anderen Wert als den Systemstandardwert enthält.

Wenn Governor seine Operationen abgeschlossen hat, wird er für die in der Konfigurationsdatei angegebene Zeitdauer inaktiviert. Nach Ablauf dieses Intervalls wird Governor wieder aktiviert (wake-up) und führt dieselbe Abfolge von Operationen erneut aus.

Trifft Governor auf einen Fehler oder ein Stoppsignal, führt er vor der Beendigung eine Bereinigung durch. Bei der Bereinigung werden mithilfe einer Liste von Anwendungen, deren Prioritäten geändert wurden, die Prioritäten aller Anwendungsagenten zurückgesetzt. Anschließend werden die Prioritäten derjenigen Agenten zurückgesetzt, die nicht mehr für eine Anwendung arbeiten. Dadurch wird sichergestellt, dass keine Agenten nach Beendigung von Governor mit anderen als den Standardprioritäten aktiv bleiben. Falls ein Fehler auftritt, schreibt Governor eine Nachricht in das Benachrichtigungsprotokoll für die Verwaltung, um seine abnormale Beendigung anzuzeigen.

Anmerkung: Obwohl der Governor-Dämon keine Datenbankanwendung ist und daher auch keine Verbindung (Connect) zur Datenbank unterhält, hat er dennoch eine Instanzverbindung (Attach). Da er Anforderungen für Momentaufnahmen absetzen kann, kann der Governor-Dämon erkennen, wenn der Datenbankmanager beendet wird.

Konfigurieren von Governor

Zum Konfigurieren von Governor erstellen Sie eine Konfigurationsdatei, in der die Datenbank, die von einer Governor-Instanz überwacht wird, und die Verwaltung von Abfragen festgelegt werden.

Die Konfigurationsdatei besteht aus einer Menge von Regeln. Die ersten drei Regeln geben die zu überwachende Datenbank, das Intervall, in dem Protokollsätze zu schreiben sind, und das Intervall an, in dem der Governor-Dämon zur Überwachung aktiv wird. Die übrigen Regeln geben an, wie der Datenbankserver zu überwachen ist und welche Aktionen unter bestimmten Umständen auszuführen sind.

Gehen Sie wie folgt vor, um eine Governor-Konfigurationsdatei zu erstellen:

1. Erstellen Sie in einem Verzeichnis, das von allen Datenbankpartitionen aus zugänglich ist bzw. für sie angehängt ist, eine ASCII-Datei mit einem beschreibenden Namen. Zum Beispiel könnte eine Konfigurationsdatei für eine Governor-Instanz, die eine Datenbank **sales** überwacht, den Namen *govcfgsales* haben.
2. Öffnen Sie die Datei in einem beliebigen Texteditor und geben Sie Konfigurationsinformationen und Aktionsbedingungen ein.

   Schließen Sie jede Regel mit einem Semikolon (;) ab. Die folgenden Konfigurationsdaten werden empfohlen:

   - dbname: Der Name oder Aliasname der zu überwachenden Datenbank.
   - account: Die Zahl Minuten, nach denen die Governor-Instanz Statistikdaten zur CPU-Nutzung in die zugehörige Protokolldatei schreibt.
   - interval: Die Zahl Sekunden, nach denen der Governor-Dämon aktiv wird, um die Überwachungsfunktion auszuführen. Wenn Sie kein Intervall angeben, wird der Standardwert von 120 Sekunden verwendet.

   Zum Beispiel könnten die ersten drei Regeln in der Konfigurationsdatei wie folgt aussehen:

   ```
   { Einmal pro Sekunde aktiv werden, Datenbankname lautet sales,
    Abrechnungsdatensätze alle 30 Minuten }
   interval 1; dbname sales; account 30;
   ```

   Fügen Sie Regeln hinzu, welche die zu überwachenden Bedingungen und die Aktion angeben, die auszuführen ist, wenn die jeweilige Regel als wahr ausgewertet wird. Sie könnten zum Beispiel wie folgt eine Regel hinzufügen, welche die Zeit, die eine UOW (Unit of Work, Arbeitseinheit) ausgeführt werden kann, bevor die Anwendung zwangsweise von der Datenbank getrennt wird, auf eine Stunde begrenzt:

   ```
   setlimit uowtime 3600 action force;
   ```
3. Sichern Sie die Datei.

Die Konfigurationsdatei von Governor

Beim Starten von Governor geben Sie die Konfigurationsdatei an, die die Regeln zum Überprüfen der in einer Datenbank aktiven Anwendungen enthält. Governor wertet jede Regel aus und führt die angegebene Aktion aus, wenn die entsprechende Regel zutrifft.

Wenn sich Ihre Anforderungen für die Regeln ändern, editieren Sie die Konfigurationsdatei, ohne Governor zu stoppen. Jeder Governor-Dämon erkennt die geänderte Datei und liest sie erneut.

Die Konfigurationsdatei muss in einem Verzeichnis erstellt werden, das für alle Datenbankpartitionen angehängt ist, sodass der Governor-Dämon für die einzelnen Datenbankpartitionen dieselbe Konfigurationsdatei lesen kann.

Die Konfigurationsdatei besteht aus drei erforderlichen Regeln, die die zu überwachende Datenbank, das Intervall, in dem Protokolldatensätze geschrieben werden und das Inaktivierungsintervall der Governor-Dämonen festlegen. Im Anschluss an diese Parameter enthält die Konfigurationsdatei eine Reihe optionaler Regeln zur Anwendungsüberwachung und zugehöriger Aktionen. Die folgenden Anmerkungen gelten für alle Regeln:

- Kommentare werden mit geschweiften Klammern { } begrenzt.
- Die meisten Einträge können in Großbuchstaben, Kleinbuchstaben oder gemischt angegeben werden. Die Ausnahme ist der Anwendungsname, der als Argument für die Regel `applname` angegeben wird, bei dem die Groß-/Kleinschreibung zu beachten ist.

- Jede Regel endet mit einem Semikolon (;).

Erforderliche Regeln

Die folgenden Regeln geben die zu überwachende Datenbank und das Intervall an, in dem der Dämon nach jedem Durchlauf der Aktivitäten wieder aktiv wird. Jede dieser Regeln wird nur einmal in der Datei angegeben.

dbname
Der Name oder Aliasname der zu überwachenden Datenbank.

account *nnn*
Abrechnungsdatensätze, die statistische Daten zur CPU-Auslastung für jede Verbindung enthalten, werden jeweils nach Ablauf des in Minuten angegebenen Intervalls geschrieben.

Anmerkung: Diese Option ist in der Windows-Umgebung nicht verfügbar.

Findet eine kurze Verbindungssitzung vollständig innerhalb eines Abrechnungsintervalls statt, so wird kein Protokollsatz geschrieben. Wenn Protokollsätze geschrieben werden, enthalten sie CPU-Statistiken, die Aufschluss über die CPU-Verwendung der Verbindung seit dem vorangegangenen Protokollsatz geben. Wird Governor gestoppt und erneut gestartet, wird die CPU-Verwendung möglicherweise in zwei Protokollsätzen aufgezeichnet. Diese können über die Anwendungs-IDs in den Protokollsätzen ermittelt werden.

interval
Das Intervall in Sekunden, nach dem der Dämon jeweils aktiv wird. Wenn Sie kein Intervall angeben, wird der Standardwert von 120 Sekunden verwendet.

Regeln, die Aktionen festlegen

Im Anschluss an die erforderlichen Regeln können Sie Regeln hinzufügen, die angeben, wie Anwendungen zu behandeln sind. Diese Regeln bestehen aus kleineren Komponenten, den so genannten Regelklauseln. Bei ihrer Verwendung müssen die Klauseln in einer bestimmten Reihenfolge in der Regelanweisung wie folgt angegeben werden:

1. **desc** (optional): Ein in Anführungszeichen eingeschlossener Kommentar über die Regel.
2. **time** (optional): Die Tageszeit, zu der die Regel ausgewertet wird.
3. **authid** (optional): Mindestens eine Berechtigungs-ID, unter der die Anwendung Anweisungen ausführt.
4. **applname** (optional): Der Name der ausführbaren Datei oder der Objektdatei, welche die Verbindung zur Datenbank herstellt. Dieser Name ist von der Groß-/Kleinschreibung abhängig. Der Anwendungsname muss in doppelte Anführungszeichen gesetzt werden, wenn er Leerzeichen enthält.
5. **setlimit**: Die Grenzwerte, die von Governor geprüft werden. Dies kann einer von mehreren Werten sein, wie zum Beispiel die CPU-Zeit, die Anzahl zurückgegebener Zeilen oder die inaktive Zeit.
6. **action** (optional): Die Aktion, die auszuführen ist, wenn ein Grenzwert erreicht wird. Wenn keine Aktion angegeben wird, verringert Governor die Priorität von Agenten, die für die Anwendung aktiv sind, um den Wert 10, wenn ein Grenzwert erreicht wird. Aktionen, die an der Datenbank ausgeführt werden können, sind zum Beispiel eine Verringerung der Agentenpriorität, eine

erzwungene Trennung der Verbindung der Anwendung zur Datenbank oder eine Einstellung von Zeitplanoptionen für die Operationen der Anwendung.

Zur Definition einer Regel kombinieren Sie die Regelklauseln, wobei Sie jede Klausel in einer Regel nur einmal verwenden, und schließen jede Regel mit einem Semikolon ab, wie in den folgenden Beispielen zu sehen ist:

```
desc "Keine UOW darf länger als 1 Stunde dauern"
setlimit uowtime 3600 action force;

desc "Beschränken der Nutzung von db2 CLP durch neuen Benutzer novice"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;
```

Wenn mehrere Regeln auf eine Anwendung zutreffen, werden alle Regeln angewendet. Gewöhnlich wird die Aktion, die dem zuerst angetroffenen Grenzwert einer Regel zugeordnet ist, auch zuerst angewendet. Ausnahmen treten dann auf, wenn Sie den Wert -1 für eine Klausel in einer Regel angeben. In diesem Fall kann ein in der Klausel der nachfolgenden Regel angegebener Wert nur den zuvor in der *gleichen* Klausel angegebenen Wert überschreiben: die übrigen Klauseln in der vorhergehenden Regel bleiben weiterhin gültig. Eine Regel gibt in den Klauseln `rowssel 100000 uowtime 3600` zum Beispiel an, dass die Priorität einer Anwendung verringert werden muss, wenn ihre abgelaufene Zeit 1 Stunde überschreitet oder die Anwendung mehr als 100.000 Zeilen auswählt. Eine nachfolgende Regel enthält die Klausel `uowtime -1`, um anzugeben, dass die gleiche Anwendung über unbegrenzte Zeit verfügen kann. Wenn in diesem Fall die Anwendung länger als 1 Stunde ausgeführt wird, wird ihre Priorität nicht geändert. Das heißt, die Klausel `uowtime -1` überschreibt die Klausel `uowtime 3600`. Wenn die Anwendung jedoch über 100.000 Zeilen auswählt, wird ihre Priorität herabgesetzt, weil die Klausel `rowssel 100000` weiterhin gültig ist.

Reihenfolge der Regelanwendung

Der Governor verarbeitet die Regeln in der Konfigurationsdatei vom Anfang der Datei nach unten. Wenn jedoch die Klausel setlimit einer späteren Regel weniger restriktiv ist als eine vorherige Regel, wird die restriktivere Regel immer noch angewendet. Zum Beispiel wird `admin` in der folgenden Konfigurationsdatei trotz der späteren Regel auf 5000 Zeilen begrenzt, da die erste Regel restriktiver ist.

```
desc "Erzwingen, dass alle 5000 oder mehr Zeilen auswählen"
setlimit rowssel 5000 action force;

desc "Benutzer admin das Auswählen von weiteren Zeilen erlauben"
authid admin
setlimit rowssel 10000 action force;
```

Um sicherzustellen, dass eine weniger restriktive Regel eine restriktivere Regel außer Kraft zu setzen, die in der Datei weiter oben steht, können Sie die Option -1 angeben, um die vorherige Regel vor dem Anwenden der neuen Regel zu löschen. Beispielsweise begrenzt die Anfangsregel in der folgenden Konfigurationsdatei alle Benutzer auf 5000 Zeilen. Die zweite Regel löscht diese Begrenzung für `admin` und die dritte Regel setzt die Begrenzung für `admin` auf 10000 Zeilen zurück.

```
desc "Erzwingen, dass alle 5000 oder mehr Zeilen auswählen"
setlimit rowssel 5000 action force;

desc "Begrenzung rowssel für admin löschen"
authid admin
setlimit rowssel -1;
```

```
desc "Nun die höhere Begrenzung rowssel für admin festlegen"
authid admin
setlimit rowssel 10000 action force;
```

Elemente für Governor-Regeln

Jede Regel in der Konfigurationsdatei von Governor wird aus Klauseln zusammengesetzt, welche die Bedingungen zur Anwendung der Regel und die Aktion angeben, die folgt, wenn die Regel als wahr ausgewertet wird. Die Klauseln müssen in der angezeigten Reihenfolge angegeben werden. In den Beschreibungen der Klauseln weisen geschweifte Klammern [ ] auf eine optionale Klausel hin.

Optionale Startelemente

[desc] Gibt eine Textbeschreibung der Regel an. Die Beschreibung muss in einfachen oder doppelten Anführungszeichen stehen.

[time] Gibt die Zeitspanne an, während der die Regel auswertet werden soll.

Die Zeitspanne muss im Format `time hh:mm hh:mm` angegeben werden, z. B. `time 8:00 18:00`. Wird diese Klausel nicht angegeben, ist die Regel 24 Stunden am Tag gültig.

[authid]

Gibt eine oder mehrere Berechtigungs-IDs (authid) an, mit denen die Anwendung ausgeführt wird. Mehrere Berechtigungs-IDs müssen durch ein Komma (,) voneinander getrennt werden, z. B. `authid gene, michael, james`. Wird diese Klausel in einer Regel nicht angegeben, gilt die Regel für alle Berechtigungs-IDs.

[applname]

Gibt den Namen der ausführbaren Datei (oder Objektdatei) an, welche die Verbindung zur Datenbank herstellt.

Mehrere Anwendungsnamen müssen durch ein Komma (,) voneinander getrennt werden, z. B. `applname db2bp, batch, geneprog`. Wird diese Klausel in einer Regel nicht angegeben, gilt die Regel für alle Anwendungsnamen.

Anmerkung:

1. Bei Anwendungsnamen muss Groß-/Kleinschreibung beachtet werden.
2. Der Datenbankmanager schneidet alle Anwendungsnamen auf 20 Zeichen ab. Stellen Sie sicher, dass die zu prüfende Anwendung mit den ersten 20 Zeichen ihres Anwendungsnamens eindeutig identifiziert wird. Ist das nicht der Fall, wird möglicherweise unbeabsichtigt eine andere Anwendung überprüft.

   Anwendungsnamen in der Governor-Konfigurationsdatei werden auf 20 Zeichen abgeschnitten, damit sie mit ihrer internen Darstellung übereinstimmen.

Grenzwertklauseln

setlimit

Gibt mindestens einen Grenzwerte an, den Governor überprüfen soll. Die Grenzwerte dürfen nur den Wert -1 oder Werte größer als 0 annehmen (z. B. `cpu -1 locks 1000 rowssel 10000`). Sie müssen mindestens eine der Begrenzungen (cpu, locks, rowsread, uowtime) angeben. Jede nicht angegebene Begrenzung wird durch die jeweilige Regel nicht eingeschränkt. Governor kann folgende Begrenzungen überprüfen:

cpu *nnn*
Gibt die Anzahl der CPU-Sekunden an, die eine Anwendung nutzen kann. Wenn Sie den Wert -1 angeben, schränkt Governor die CPU-Nutzung der Anwendung nicht ein.

locks *nnn*
Gibt die Anzahl der Sperren an, die eine Anwendung halten kann. Wenn Sie den Wert -1 angeben, schränkt Governor die Anzahl der durch die Anwendung gehaltenen Sperren nicht ein.

rowssel *nnn*
Gibt die Anzahl von Zeilen an, die an die Anwendung zurückgegeben werden. Dieser Wert ist nur auf dem Koordinatorknoten ungleich Null. Wenn Sie den Wert -1 angeben, schränkt Governor die Anzahl der auswählbaren Zeilen nicht ein. Der Maximalwert, der für *nnn* angegeben werden kann, lautet 4.294.967.298.

uowtime *nnn*
Gibt die Zeitspanne in Sekunden an, die verstreichen kann, nachdem eine UOW (Unit of Work, Arbeitseinheit) (UOW) zum ersten Mal aktiv wird. Wenn Sie den Wert -1 angeben, wird die abgelaufene Zeit nicht eingeschränkt.

Anmerkung: Wenn Sie die API `sqlmon` (den Datenbanksystemmonitor-Schalter) zum Inaktivieren des Monitorschalters für die UOW (Unit of Work, Arbeitseinheit) oder des Monitorschalters für die Zeitmarke verwendet haben, wirkt sich dies negativ auf die Fähigkeit von Governor aus, Anwendungen auf der Grundlage der abgelaufenen Zeit der UOW (Unit of Work, Arbeitseinheit) zu regeln. Governor verwendet das Überwachungsprogramm zum Sammeln von Systeminformationen. Wenn Sie die Schalter in der Konfigurationsdatei des Datenbankmanagers ausschalten, wird dieses Programm für die gesamte Instanz abgeschaltet. Dadurch erhält Governor diese Informationen nicht mehr.

idle *nnn*
Gibt die für eine Verbindung zulässige Leerlaufzeit in Sekunden an, bevor eine Aktion ausgeführt wird. Wenn Sie den Wert -1 angeben, wird die Leerlaufzeit der Verbindung nicht eingeschränkt.

Anmerkung: Bestimmte Datenbankdienstprogramme wie z. B. die Dienstprogramme für Backup und Restore stellen eine Verbindung zur Datenbank her und führen die anfallenden Arbeiten dann über EDUs aus, die für Governor nicht sichtbar sind. Diese Datenbankverbindungen werden als im Leerlauf befindlich identifiziert, sodass es zu einer Überschreitung des für die Leerlaufzeit definierten Zeitlimits kommen kann. Um zu verhindern, dass Governor entsprechende Maßnahmen in Bezug auf diese Dienstprogramme einleitet, können Sie für diese über die Berechtigungs-ID, über die sie aufgerufen wurden, den Wert -1 angeben. Um beispielsweise zu verhindern, dass Governor für Dienstprogramme aktiv wird, die unter der Berechtigungs-ID DB2SYS ausgeführt werden, müssen Sie ″authid DB2SYS setlimit idle -1″ angeben.

rowsread *nnn*
Gibt die Anzahl der von einer Anwendung auswählbaren Zeilen an. Wenn Sie den Wert -1 angeben, kann die Anwendung eine

unbegrenzte Anzahl an Zeilen auswählen. Der Maximalwert, der für *nnn* angegeben werden kann, lautet 4.294.967.298.

Anmerkung: Diese Begrenzung ist nicht mit rowssel identisch. Der Unterschied besteht darin, dass sich rowsread auf die Anzahl der Zeilen bezieht, die gelesen werden mussten, um eine Ergebnismenge zurückgeben zu können. Die Anzahl der gelesenen Zeilen schließt Lesevorgänge der Katalogtabellen durch die Steuerkomponente ein und kann durch die Verwendung von Indizes verringert werden.

Aktionsklauseln

[action]

Gibt die Aktion an, die ausgeführt werden muss, wenn ein angegebener Grenzwert überschritten wird. Sie können folgende Aktionen angeben:

Anmerkung: Wenn ein Grenzwert überschritten wird und die Aktionsklausel nicht angegeben ist, verringert Governor die Priorität der für die Anwendung aktiven Agenten um 10.

nice *nnn*

Gibt eine Änderung der relativen Priorität der für die Anwendung aktiven Agenten an. Gültige Werte: -20 bis +20.

Dieser Parameter ist unter folgenden Voraussetzungen wirksam:

- Auf UNIX-Plattformen muss der Parameter *agentpri* des Datenbankmanagers den Standardwert aufweisen. Andernfalls überschreibt er die Prioritätsklausel.
- Auf Windows-Plattformen kann der Datenbankmanagerparameter *agentpri* gemeinsam mit der Aktion *priority* verwendet werden.

Sie können mit Governor die Priorität von Anwendungen steuern, die in der Superklasse des Standardbenutzerserivce SYSDEFAULTUSERCLASS ausgeführt werden. Wenn Sie Governor verwenden, um die Priorität einer Anwendung zu senken, die in dieser Service-Superklasse ausgeführt wird, trennt sich der Agent selbst von seinem abgehenden Korrelator (falls ihm einer zugeordnet ist) und legt seine relative Priorität entsprechend der durch Governor angegebenen Priorität fest. Sie können mit Governor keine Prioritäten von Agenten in benutzerdefinierten Service-Superklassen oder -Unterklassen ändern. Stattdessen müssen Sie die Einstellung für die Agentenpriorität der Service-Superklasse oder -Unterklasse verwenden, um Anwendungen zu steuern, die in diesen Serviceklassen ausgeführt werden. Sie können Governor jedoch verwenden, um Verbindungen in jeder Serviceklasse zwangsweise zu trennen.

Anmerkung: Unter AIX 5.3 muss der Eigner die Berechtigungsgruppe (Capability) CAP_NUMA_ATTACH haben, um die relative Priorität von Agenten erhöhen zu können, die für die Anwendung tätig sind. Zum Erteilen dieser Berechtigungsgruppe melden Sie sich mit Rootberechtigung an und führen den folgenden Befehl aus:

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

force Gibt an, dass der Agent, der die Anwendung bedient, zur Beendigung gezwungen wird. (Setzt die Anweisung FORCE APPLICATION ab, um den koordinierenden Agenten zu beenden.)

Anmerkung: In Umgebungen mit partitionierten Datenbanken wird die Aktion FORCE nur ausgeführt, wenn der Governor-Dämon in der koordinierenden Datenbankpartition der Anwendung aktiv ist. Wenn also ein Governor-Dämon in Datenbankpartition A aktiv ist und ein Grenzwert für eine Anwendung überschritten wird, deren koordinierende Datenbankpartition die Datenbankpartition B ist, wird die Aktion FORCE ausgelassen.

schedule [class]

Durch eine Zeitzuweisung werden die Prioritäten der für die Anwendungen aktiven Agent verbessert, um die durchschnittlichen Antwortzeiten zu minimieren, ohne Anwendungen zu benachteiligen.

Governor wählt die Anwendungen mit den jeweils höchsten Werten für die Zeitzuweisung nach den folgenden drei Kriterien aus:

- Die Anwendung, welche die meisten Sperren hält

  Diese Auswahl erfolgt mit dem Ziel, die Anzahl von Situationen zu verringern, in denen auf Sperren gewartet werden muss (lockwaits).
- Die älteste Anwendung
- Die Anwendung mit der kürzesten geschätzten verbleibenden Ausführungszeit

  Diese Auswahl ist ein Versuch, möglichst viele kurzzeitig aktive Anweisungen während des Intervalls zum Abschluss zu bringen.

Die drei bei den jeweiligen Kriterien höchstplatzierten Anwendungen erhalten höhere Prioritäten als alle anderen Anwendungen. Das bedeutet, der höchstplatzierten Anwendung in jeder Kriteriumsgruppe wird die höchste Priorität, den nächsthöchsten Anwendungen die zweithöchste Priorität und den dritthöchsten Anwendungen die dritthöchste Priorität erteilt. Wenn eine einzelne Anwendung auf einem der drei höchsten Plätze in mehreren Kategorien rangiert, erhält sie die entsprechende Priorität für das Kriterium, bei dem sie den höchsten Rang innehat. Die nächsthöchste Anwendung erhält die nächsthöchste Priorität für das andere Kriterium. Wenn zum Beispiel Anwendung A die meisten Sperren hält, jedoch die drittkürzeste geschätzte verbleibende Ausführungszeit hat, erhält sie die höchste Priorität für das erste Kriterium. Die Anwendung, die mit der kürzesten geschätzten Ausführungszeit auf Platz vier rangiert erhält dadurch die dritthöchste Priorität für dieses zweite Kriterium.

Die Anwendungen, die von dieser Governor-Regel ausgewählt werden, werden in bis zu drei Klassen unterteilt. Für jede Klasse wählt Governor neun Anwendungen aus, welche nach den oben aufgeführten Kriterien jeweils die drei höchstplatzierten Anwendungen jeder Klasse sind. Wenn Sie die Klassenoption (class) angeben, werden alle Anwendungen, die von dieser Regel ausgewählt werden, als eine einzige Klasse betrachtet, wobei neun Anwendungen ausgewählt werden und wie oben beschrieben höhere Prioritäten erhalten.

Wenn eine Anwendung in mehreren Governor-Regeln ausgewählt wird, wird sie von der letzten Regel regiert, in der sie ausgewählt wird.

Anmerkung: Wenn Sie die API `sqlmon` (den Datenbanksystemmonitor-Schalter) zum Inaktivieren des Schalters der Anweisung verwendet haben, wird die Fähigkeit von Governor beeinträchtigt, Anwendungen auf der Grundlage der abgelaufenen Zeit der Anweisung zu überprüfen. Governor verwendet das Überwachungsprogramm zum Sammeln von Systeminformationen. Wenn Sie die Schalter in der Konfigurationsdatei des Datenbankmanagers ausschalten, wird dieses Programm für die gesamte Instanz abgeschaltet. Dadurch erhält Governor diese Informationen nicht mehr.
Eine Zeitzuweisungsaktion kann folgenden Zwecken dienen:

- Anwendungen in unterschiedlichen Gruppen erhalten Zeit zugewiesen, ohne dass diese Zeit gleichmäßig unter allen Anwendungen aufgeteilt wird.

  Wenn beispielsweise 14 Anwendungen (3 kurze, 5 mittlere und 6 lange) gleichzeitig aktiv sind, haben möglicherweise alle eine schlechte Antwortzeit, weil sie die CPU-Zeit teilen. Der Datenbankadministrator kann zwei Gruppen einrichten, eine mit mittleren Anwendungen und eine mit langen Anwendungen. Mithilfe der Prioritäten kann Governor alle kurzen Anwendungen ausführen und sicherstellen, dass höchstens drei mittlere und drei lange Anwendungen gleichzeitig aktiv sind. Zu diesem Zweck enthält die Governor-Konfigurationsdatei eine Regel für mittlere und eine Regel für lange Anwendungen.

  Das folgende Beispiel zeigt einen Abschnitt einer Governor-Konfigurationsdatei, der dies veranschaulicht:

  ```
  desc "Mittellange Anwendungen in 1 Zeitzuweisungsklasse zusammenfassen"
  applname medq1, medq2, medq3, medq4, medq5
  setlimit cpu -1
  action schedule class;

  desc "Lange Anwendungen in 1 Zeitzuweisungsklasse zusammenfassen"
  applname longq1, longq2, longq3, longq4, longq5, longq6
  setlimit cpu -1
  action schedule class;
  ```

- Unterschiedliche Benutzergruppen (z. B. Abteilungen in Unternehmen) erhalten gleiche Kriterien für die Vergabe von Prioritäten.

  Wenn eine Gruppe eine Vielzahl von Anwendungen ausführt, kann der Administrator sicherstellen, dass andere Gruppen dennoch akzeptable Antwortzeiten für ihre Anwendungen erhalten. Sind zum Beispiel drei Abteilungen beteiligt (Finanzen, Lagerbestand und Planung), kann man alle Benutzer der Finanzabteilung in eine Gruppe legen, alle Benutzer aus dem Lager in eine andere und alle Planer in die dritte. Dann werden die Verarbeitungskapazitäten in etwa gleichmäßig unter den drei Abteilungen aufgeteilt. Das folgende Beispiel zeigt einen Abschnitt einer Governor-Konfigurationsdatei, der dies veranschaulicht:

  ```
  desc "Benutzer der Finanzwesenabteilung in Klasse zusammenfassen"
  authid tom, dick, harry, mo, larry, curly
  setlimit cpu -1
  ```

```
action schedule class;

desc "Benutzer der Lagerabteilung in Klasse zusammenfassen"
authid pat, chris, jack, jill
setlimit cpu -1
action schedule class;

desc "Benutzer der Planungsabteilung in Klasse zusammenfassen"
authid tara, dianne, henrietta, maureen, linda, candy
setlimit cpu -1
action schedule class;
```

- Governor bindet alle Anwendungen in eine Zeitzuweisung ein.

  Wenn die Option class nicht mit der Aktion angegeben wird, erstellt Governor eigene Klassen nach der Anzahl der aktiven Anwendungen, für die diese Aktion gilt, und ordnet die Anwendungen in verschiedene Klassen ein. Dies erfolgt entsprechend der Aufwandsschätzung durch den DB2-Abfragecompiler für die Abfrage, die von der Anwendung ausgeführt wird. Der Administrator kann alle Anwendungen für die Zeitzuweisung (Scheduling) auswählen, indem er nicht angibt, welche Anwendungen auszuwählen sind. Die Klauseln *applname* und *authid* werden also nicht angegeben, und die Klausel *setlimit* verursacht keine Einschränkungen.

Anmerkung: Wenn ein Grenzwert überschritten wird und die Aktionsklausel nicht angegeben ist, verringert Governor die Priorität der für die Anwendung aktiven Agenten.

Beispiel für eine Governor-Konfigurationsdatei

Das folgende Beispiel zeigt eine Governor-Konfigurationsdatei, die verschiedene Regeln mit Aktionen definiert:

```
 {Einmal pro Sekunde aktiv werden, Datenbankname lautet ibmsampl,
 Abrechnungsdatensätze alle 30 Minuten }
interval 1; dbname ibmsampl; account 30;

desc "CPU-Einschränkungen gelten 24 Stunden pro Tag für alle"
setlimit cpu 600 rowssel 1000000 rowsread 5000000;

desc "Keine UOW darf länger als 1 Stunde dauern"
setlimit uowtime 3600 action force;

desc 'Verlangsamen einer Untermenge von Anwendungen'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;

desc "Governor soll diese 6 langen Anwendungen in 1 Prioritätenklasse einordnen"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;

desc "Zeitzuweisung für alle Anwendungen von der Planungsabteilung"
authid planid1, planid2, planid3, planid4, planid5
setlimit cpu -1
action schedule;

desc "Einordnen aller CPU-Fresser in eine Klasse zur Verbrauchssteuerung"
setlimit cpu 3600
action schedule class;

desc "Beschränken der Nutzung von db2 CLP durch neuen Benutzer novice"
authid novice
applname db2bp.exe
```

```
setlimit cpu 5 locks 100 rowssel 250;

desc "Zur Tagesarbeitszeit darf keine Anwendung länger als 10 Sek. aktiv sein"
time 8:30 17:00 setlimit cpu 10 action force;

desc "Einige Benutzer, die mit Leistungsoptimierung befasst sind, dürfen
          einige Ihrer Anwendungen in der Mittagspause durchführen"
time 12:00 13:00 authid ming, geoffrey, john, bill
applname tpcc1, tpcc2, tpcA, tpvG setlimit cpu 600 rowssel 120000 action force;

desc "Einige Benutzer sollten nicht eingeschränkt werden -- Datenbankadministrator
        und einige andere. Da dies die letzte Angabe in der Datei ist,
      wird zuvor Angegebenes hierdurch überschrieben."
authid gene, hershel, janet setlimit cpu -1 locks -1 rowssel -1 uowtime -1;

desc "Erhöhen der Priorität einer wichtigen Anwendung, sodass sie immer
           schnell verarbeitet wird"
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;
```

Governor-Protokolldateien

Wenn ein Governor-Dämon eine Aktion ausführt, schreibt er einen Datensatz in seine Protokolldatei. Zu diesen Aktionen gehören:

- Erzwungene Beendigung einer Anwendung
- Lesen der Governor-Konfigurationsdatei
- Ändern einer Anwendungspriorität
- Feststellen einer Fehler- oder Warnbedingung
- Starten oder Stoppen

Jeder Governor-Dämon besitzt eine eigene Protokolldatei. Getrennte Protokolldateien vermeiden Engpässe durch Sperrungen von Dateien, die auftreten könnten, wenn viele Governor-Dämonen gleichzeitig in dieselbe Datei schreiben würden. Mit dem Dienstprogramm `db2govlg` können Sie die Protokolldateien zusammenfügen und Abfragen ausführen.

Die Protokolldateien werden im Unterverzeichnis `log` des Verzeichnisses `sqllib` gespeichert. Für Windows-Betriebssysteme gilt jedoch die Ausnahme, dass sich das Unterverzeichnis `log` unterhalb des Instanzverzeichnisses befindet. Wenn Sie Governor mit dem Befehl `db2gov` starten, geben Sie den Basisnamen für die Protokolldatei an. Stellen Sie sicher, dass der Protokolldateiname den Datenbanknamen enthält, um die Protokolldateien in jeder Datenbankpartition unterscheiden zu können, in der Governor ausgeführt wird. Um in einer Umgebung mit partitionierten Datenbanken sicherzustellen, dass der Dateiname für jeden Governor eindeutig ist, wird die Partitionsnummer der Datenbank, in der der Governor-Dämon ausgeführt wird, automatisch an den Protokolldateinamen angefügt.

Datensatzformat der Protokolldatei

Ein Datensatz in der Protokolldatei hat folgendes Format:

```
Datum Uhrzeit Knoten-Nr. Satztyp Nachricht
```

Anmerkung: Das Format der Felder *Datum* und *Uhrzeit* ist jjjj-mm-tt hh.mm.ss. Sie können die Protokolldateien in jeder Datenbankpartition durch Sortieren nach diesem Feld zusammenfügen.

Das Feld *Knoten-Nr.* gibt die Nummer der Datenbankpartition an, in der Governor aktiv ist.

Das Feld *Satztyp* enthält unterschiedliche Werte, je nach Typ des in das Protokoll geschriebenen Eintrags. Folgende Werte können eingetragen werden:

- `START`: Governor wurde gestartet.
- `STOP`: Governor wurde gestoppt.
- `FORCE`: Eine Anwendung wurde zwangsweise beendet.
- `NICE`: Die Priorität einer Anwendung wurde geändert.
- `ERROR`: Ein Fehler ist aufgetreten.
- `WARNING`: Eine Warnung ist aufgetreten.
- `READCFG`: Governor hat die Konfigurationsdatei gelesen.
- `ACCOUNT`: Die Abrechnungsstatistik der Anwendung.
- `SCHEDGRP`: Eine Änderung der Agentenpriorität ist aufgetreten.

Einige dieser Werte werden im Folgenden näher beschrieben.

START
Der Datensatz START wird geschrieben, wenn der Governor gestartet wird. Er hat folgendes Format:

```
Database = <datenbankname>
```

STOP Der Datensatz STOP wird geschrieben, wenn der Governor gestoppt wird. Er hat folgendes Format:

```
Database = <datenbankname>
```

FORCE
Der Datensatz wird immer dann geschrieben, wenn der Governor feststellt, dass die Beendigung einer Anwendung entsprechend einer Regel in der Konfigurationsdatei des Governor zu erzwingen ist. Der Datensatz FORCE hat folgendes Format:

```
<anwendungsname> <authentifizierungs-id> <anwendungs-id>
<koordinierende_partition> <konfigurationszeile>
<einschränkung_überschritten>
```

Dabei gilt Folgendes:

<koordinierende_partition>
Gibt die Nummer der koordinierenden Datenbankpartition der Anwendung an.

<konfigurationszeile>
Gibt die Zeilennummer in der Governor-Konfigurationsdatei an, in der sich die Regel befindet, die das Erzwingen der Anwendungsbeendigung verursacht.

<einschränkung_überschritten>
Stellt Einzelangaben dazu zur Verfügung, wie die Regel überschritten wurde. Dies können die folgenden Werte sein:

- `CPU`: Die gesamte USR CPU plus die SYS CPU-Zeit in Sekunden.
- `Locks`: Die Summe der Sperren, die von der Anwendung gehalten werden
- `Rowssel`: Die Summe der Zeilen, die von der Anwendung ausgewählt werden
- `Rowsread`: Die Summe der Zeilen, die von der Anwendung gelesen werden
- `Idle`: Die Zeitdauer, die die Anwendung inaktiv war

- ET: Die abgelaufene Zeit seit dem Start der aktuellen UOW der Anwendung (das uowtime-Zeitlimit wurde überschritten)

NICE Der Datensatz NICE wird geschrieben, wenn die Priorität einer Anwendung durch eine Prioritätsaktion geändert wird, die in der Konfigurationsdatei des Governor angegeben wird. Der Datensatz NICE hat folgendes Format:

```
<anwendunsname> <authentifizierungs-id> <anwendungs-id> <nice-wert>
(<konfigurationszeile>) <einschränkung_überschritten>
```

Dabei gilt Folgendes:

<nice-wert>
Gibt das Inkrement (oder Dekrement) an, das auf den Prioritätswert des Agentenprozesses der Anwendung angewendet wird.

<konfigurationszeile>
Gibt die Zeilennummer in der Konfigurationsdatei des Governor an, in der sich die Regel befindet, die das Ändern der Anwendungspriorität bewirkt.

<einschränkung_überschritten>
Stellt Einzelangaben dazu zur Verfügung, wie die Regel überschritten wurde. Dies können die folgenden Werte sein:

- CPU: Die gesamte USR CPU plus die SYS CPU-Zeit in Sekunden.
- Locks: Die Summe der Sperren, die von der Anwendung gehalten werden
- Rowssel: Die Summe der Zeilen, die von der Anwendung ausgewählt werden
- Rowsread: Die Summe der Zeilen, die von der Anwendung gelesen werden
- Idle: Die Zeitdauer, die die Anwendung inaktiv war
- ET: Die abgelaufene Zeit seit dem Start der aktuellen UOW der Anwendung (das uowtime-Zeitlimit wurde überschritten)

ERROR
Der Datensatz ERROR wird geschrieben, wenn der Governor-Dämon beendet werden muss.

WARNING
Der Datensatz WARNING wird in den folgenden Situationen in das Governor-Protokoll geschrieben:

- Die API sqlefrce wurde aufgerufen, um die Beendigung einer Anwendung zu erzwingen, jedoch wurde ein positiver SQLCODE-Wert zurückgegeben.
- Ein Momentaufnahmenaufruf gab einen positiven SQLCODE-Wert zurück, der nicht 1611 (″SQL1611 Es wurden keine Daten zurückgegeben″) war.
- Ein Momentaufnahmenaufruf gab einen negativen SQLCODE-Wert zurück, der nicht -1224 (″SQL1224N Ein Datenbankagent konnte nicht für die Anforderung gestartet werden, oder er wurde aufgrund eines Systemabschlusses der Datenbank bzw. durch den Befehl FORCE beendet″) oder -1032 (″SQL1032N Der Befehl DB2START wurde nicht abgesetzt″) war. Diese Rückkehrcodes treten auf, wenn eine zuvor aktive Instanz inaktiv wurde.

- In einer UNIX-Umgebung wurde ein Versuch unternommen, eine Signalroutine zu installieren. Der Versuch ist fehlgeschlagen.

ACCOUNT

Der Datensatz ACCOUNT wird in den folgenden Situationen in das Governor-Protokoll geschrieben:

- Der Wert von agent_usr_cpu oder agent_sys_cpu für diese Anwendung wurde geändert, seitdem der letzte Datensatz ACCOUNT für diese Anwendung geschrieben wurde.
- Es wird festgestellt, dass die Anwendung nicht mehr aktiv ist.

Der Datensatz ACCOUNT hat folgendes Format:

<authentifizierungs-id> <anwendungs-id> <anwendungsname> <verbindungsdauer> <agent_usr_cpu delta> <agent_sys_cpu delta>

SCHEDGRP

Der Datensatz SCHEDGRP wird immer unter den folgenden Umständen geschrieben:

- Die Anwendung wird einer Zeitplanungsgruppe hinzugefügt.
- Die Anwendung wird von einer Zeitplanungsgruppe in eine andere versetzt.

Der Datensatz SCHEDGRP hat folgendes Format:

```
<anwendungsname> <authentifizierungs-id> <anwendungs-id>
<konfigurationszeile> <einschränkung_überschritten>
```

Dabei gilt Folgendes:

<konfigurationszeile>
Gibt die Zeilennummer in der Governor-Konfigurationsdatei an, in der sich die Regel befindet, die die Terminierung der Anwendung bewirkt.

<einschränkung_überschritten>
Stellt Einzelangaben dazu zur Verfügung, wie die Regel überschritten wurde. Dies können die folgenden Werte sein:

- `CPU`: Die gesamte USR CPU plus die SYS CPU-Zeit in Sekunden.
- `Locks`: Die Summe der Sperren, die von der Anwendung gehalten werden
- `Rowssel`: Die Summe der Zeilen, die von der Anwendung ausgewählt werden
- `Rowsread`: Die Summe der Zeilen, die von der Anwendung gelesen werden
- `Idle`: Die Zeitdauer, die die Anwendung inaktiv war
- `ET`: Die abgelaufene Zeit seit dem Start der aktuellen UOW der Anwendung (das uowtime-Zeitlimit wurde überschritten)

Da standardisierte Werte geschrieben werden, können Sie die Protokolldateien nach unterschiedlichen Aktionstypen abfragen. Das Feld *Nachricht* enthält andere nicht standardisierte Informationen, die sich nach dem im Feld *Satztyp* angegebenen Wert ändern. Ein Eintrag des Typs `FORCE` oder `NICE` beispielsweise zeigt Informationen zur Anwendung im Feld *Nachricht* an, während ein Eintrag des Typs `ERROR` eine Fehlernachricht enthält.

Eine Beispielprotokolldatei könnte folgendermaßen aussehen:

```
1995-12-11 14.54.52    0 START      Database = TQTEST
1995-12-11 14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
1995-12-11 14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
1995-12-11 14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

Abfragen von Governor-Protokolldateien

Jeder Governor-Dämon schreibt in eine eigene Protokolldatei. Mit dem Dienstprogramm `db2govlg` können Sie die Protokolldatei abfragen. Sie können die Protokolldateien einer einzigen oder aller Datenbankpartitionen, nach Datum und Uhrzeit sortiert, auflisten. Außerdem können Sie Abfragen auf der Grundlage des Protokollfelds *RecType* (Satztyp) durchführen. `db2govlg` hat folgende Syntax:

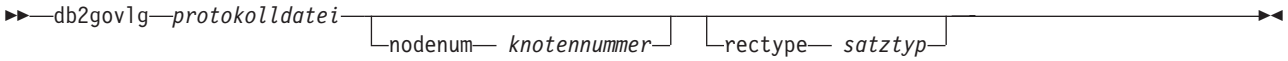


Abbildung 19. Syntax von db2govlg

Der Befehl hat die folgenden Parameter:

protokolldatei
: Der Basisdateiname der Protokolldatei (oder Dateien), die Sie abfragen möchten

nodenum *knotennummer*
: Die Knotennummer der Datenbankpartition, auf der Governor aktiv ist

rectype *satztyp*
: Der Satztyp, die Sie abfragen möchten. Es gibt folgende Satztypen:
  - START
  - READCFG
  - STOP
  - FORCE
  - NICE
  - ERROR
  - WARNING
  - ACCOUNT

Für die Verwendung dieses Dienstprogramms gibt es keinerlei Berechtigungseinschränkungen. Dadurch können alle Benutzer abfragen, ob Governor ihre Anwendung geändert hat. Wenn Sie den Zugriff auf dieses Dienstprogramm beschränken möchten, können Sie die Gruppenberechtigungen für die Datei `db2govlg` ändern.

Kapitel 16. Durchführen von Vergleichstests

Die Durchführung von Vergleichstests (Benchmarktests) ist ein natürlicher Bestandteil des Entwicklungszyklus für Anwendungen. Sie erfordert die Zusammenarbeit von Anwendungsentwicklern und Datenbankadministratoren (DBAs) und sollte für eine Anwendung stattfinden, um Daten über die aktuelle Leistung zu erhalten und Ansätze zur Leistungsoptimierung zu ermitteln. Wenn der Code einer Anwendung bereits mit größtmöglicher Effizienz arbeitet, können eventuell weitere Leistungsvorteile durch die Optimierung der Konfigurationsparameter der Datenbank und des Datenbankmanagers realisiert werden. Sie können außerdem Anwendungsparameter optimieren, um die Anforderungen einer Anwendung besser zu erfüllen.

Durch verschiedene Arten von Vergleichstests lassen sich bestimmte Arten von Informationen zu gewinnen:

- Ein Vergleichstest der Art *Transaktion pro Sekunde* liefert Anhaltspunkte über die Leistungskapazität des Datenbankmanagers unter bestimmten, eingeschränkten Laborbedingungen.
- Ein *Anwendungsvergleichstest* prüft die gleichen Durchsatzkapazitäten unter Bedingungen, die den Produktionsbedingungen näher kommen.

Die Optimierung von Konfigurationsparametern durch Vergleichstests legt diese *Realbedingungen* zugrunde und erfordert ein wiederholtes Ausführen von SQL aus der Anwendung mit variierenden Parameterwerten, bis die Anwendung mit der höchstmöglichen Effizienz arbeitet.

Die hier beschriebenen Vergleichstestmethoden sind auf die Optimierung von Konfigurationsparametern ausgerichtet. Jedoch kann derselbe Grundansatz auch für die Optimierung anderer, die Leistung beeinflussender Faktoren herangezogen werden, wie zum Beispiel:

- SQL-Anweisungen
- Indizes
- Tabellenbereichskonfiguration
- Anwendungscode
- Hardwarekonfiguration

Vergleichstests geben Aufschluss darüber, wie der Datenbankmanager unter unterschiedlichen Bedingungen reagiert. Es können Szenarios entwickelt werden, um die Behandlung von Deadlocks, die Leistung von Dienstprogrammen, die verschiedenen Methoden zum Laden von Daten, die Transaktionsgeschwindigkeit bei wachsenden Benutzerzahlen und auch die Auswirkungen der Verwendung eines neuen Produktrelease auf die Anwendung zu testen.

Vergleichstestmethoden

Bei der Durchführung von Vergleichstests wird eine reproduzierbare Umgebung zugrunde gelegt, sodass der gleiche Test unter den gleichen Bedingungen Ergebnisse liefert, deren Vergleich legitim ist.

Sie können mit den Vergleichstests beginnen, indem Sie die Testanwendung in einer Normalumgebung ausführen. Wenn Sie ein Leistungsproblem orten, können Sie spezielle Testszenarios entwickeln, die den Wirkungsbereich der getesteten Funktion begrenzen. Die speziellen Testszenarios brauchen nicht eine gesamte

Anwendung zu emulieren, um wertvolle Informationen zu liefern. Es empfiehlt sich, mit einfachen Messungen zu beginnen und die Komplexität der Methoden nur dann zu erhöhen, wenn dies erforderlich ist.

Gute Vergleichstests oder Messungen besitzen folgende Merkmale:

- Die Tests sind wiederholbar.
- Jede Wiederholung eines Tests beginnt im gleichen Systemstatus.
- Es sind keine anderen Funktionen bzw. Anwendungen im System aktiv, sofern vom Testszenario nicht vorgesehen ist, dass bestimmte andere Systemaktivitäten gleichzeitig stattfinden.

  Anmerkung: Gestartete Anwendungen belegen Speicher, selbst wenn Sie auf Symbolgröße verkleinert wurden oder momentan inaktiv sind. Dadurch erhöht sich die Wahrscheinlichkeit, dass Seitenauslagerungen zu einer Verzerrung der Vergleichstestergebnisse führen, wodurch die Wiederholbarkeitsregel verletzt wird.
- Die Hardware und die Software, die für die Vergleichstests verwendet werden, entsprechen der realen Produktionsumgebung.

Zur Durchführung von Vergleichstests erstellen Sie ein Szenario und führen die Anwendungen mehrere Male in diesem Szenario aus, indem Sie bei jeder Ausführung wichtige Informationen erfassen. Die Erfassung wichtiger Informationen nach jeder Ausführung ist von höchster Bedeutung bei der Ermittlung der Änderungen, die eine Leistungsverbesserung sowohl für die Anwendung als auch für die Datenbank bewirken können.

Vorbereiten von Vergleichstests

Schließen Sie den logischen Entwurf der Datenbank ab, für die Sie die Anwendung ausführen, bevor Sie mit den Leistungsvergleichstests beginnen. Richten Sie Tabellen, Sichten und Indizes ein und füllen Sie sie mit Daten. Normalisieren Sie Tabellen, binden Sie Anwendungspakete und füllen Sie Tabellen mit realistischen Daten.

Das endgültige physische Modell der Datenbank sollte bereits ebenfalls feststehen. Platzieren Sie Datenbankmanagerobjekte an ihre endgültigen Datenträgerpositionen, definieren Sie die Größe von Protokolldateien, legen Sie die Position von Arbeitsdateien und Backups fest und testen Sie die Backup-Prozeduren. Prüfen Sie ferner Ihre Pakete, um sicherzustellen, dass Leistungsoptionen wie Zeilenblockung aktiviert werden, wenn dies möglich ist.

Sie sollten einen Stand der Programmier- und Testphasen für die Anwendung erreicht haben, der es Ihnen ermöglicht, die Vergleichstestprogramme zu erstellen. Während der Vergleichstests können die praktischen Grenzen einer Anwendungen zutage treten. Jedoch liegt das Ziel der hier beschriebenen Vergleichstests in der Messung der Leistung und nicht in der Feststellung von Fehlern oder abnormalen Beendigungsbedingungen.

Das Vergleichstestprogramm muss in einer möglichst wirklichkeitsnahen Nachbildung der tatsächlichen Produktionsumgebung ausgeführt werden. Im Idealfall sollte ein Server des gleichen Modells mit derselben Speicher- und Festplattenkonfiguration verwendet werden. Dies ist besonders dann von Bedeutung, wenn die Anwendung letztendlich eine große Anzahl von Benutzern und große Mengen von Daten verarbeiten soll. Das Betriebssystem und mögliche Einrichtungen der Datenübertragung oder des Dateiservice sollten ebenfalls bereits optimiert worden sein.

Stellen Sie sicher, dass Sie Vergleichstests mit einer Datenbank in Größe der Produktionsdatenbank durchführen. Eine einzelne SQL-Anweisung sollte die gleiche Menge an Daten liefern und den gleichen Aufwand für Sortiervorgänge erfordern wie in der Geschäftsumgebung. Diese Regel gewährleistet, dass die Anwendung repräsentative Speicheranforderungen testet.

Die zu testenden SQL-Anweisungen sollten entweder zur Kategorie *repräsentatives SQL* oder zur Kategorie *Extremfall-SQL* (Worst-Case) gehören, wie im Folgenden erläutert wird:

Repräsentatives SQL

Zu repräsentativem SQL werden solche Anweisungen gezählt, die während eines typischen Einsatzes der zu testenden Anwendung ausgeführt werden. Die Anweisungen, die zum Test ausgewählt werden, sind von der Art der Anwendung abhängig. Beispielsweise kann für eine Dateneingabeanwendung eine Anweisung INSERT getestet werden, während für eine Banktransaktion eine Anweisung FETCH, eine Anweisung UPDATE und mehrere Anweisungen INSERT getestet werden können. Die Häufigkeit, mit der die Anwendung ausgeführt wird, und der Umfang der von den ausgewählten Anweisungen verarbeiteten Daten sollte als durchschnittlich angesehen werden können. Wenn die aufgrund der Anweisungen zu verarbeitenden Datenmengen sehr umfangreich sind, sind diese Anweisungen unter der Kategorie *Extremfall-SQL* zu betrachten, selbst wenn es sich um typische SQL-Anweisungen handelt.

Extremfall-SQL

Zu dieser Kategorie gehören Anweisungen mit folgenden Merkmalen:

- Anweisungen, die häufig ausgeführt werden
- Anweisungen, durch die umfangreiche Datenmengen verarbeitet werden
- Anweisungen, die zeitkritisch sind

  Ein Beispiel hierfür ist eine Anwendung, die ausgeführt wird, wenn ein Telefonanruf von einem Kunden eingeht, und deren Anweisungen ausgeführt werden müssen, um Daten des Kunden abzurufen oder zu aktualisieren, während der Kunde wartet.
- Anweisungen mit der größten Anzahl zu verknüpfender Tabellen, oder mit dem komplexesten SQL in der Anwendung

  Ein Beispiel wäre eine Bankanwendung, die kombinierte Kontoauszüge über die monatlichen Kontobewegungen für sämtliche verschiedene Arten von Konten eines Kunden erstellt. Eine allgemeine Tabelle könnte vielleicht die Kundenadressen und die Kontonummern enthalten; jedoch müssen mehrere andere Tabellen verknüpft werden, um alle benötigten Daten über Kontotransaktionen verarbeiten und zusammenstellen zu können. Die Multiplikation des Aufwands, der für ein Konto erforderlich ist, mit den mehreren Tausend Konten, die im gleichen Zeitraum verarbeitet werden müssen, macht deutlich, das jede potenzielle Zeiteinsparung die Leistungsanforderungen erhöht.
- Anweisungen, die einen ungünstigen Zugriffspfad verwenden, zum Beispiel eine Anweisung, die nicht sehr oft ausgeführt wird und nicht von den Indizes unterstützt wird, die für die betroffenen Tabellen erstellt wurden
- Anweisungen, die über einen langen Zeitraum hinweg laufen
- Eine Anweisung, die nur bei der Initialisierung einer Anwendung ausgeführt wird, jedoch überproportionale Ressourcenanforderungen stellt

Ein Beispiel wäre eine Anwendung, die eine Liste der Arbeiten für Konten erstellt, die während des Arbeitstages auszuführen sind. Wenn die Anwendung gestartet wird, löst die erste größere SQL-Anweisung einen Join über zahlreiche Tabellen aus, um eine sehr umfangreiche Liste aller Konten zu erstellen, für die der Benutzer der Anwendung verantwortlich ist. Die Anweisung wird vielleicht nur wenige Male am Tag ausgeführt, aber ihre Ausführung nimmt einige Minuten in Anspruch, wenn sie nicht ausreichend optimiert wurde.

Erstellung von Vergleichstests

Beim Entwurf und bei der Implementierung eines Vergleichstestprogramms sind eine Reihe von Faktoren zu beachten. Da der Hauptzweck des Programms darin besteht, eine Benutzeranwendung zu simulieren, ist die allgemeine Struktur des Programms unterschiedlich. Sie können die gesamte Anwendung zum Vergleichstest verwenden, sodass Sie nur die entsprechenden Mittel zur Erfassung der Zeiten für die zu analysierenden SQL-Anweisungen einfügen. Bei großen oder komplexen Anwendungen ist es eventuell praktischer, nur die Blöcke mit den wichtigen Anweisungen in das Vergleichstestprogramm aufzunehmen.

Zum Testen der Leistung bestimmter SQL-Anweisungen können Sie diese Anweisungen allein in das Vergleichstestprogramm aufnehmen und die benötigten Anweisungen CONNECT, PREPARE, OPEN und andere sowie Mechanismen zur Zeiterfassung hinzufügen.

Ein weiterer wichtiger Gesichtspunkt ist die Art des Vergleichs, die verwendet werden sollte. Eine Möglichkeit ist die, eine Gruppe von SQL-Anweisungen über ein Zeitintervall wiederholt auszuführen. Das Verhältnis der Anzahl der ausgeführten Anweisungen zu diesem Zeitintervall ergäbe einen Wert für den Durchsatz für die Anwendung. Eine andere Möglichkeit ist die, einfach die für die Ausführung einzelner SQL-Anweisungen erforderliche Zeit zu bestimmen.

Für alle Vergleichstests benötigen Sie ein effizientes Zeiterfassungssystem, um die während der Verarbeitung entweder einzelner SQL-Anweisungen oder der gesamten Anwendung abgelaufene Zeit zu berechnen. Bei der Simulation von Anwendungen, in denen einzelne SQL-Anweisungen isoliert ausgeführt werden, kann es wichtig sein, die Zeiten für die Anweisungen CONNECT, PREPARE und COMMIT zu verfolgen. Bei Programmen jedoch, die viele verschiedene Anweisungen verarbeiten, wird vielleicht nur eine einzige Anweisung CONNECT oder COMMIT benötigt, sodass die Erfassung der Ausführungszeit für eine einzelne Anweisung Priorität haben könnte.

Obwohl die Erfassung der benötigten Zeit für jede Abfrage einen wichtigen Faktor bei der Leistungsanalyse darstellt, werden durch sie nicht unbedingt potenzielle Leistungsengpässe offen gelegt. Zum Beispiel könnten Informationen über die CPU-Auslastung, über aktive Sperren und Pufferpoolein-/ausgaben Hinweise darauf geben, dass eine Anweisung durch die Ein-/Ausgabeaktivitäten gebremst wird und nicht die volle Kapazität der CPU nutzt. Ein Vergleichstestprogramm sollte es ermöglichen, diese Art von Daten für eine detailliertere Analyse bei Bedarf zu erfassen.

Nicht alle Anwendungen senden die gesamte Menge der durch eine Abfrage abgerufenen Zeilen an eine Ausgabeeinheit. Zum Beispiel könnte die gesamte Antwortmenge als Eingabe für ein anderes Programm dienen, sodass keine Zeilen aus der ersten Anwendung als Ausgabe gesendet werden. Die Formatierung von Daten zur Ausgabe auf der Anzeige verursacht in der Regel einen hohen CPU-Aufwand und

spiegelt den Benutzerbedarf nicht unbedingt wider. Um eine genaue Simulation zu erhalten, sollte ein Vergleichstestprogramm die Zeilenbehandlung der bestimmten Anwendung berücksichtigen. Wenn Zeilen an eine Ausgabeeinheit gesendet werden, könnte ineffizientes Formatieren den Hauptanteil der CPU-Verarbeitungszeit in Anspruch nehmen und so zu einer Verzerrung der tatsächlichen Leistungsdaten für die SQL-Anweisung als solche führen.

Das Vergleichstest-Tool db2batch: Es steht ein Vergleichstest-Tool (`db2batch`) im Unterverzeichnis `bin` des Verzeichnisses `sqllib` Ihrer Instanz zur Verfügung. Dieses Tool setzt viele der Richtlinien zur Erstellung eines Vergleichstestprogramms um. Dieses Tool kann SQL-Anweisungen entweder aus einer unstrukturierten Datei oder von der Standardeingabeeinheit lesen, die Anweisungen dynamisch beschreiben und vorbereiten und eine Antwortmenge zurückliefern. Es gibt Ihnen außerdem die Möglichkeit, die Größe der Antwortmenge und die Anzahl der Zeilen, die aus dieser Anwortmenge an eine Ausgabeeinheit gesendet werden, zu steuern.

Sie können die Stufe der leistungsbezogenen Daten angeben, die geliefert werden sollen, einschließlich der benötigten Zeit, CPU- und Pufferpoolauslastung, Sperren sowie anderer statistischer Daten aus dem Datenbankmonitor. Bei der Zeitmessung für eine Gruppe von SQL-Anweisungen erstellt `db2batch` auch eine Ergebnisübersicht und berechnet arithmetische und geometrische Mittelwerte. Sie können Syntaxoptionen anzeigen, indem Sie den Befehl `db2batch -h` in eine Befehlszeile eingeben.

Ausführung von Vergleichstests

Für eine Art von Datenbankvergleichstest wählen Sie einen Konfigurationsparameter aus und führen den Test mit verschiedenen Werten für den gewählten Parameter aus, bis die maximale Leistungssteigerung erzielt ist. Ein einzelner Test sollte die mehrmalige Ausführung der Anwendung (z. B. zwanzig- oder dreißigmal) mit demselben Parameterwert beinhalten, um einen Durchschnittswert für die benötigte Zeit zu ermitteln, der die Auswirkungen einer Änderung des Parameterwertes klarer wiedergibt.

Bei der Ausführung des Vergleichstests sollte der erste Durchlauf, der so genannte Aufwärmdurchlauf (Warm-up Run) von den nachfolgenden Iterationen, d. h. den Normaldurchläufen (Normal Run), getrennt betrachtet werden. Da der Aufwärmdurchlauf einige Startaktivitäten, wie zum Beispiel die Initialisierung des Pufferpools, durchführt, dauert er etwas länger als Normaldurchläufe. Obwohl die Daten aus dem Aufwärmdurchlauf durchaus realistische Werte darstellen können, sind sie für die statistische Auswertung nicht relevant. Zur Berechnung des Durchschnittswerts für die benötigte Zeit oder die CPU-Auslastung für eine bestimmte Gruppe von Parameterwerten sollten Sie nur die Ergebnisse aus Normaldurchläufen verwenden.

Sie können auch in Betracht ziehen, mit dem *Konfigurationsadvisor* den Aufwärmdurchlauf des Vergleichstests zu erstellen. Die im Konfigurationsadvisor gestellten Fragen geben einen Einblick in die Gesichtspunkte, die bei der Anpassung der Konfiguration Ihrer Umgebung für Normaldurchläufe in der Vergleichstestphase zu beachten sind. Sie können den Konfigurationsadvisor über die Steuerzentrale oder durch Ausführen des Befehls 'db2 autoconfigure' mit den entsprechenden Optionen starten.

Wenn Vergleichstests mit einzelnen Abfragen ausgeführt werden, müssen Sie sicherstellen, dass Sie die potenziellen Auswirkungen früherer Abfragen minimie-

ren, indem Sie den Pufferpool reinigen. Zum Reinigen des Pufferpools füllen Sie ihn mit einer Anzahl von Seiten, die für die Abfrage irrelevant sind (Flushing).

Nach der Ausführung der Durchläufe für eine bestimmte Gruppe von Parameterwerten können Sie den Wert eines einzelnen Parameters ändern. Zwischen den einzelnen Durchläufen müssen Sie jedoch folgende Maßnahmen durchführen, um die Vergleichstestumgebung wieder in den Ausgangszustand zurückzusetzen:

- . Wenn die Katalogstatistiken für den Test aktualisiert wurden, stellen Sie sicher, dass für jeden Durchlauf dieselben Werte für die Statistiken verwendet werden.
- Die Daten, die im Test verwendet werden, müssen konsistent sein, wenn sie durch die Tests aktualisiert werden. Dies kann folgendermaßen sichergestellt werden:
  - Durch Verwenden des Dienstprogramms RESTORE, um die gesamte Datenbank wiederherzustellen. Die Backup-Kopie der Datenbank enthält den früheren Zustand, der für den nächsten Test bereit ist.
  - Durch Verwenden des Dienstprogramms IMPORT oder LOAD, um eine exportierte Kopie der Daten wiederherzustellen. Diese Methode erlaubt den Restore nur der Daten, die vom Test betroffen waren. Die Dienstprogramme REORG und RUNSTATS sollten für die Tabellen und Indizes, die diese Daten enthalten, ausgeführt werden.
- Binden Sie die Anwendung erneut an die Datenbank, um sie in ihren Ausgangszustand zurückzusetzen.

Zusammenfassung: Führen Sie die folgenden Schritte oder Durchläufe aus, um Vergleichstests für eine Datenbankanwendung durchzuführen:

Schritt 1

Belassen Sie alle Parameter zur Optimierung der Datenbank und des Datenbankmanagers außer folgenden auf ihren **Standardwerten**:

- Die Parameter, die für die Belastung durch den Test und die Zielsetzung des Tests von Bedeutung sind. (Sie werden selten genügend Zeit haben, um Vergleichstests zur Optimierung aller Parameter durchzuführen, sodass es empfehlenswert ist, zu Beginn gute Schätzwerte für einige Parameter festzulegen und von diesen ausgehend die Optimierung vorzunehmen.)
- Protokolldateigrößen, die während der Einheiten- oder Systemtests für Ihre Anwendung festgelegt werden sollten.
- Alle die Parameter, die geändert werden müssen, damit die Anwendung ausgeführt werden kann (d. h., Änderungen zur Verhinderung negativer SQL-Rückkehrcodes aufgrund von Ereignissen wie Speicherknappheit für den Anweisungszwischenspeicher).

Führen Sie Ihre Anzahl von Durchläufen (Iterationen) für diesen Anfangszustand aus, und berechnen Sie den Durchschnittswert für die Zeitmesswerte oder die CPU.

Schritt 2

Wählen Sie einen und nur einen für die Optimierung zu testenden Parameter aus, und ändern Sie seinen Wert.

Schritt 3

Führen Sie eine weitere Anzahl von Durchläufen (Iterationen) aus, und berechnen Sie den Durchschnittswert für die Zeitmesswerte oder die CPU.

Schritt 4

Ergreifen Sie in Abhängigkeit von den Ergebnissen des Vergleichstests eine der folgenden Maßnahmen:

- Wenn die Leistung besser wird, ändern Sie den Wert desselben Parameters und kehren zu Schritt 3 zurück. Ändern Sie diesen Parameter so lange, bis der maximale Leistungswert gezeigt wird.
- Wenn die Leistung sinkt oder unverändert bleibt, setzen Sie den Parameter auf seinen vorigen Wert zurück, kehren zu Schritt 2 zurück und wählen einen anderen Parameter. Wiederholen Sie diese Prozedur, bis alle Parameter getestet sind.

  Anmerkung: Wenn Sie die Leistungsergebnisse in grafischer Darstellung festhalten würden, müssten Sie nach Stellen suchen, an denen die Kurve einen Maximalwert erreicht und dort verbleibt bzw. wieder absinkt.

Sie können ein Treiberprogramm schreiben, das Sie bei der Durchführung der Vergleichstests unterstützt. Dieses Treiberprogramm könnte in einer Sprache wie REXX bzw. für Linux- und UNIX-Plattformen mithilfe von Shell-Scripts erstellt werden.

Dieses Treiberprogramm könnte das Vergleichstestprogramm ausführen, ihm dabei die richtigen Parameter übergeben, den Test durch mehrere Durchläufe führen, die Umgebung in einen konsistenten Zustand zurückversetzen, den nächsten Test mit neuen Parameterwerten vorbereiten und die Testdaten sammeln bzw. konsolidieren. Diese Treiberprogramme können so flexibel gestaltet werden, dass sie zur Ausführung einer ganzen Reihe von Vergleichstests, zur Analyse der Ergebnisse und zur Erstellung eines Berichts über die endgültigen und optimalen Parameterwerte für einen bestimmten Test verwendet werden könnten.

Vergleichstestanalyse - Beispiel

Die Ausgabe des Vergleichstestprogramms sollte eine Kennung für jeden Test (Test Number), die Iteration (Durchlauf) der Programmausführung (Iteration Number), die Anweisungsnummer (Statement Number) und die für die Ausführung erfasste Zeit (Timing) enthalten.

Eine Übersicht über Vergleichstestergebnisse nach einer Reihe von Messungen könnte wie folgt aussehen:

```
Test     Iter.   Stmt    Timing         SQL Statement
Numbr    Numbr   Numbr   (hh:mm:ss.ss)
 002      05      01     00:00:01.34    CONNECT TO SAMPLE
 002      05      10     00:02:08.15    OPEN cursor_01
 002      05      15     00:00:00.24    FETCH cursor_01
 002      05      15     00:00:00.23    FETCH cursor_01
 002      05      15     00:00:00.28    FETCH cursor_01
 002      05      15     00:00:00.21    FETCH cursor_01
 002      05      15     00:00:00.20    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      20     00:00:00.84    CLOSE cursor_01
 002      05      99     00:00:00.03    CONNECT RESET
```

Abbildung 20. Beispielergebnisse eines Vergleichstestprogramms

Anmerkung: Die Daten im gezeigten Bericht dienen nur der Illustration. Sie stellen **keine** durch Tests ermittelten Ergebnisse dar.

Eine Analyse zeigt, dass die Anweisung CONNECT (Anweisung 01) 1,34 Sekunden dauerte, die Anweisung OPEN CURSOR (Anweisung 10) 2 Minuten und 8,15 Sekunden, die Anweisungen FETCH (Anweisung 15) sieben Zeilen mit der längsten Verzögerung von 0,28 Sekunden lieferten, die Anweisung CLOSE CURSOR (Anweisung 20) 0,84 Sekunden benötigte und die Anweisung CONNECT RESET (Anweisung 99) 0,03 Sekunden in Anspruch nahm.

Wenn das Programm die Daten in einem DEL-Format (Delimited ASCII) ausgeben könnte, könnten diese später in eine Datenbanktabelle oder ein Arbeitsblatt zur weiteren statistischen Analyse importiert werden.

Eine Beispielausgabe für einen Vergleichstestbericht könnte folgendermaßen aussehen:

```
PARAMETER        VALUES FOR EACH BENCHMARK TEST
TEST NUMBER      001     002     003    004     005
locklist         63      63      63      63      63
maxappls         8       8       8       8       8
applheapsz       48      48      48      48      48
dbheap           128     128     128     128     128
sortheap         256     256     256     256     256
maxlocks         22      22      22      22      22
stmtheap         1024    1024    1024    1024    1024
SQL STMT         AVERAGE TIMINGS (seconds)
  01             01.34   01.34   01.35   01.35   01.36
  10             02.15   02.00   01.55   01.24   01.00
  15             00.22   00.22   00.22   00.22   00.22
  20             00.84   00.84   00.84   00.84   00.84
  99             00.03   00.03   00.03   00.03   00.03
```

Abbildung 21. Beispielbericht zu den vom Vergleichstest ermittelten Ausführungszeiten

Anmerkung: Die Daten im gezeigten Bericht dienen nur der Illustration. Sie stellen **keine** durch Tests ermittelten Ergebnisse dar.

Kapitel 17. Designadvisor

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 erweisen. 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 MQTs (Materialized Query Tables)
- Hinzufügen von Clusterindizes
- 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

Beim Entwurf Ihrer Datenbank können Sie den Designadvisor zu folgenden Zwecken einsetzen:

- Generieren von Entwurfsalternativen in einer Testumgebung für eine partitionierte Datenbankumgebung sowie für Indizes, MQTs und MDC-Tabellen.
- Für partitionierte Datenbankumgebungen kann der Designadvisor mit folgenden Zielen verwendet werden:
  - Ermitteln der Strategie für eine Datenbankpartitionierung vor dem Laden von Daten in eine Datenbank.
  - Unterstützen der Migration von einer DB2-Einzelpartitionsdatenbank auf eine DB2-Datenbank mit mehreren Partitionen.
  - Unterstützen der Migration von einem anderen Datenprodukt auf eine DB2-Datenbank mit mehreren Partitionen.
- Bewerten von Indizes, MQTs, MDC-Tabellen oder Strategien für eine Datenbankpartitionierung, die manuell generiert wurden.

Optimieren der Auslastungsleistung

Nach der Einrichtung Ihrer Datenbank können Sie den Designadvisor zu folgenden Zwecken einsetzen:

- Verbessern der Leistung einer bestimmten Anweisung oder Auslastung
- Verbessern der allgemeinen Datenbankleistung unter Verwendung der Leistung einer Musterauslastung als Messgröße

- Verbessern der Leistung der am häufigsten ausgeführten Abfragen, wie sie zum Beispiel durch den Aktivitätsmonitor ermittelt werden
- Bestimmen, wie sich die Leistung einer neuen kritischen Abfrage optimieren lässt
- Reagieren auf Empfehlungen der Diagnosezentrale im Hinblick auf Probleme mit dem gemeinsamen Dienstprogrammspeicher oder dem Sortierspeicher bei einer sortierintensiven Auslastung
- Ermitteln von Objekten, die in keiner Auslastung verwendet werden

Ausgabe des Designadvisors

Über die grafische Benutzerschnittstelle (GUI) des Designadvisors können Sie Empfehlungen aus dem Designadvisor anzeigen, speichern oder implementieren. Wenn Sie den Designadvisor über die Befehlszeile ausführen, wird die Ausgabe standardmäßig an die Standardausgabeeinheit (stdout) gesendet und in den Tabellen ADVISE_TABLE und ADVISE_INDEX gespeichert.

- Die Tabelle ADVISE_TABLE enthält USE_TABLE='Y' für MQT- und MDC-Tabellen sowie Empfehlungen zu Strategien für die Datenbankpartitionierung.

  Die MQT-Empfehlungen sind außerdem in der Tabelle ADVISE_MQT, die MDC-Empfehlungen in der Tabelle ADVISE_TABLE und die Empfehlungen zu Strategien für die Datenbankpartitionierung in der Tabelle ADVISE_PARTITION zu finden. Der Wert RUN_ID in diesen Tabellen entspricht dem Wert START_TIME in der Tabelle ADVISE_INSTANCE für jede Ausführung des Designadvisors.
- Die Tabelle ADVISE_INDEX enthält USE_INDEX='Y' oder 'R' für Indexempfehlungen.

Die Tabelle ADVISE_INSTANCE wird ebenfalls mit einer Zeile pro Ausführung des Designadvisors aktualisiert:

- Das Feld START_TIME und das Feld END_TIME zeigen die Start- bzw. Stoppzeiten des Dienstprogramms.
- In das Feld STATUS wird der Wert 'COMPLETED' eingetragen, wenn das Dienstgramm erfolgreich beendet wird.
- Das Feld MODE zeigt an, ob die Option -m verwendet wurde.
- Das Feld COMPRESSION gibt den Typ der verwendeten Komprimierung an.

Sie können die Empfehlungen des Designadvisors über die Option -o in einer Datei speichern. Die gespeicherte Ausgabe des Designadvisors besteht aus den folgenden Elementen:

- Die CREATE-Anweisungen, die für neue Indizes, MQTs, Strategien für die Datenbankpartitionierung und MDC-Tabellen angegeben werden
- REFRESH-Anweisungen für MQTs
- RUNSTATS-Befehle für neue Objekte
- Empfohlene, bereits vorhandene MQTs, sofern sie zur Ausführung der Auslastung verwendet wurden und werden

  Anmerkung: Die Spalte COLSTATS der Tabelle ADVISE_MQT enthält die Spaltenstatistiken für eine MQT. Die Statistiken besitzen eine XML-Struktur wie die folgende:

```
<?xml version=\"1.0\" encoding=\"USASCII\"?>
<colstats>
    <column>
         <name>COLNAME1</name>
         <colcard>1000</colcard>
```

```
        <high2key>999</high2key>
        <low2key>2</low2key>
    </column>
    ....

    <column>
        <name>COLNAME100</name>
        <colcard>55000</colcard>
        <high2key>49999</high2key>
        <low2key>100</low2key>
    </column>
</colstats>
```

Die Basistabelle, für die ein Index definiert wird, wird ebenfalls hinzugefügt. Eine Rangfolge von Indizes und MQTs kann anhand des Werts 'benefit' ermittelt werden. Sie können eine Rangfolge für MQTs auch anhand der Formel 'benefit - 0,5 * overhead' und für Indizes anhand der Formel 'benefit - overhead' ermitteln. Nach der Liste von Indizes und MQTs folgt die Liste von Anweisungen in der Auslastung, die den SQL-Text, die Anweisungsnummer der Anweisung, die geschätzte Leistungsverbesserung (benefit) aus den Empfehlungen sowie die Liste von Tabellen, Indizes und MQTs enthält, die von der jeweiligen Anweisung verwendet wurden.

Anmerkung: Die ursprünglichen Abstände im SQL-Text werden in dieser Ausgabe beibehalten, jedoch wird der SQL-Text aus Gründen der Lesbarkeit in Kommentarzeilen von 80 Zeichen aufgeteilt.
MDC und Partitionierung werden in dieser XML-Ausgabe nicht explizit angezeigt. Ein Beispiel dieser Ausgabe sieht folgendermaßen aus:

```
--<?xml version="1.0"?>
--<design-advisor>
--<mqt>
--<identifier>
--<name>MQT612152202220000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<statementlist>3</statementlist>
--<benefit>1013562.481682</benefit>
--<overhead>1468328.200000</overhead>
--<diskspace>0.004906</diskspace>
--</mqt>
.....
--<index>
--<identifier>
--<name>IDX612152221400000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<table><identifier>
--<name>PART</name>
--<schema>TPCD </schema>
--</identifier></table>
--<statementlist>22</statementlist>
--<benefit>820160.000000</benefit>
--<overhead>0.000000</overhead>
--<diskspace>9.063500</diskspace>
--</index>
.....
--<statement>
--<statementnum>11</statementnum>
--<statementtext>
--
-- select
-- c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice,
-- sum(l_quantity) from tpcd.customer, tpcd.orders,
-- tpcd.lineitem where o_orderkey in( select
```

```
-- l_orderkey from tpcd.lineitem group by l_orderkey
-- having sum(l_quantity) > 300 ) and c_custkey
-- = o_custkey and o_orderkey = l_orderkey group by
-- c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
-- order by o_totalprice desc, o_orderdate fetch first
-- 100 rows only
--</statementtext>
--<objects>
--<identifier>
--<name>MQT612152202490000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>ORDERS</name>
--<schema>TPCD </schema>
--</identifier>
--<identifier>
--<name>CUSTOMER</name>
--<schema>TPCD </schema>
--</identifier>
--<identifier>
--<name>IDX612152235020000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>IDX612152235030000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>IDX612152211360000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--</objects>
--<benefit>2091459.000000</benefit>
--<frequency>1</frequency>
--</statement>
```

Beachten Sie, dass die XML-Struktur mehr als eine Spalte beinhalten kann. Für jede Spalte wird die Spaltenkardinalität (d. h. die Anzahl der Werte in der Spalte) sowie optional die Werte für HIGH2KEY und LOW2KEY angegeben.

Wenn MQT-, Datenbankpartitionierungs- oder MDC-Empfehlungen angegeben werden, wird der relevante Befehl zum Aufrufen der gespeicherten Prozedur ALTER TABLE in der Spalte ALTER_COMMAND von ADVISE_TABLE positioniert.

Anmerkung: Der Befehl zum Aufrufen der gespeicherten Prozedur ALTER TABLE schlägt möglicherweise aufgrund von Tabelleneinschränkungen für die gespeicherte Prozedur ALTOBJ fehl.

Nach einigen kleineren Modifikationen können Sie diese Ausgabedatei als Script für den Befehlszeilenprozessor (CLP) zur Erstellung der empfohlenen Objekte verwenden. Folgende Modifikationen bieten sich eventuell an:

- Kombinieren aller RUNSTATS-Befehlsanweisungen zu einem einzigen RUNSTATS-Aufruf für die neuen oder modifizierten Objekte
- Angeben geeigneterer Objektnamen als die systemgenerierten IDs
- Entfernen aller DDL-Anweisungen für Objekte, die nicht sofort implementiert werden sollen, aus dem Code durch Löschen oder Verwenden von Kommentarzeichen

Verwenden des Designadvisors

Sie können den Designadvisor über die Befehlszeile ausführen oder die grafische Benutzerschnittstelle (GUI) des Designadvisors in der Steuerzentrale verwenden.

- **In der Steuerzentrale:**
  1. Definieren Sie Ihre Auslastung. Siehe „Definieren einer Auslastung für den Designadvisor".
  2. Öffnen Sie die Steuerzentrale.
  3. Führen Sie den Designadvisor über die Steuerzentrale aus, indem Sie die gewünschte Datenbank oder einen bestimmten Index in der Datenbank mit der rechten Maustaste anklicken und **Designadvisor** im Kontextmenü auswählen.
  4. Wenn der Designadvisor die Generierung von Empfehlungen abgeschlossen hat, können Sie auswählen, ob diese Empfehlungen in einem Bericht gespeichert werden sollen oder ob der Designadvisor einige oder alle der Empfehlungen für Sie implementieren soll.
- **Über die Befehlzeile:**
  1. Definieren Sie Ihre Auslastung. Siehe „Definieren einer Auslastung für den Designadvisor".
  2. Führen Sie den Befehl db2advis für die Auslastung aus.

     Anmerkung: Wenn die Statistikdaten für Ihre Datenbank keinen aktuellen Stand haben, sind die generierten Empfehlungen weniger zuverlässig.
  3. Interpretieren Sie die Ausgabe des Befehls db2advis, und nehmen Sie alle erforderlichen Modifikationen vor.
  4. Implementieren Sie die ausgewählten Empfehlungen des Designadvisors.

Definieren einer Auslastung für den Designadvisor

Eine *Auslastung* ist eine Gruppe von SQL-Anweisungen, die der Datenbankmanager in einem bestimmten Zeitraum verarbeiten muss. Zum Beispiel könnte der Datenbankmanager in einem Monat 1000 INSERT-, 10000 UPDATE-, 10000 SELECT- und 1000 DELETE-Anweisungen verarbeiten müssen. Der Designadvisor analysiert eine angegebene Auslastung und zieht Faktoren, wie den Typ der Auslastungsanweisungen, die Häufigkeit, mit der eine bestimmte Anweisung auftritt, sowie Merkmale Ihrer Datenbank in Betracht, um Empfehlungen zu generieren, die den Gesamtaufwand zur Ausführung der Auslastung minimieren.

Über die Seite „Auslastung" der grafischen Benutzerschnittstelle des Designadvisors können Sie eine neue Auslastungsdatei erstellen oder eine bereits vorhandene Auslastungsdatei modifizieren. Sie können Anweisungen in die Datei aus verschiedenen Quellen importieren:

- Eine Textdatei mit Begrenzern
- Eine Ereignismonitortabelle
- Query Patroller-Protokolldatentabellen durch die Option -qp über die Befehlszeile
- Mit EXPLAIN bearbeitete Anweisungen in der Tabelle EXPLAINED_STATEMENT
- Neue SQL-Anweisungen, die mit einer DB2-Momentaufnahme erfasst wurden
- Workload-Manager-Aktivitätstabellen

- Workload-Manager-Ereignismonitortabellen durch Angabe der Option -wlm über die Befehlszeile

Nach dem Importieren Ihrer SQL-Anweisungen können Sie Anweisungen hinzufügen, ändern, modifizieren oder entfernen sowie die Häufigkeit der Anweisungen modifizieren.

Über die Befehlszeile führen Sie den Designadvisor wie folgt aus:

- Mit einer einzigen SQL-Anweisung, die Sie in der Befehlzeile zusammen mit dem Befehl angeben
- Mit einer Gruppe dynamischer SQL-Anweisungen, die in einer DB2-Momentaufnahme erfasst wurden
- Mit einer Gruppe von SQL-Anweisungen, die in einer Auslastungsdatei enthalten sind
- Gehen Sie wie folgt vor, um den Designadvisor für dynamische SQL-Anweisungen auszuführen:
  1. Setzen Sie den Datenbankmonitor mit dem folgenden Befehl zurück:
     ```
     db2 reset monitor for database datenbankname
     ```
  2. Warten Sie einen geeigneten Zeitraum für die Ausführung dynamischer SQL-Anweisungen für die Datenbank ab.
  3. Führen Sie den Befehl db2advis mit der Option -g aus. Wenn Sie die dynamischen SQL-Anweisungen in der Tabelle ADVISE_WORKLOAD zur späteren Referenz speichern wollen, verwenden Sie außerdem die Option -p.
- Gehen Sie wie folgt vor, um den Designadvisor für eine Gruppe von SQL-Anweisungen auszuführen, die in einer Auslastungsdatei enthalten sind:
  1. Erstellen Sie manuell eine Auslastungsdatei, indem Sie jede SQL-Anweisung mit einem Semikolon trennen, oder importieren Sie SQL-Anweisungen aus einer oder mehreren der oben aufgeführten Quellen.
  2. Legen Sie die Häufigkeit der Anweisungen in der Auslastung fest. Jeder Anweisung in der Auslastungsdatei wird standardmäßig die Häufigkeit 1 zugeordnet. Die Häufigkeit einer SQL-Anweisung stellt die Anzahl der Vorkommen der Anweisung innerhalb der Auslastung im Verhältnis zur Anzahl der Vorkommen anderer Anweisungen dar. Eine bestimmte SELECT-Anweisung könnte zum Beispiel 100-mal in einer Auslastung vorkommen, während eine andere SELECT-Anweisung 10-mal vorkommt. Zur Darstellung der relativen Häufigkeit dieser beiden Anweisungen würden Sie der ersten SELECT-Anweisung eine Häufigkeit von 10 und der zweiten SELECT-Anweisung eine Häufigkeit von 1 zuordnen. Sie können die Häufigkeit bzw. die Wertigkeit einer bestimmten Anweisung in der Auslastung manuell ändern, indem Sie die folgende Zeile nach der Anweisung einfügen: - - # SET FREQUENCY *n*. Dabei ist *n* der Häufigkeitswert, den Sie der Anweisung zuordnen wollen.
  3. Geben Sie den Befehl db2advis mit der Option -i gefolgt vom Namen der Auslastungsdatei ein.
- Zur Ausführung des Designadvisors für eine Auslastung, die in der Tabelle ADVISE_WORKLOAD enthalten ist, geben Sie den Befehl db2advis mit der Option -w gefolgt vom Namen der Auslastung ein.

Verwenden des Designadvisors zur Migration von einer Datenbank mit einer einzelnen Partition auf eine Datenbank mit mehreren Partitionen

Sie können den Designadvisor zur Unterstützung einer Migration von einer Datenbank mit einer Einzelpartition auf eine Datenbank mit mehreren Partitionen verwenden. Der Designadvisor kann Ihnen Empfehlungen für die Verteilung Ihrer Daten und gleichzeitig Empfehlungen für neue Indizes, MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle), und MDC-Tabellen (mit mehrdimensionalem Clustering) geben.

1. Registrieren des Lizenzschlüssels für ein DB2-Produkt oder -Feature mit dem Befehl db2licm
2. Erstellen Sie mindestens einen Tabellenbereich in einer Partitionsgruppe einer Datenbank mit mehreren Partitionen.

   Anmerkung: Da der Designadvisor nur eine Umverteilung für vorhandene Tabellenbereiche empfehlen kann, müssen die Tabellenbereiche, die der Designadvisor in Betracht ziehen soll, in der partitionierten Datenbank vorhanden sein, bevor der Designadvisor ausgeführt wird.
3. Führen Sie den Designadvisor mit ausgewählter Datenbankpartitionierungsfunktion über die Designadvisor-GUI oder unter Angabe der Partitionierungsoption mit dem Befehl db2advis aus.
4. Wenn Sie mit dem Designadvisor in der Steuerzentrale arbeiten, können Sie die Datenbankpartitionierungsempfehlungen automatisch implementieren. Wenn Sie den Befehl db2advis verwenden, müssen Sie die db2advis-Ausgabedatei geringfügig modifizieren, bevor Sie die vom Designadvisor generierten DDL-Anweisungen ausführen.

Begrenzungen und Einschränkungen des Designadvisors

1. **Einschränkungen für Indexempfehlungen**
   - Indizes, die für MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle) empfohlen werden, dienen zur Verbesserung der Auslastungsleistung, und nicht der REFRESH TABLE-Leistung. Wenn außerdem Aktualisierungs-, Einfüge- und Löschoperationen kein Bestandteil der Auslastung sind, würde die Leistung einer Änderung (z. B. Aktualisierung) der MQT für mit IMMEDIATE definierte MQTs nicht berücksichtigt.
   - Der RID-Clusterindex wird nur empfohlen, wenn mehrdimensionales Clustering auszuwählen ist. Der Designadvisor schließt RID-Clusterindizes als Option anstelle der Erstellung einer MDC-Struktur für eine Tabelle mit ein.
2. **Einschränkungen für MQT-Empfehlungen**
   - Der Designadvisor empfiehlt keine inkrementellen gespeicherten MQTs. Wenn Sie inkrementelle MQTs erstellen wollen, können Sie mit REFRESH IMMEDIATE definierte MQTs nehmen und diese in inkrementelle Abfragetabellen mit einer eigenen Auswahl an Zwischenspeichertabellen konvertieren.
   - Indizes, die für MQTs empfohlen werden, dienen zur Verbesserung der Auslastungsleistung, und nicht der REFRESH-Leistung für MQTs.
   - Wenn Aktualisierungs-, Einfüge- und Löschoperationen kein Bestandteil der angegebenen Auslastung sind, wird der Leistungsaufwand zur Aktualisierung einer empfohlenen, mit REFRESH IMMEDIATE definierten MQT nicht berücksichtigt. Es wird empfohlen, für MQTs mit REFRESH IMMEDIATE

eindeutige Indizes für den implizierten eindeutigen Schlüssel zu erstellen, der sich aus den Spalten in der GROUP BY-Klausel der Definition der MQT zusammensetzt.

3. **Einschränkungen für MDC-Empfehlungen**
   - Vorhandene Tabellen müssen mit einer repräsentativen Datenmenge gefüllt sein. Ansonsten zieht der Designadvisor kein multidimensionales Clustering (MDC) für die Tabelle in Betracht. Ein Minimum von zwei bis drei Dutzend MB Daten wird empfohlen. Tabellen, die kleiner als 12 EXTENTSIZE große Speicherbereiche sind, werden immer ausgeschlossen.
   - MDC-Empfehlungen für neue MQTs werden nicht in Betracht gezogen, sofern nicht die Stichprobenoption -r mit dem Befehl verwendet wird oder die MQT-Stichprobenerstellung im GUI-Tool ausgewählt wird.
   - Der Designadvisor gibt keine MDC-Empfehlungen für typisierte oder temporäre Tabellen.
   - Der Designadvisor gibt keine MDC-Empfehlungen für föderierte Tabellen.
   - Der Speicherplatz (d. h. ca. 1 % der Tabellendaten für umfangreiche Tabellen) für die Stichprobendaten, der während der Ausführung des Designadvisors verwendet wird, muss vorhanden sein. Ansonsten wird die Stichprobentabelle nur für Basisspalten unter der Annahme unkorrelierter Daten untersucht. In diesem Fall wird eine Warnung generiert.
   - Tabellen ohne erfasste Statistiken werden bei der Analyse übersprungen.
   - Der Designadvisor gibt keine Empfehlungen für mehrspaltige Dimensionen.
   - Vorhandene Tabellen müssen Daten enthalten, damit die Stichprobenerstellung bei der MDC-Auswahl funktioniert.
4. **Einschränkungen für Datenbankpartitionierungsempfehlungen**

   Der Designadvisor kann eine Datenbankpartitionierung nur unter DB2 Enterprise Server Edition empfehlen. Wenn die Datenbankpartitionierungsoptionen mit dem Befehl db2advis angegeben werden, wird ein Fehler zurückgegeben. In der grafischen Benutzerschnittstelle (GUI) des Designadvisors kann die Datenbankpartitionierungsfunktion in einer Einzelpartitionsdatenbankumgebung nicht ausgewählt werden.
5. **Weitere Einschränkungen**

   Während der Ausführung des Designadvisors werden Simulationskatalogtabellen erstellt. Diese Tabellen werden gelöscht, wenn die Ausführung des Designadvisors beendet ist. Eine unvollständige Ausführung des Designadvisors kann dazu führen, dass einige dieser Tabellen nicht gelöscht werden. In diesem Fall können Sie den Designadvisor dazu verwenden, diese Tabellen zu löschen, indem Sie das Dienstprogramm über die Befehlszeile erneut starten. Zum Entfernen der Simulationskatalogtabellen geben Sie die Option -f und die Option -n an (geben Sie mit -n den gleichen Benutzernamen ein, der bei der unvollständigen Ausführung verwendet wurde). Wenn Sie die Option -f nicht angeben, generiert der Designadvisor die DROP-Anweisungen, die zum Entfernen der Tabellen benötigt werden.

   Anmerkung: In Version 9.5 ist die Option -f die Standardoption. Das bedeutet, dass der Datenbankmanager bei Ausführung von db2advis mit der MQT-Auswahl automatisch alle lokalen Simulationskatalogtabellen mit derselben Benutzer-ID als Schemanamen löscht.

   Sie sollten einen getrennten Tabellenbereich zum Speichern dieser simulierten Katalogtabellen erstellen und DROP TABLE RECOVERY auf den Wert OFF setzen. Dies ermöglicht eine einfachere Bereinigung und eine schnellere Ausführung des Designadvisors. Der getrennte Tabellenbereich für die Simulationskatalogtabellen muss nur auf dem Katalogknoten definiert werden.

Teil 5. Optimieren der Datenbankanwendungsleistung

Es gibt eine Reihe von Überlegungen zu Anwendungen und Umgebungen, die Sie überprüfen sollten, um die Leistung von Datenbankanwendungen zu maximieren. Führen Sie die Schritte unten aus, um die Leistung von Datenbankanwendungen zu optimieren:

1. Testen Sie Ihre Anwendung, und messen Sie die Leistung.
2. Überprüfen Sie wichtige Überlegungen zu Anwendungen, die sich auf die Leistung auswirken:
   - Isolationsstufen und Steuerung des gemeinsamen Zugriffs
   - „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 185
   - „Faktoren mit Auswirkungen auf Sperren" auf Seite 226
3. Überprüfen Sie wichtige Richtlinien, die sich auf die Optimierung von Anwendungen beziehen:
   - „Richtlinien zur Begrenzung von SELECT-Anweisungen" auf Seite 230
   - „Angeben von Zeilenblockung zur Verringerung des Systemaufwands" auf Seite 234
   - „Verbessern der Leistung durch Binden mit REOPT" auf Seite 236
   - „Stichprobendaten in SQL- und XQuery-Abfragen" auf Seite 236
   - „Zeichenkonvertierung" im Handbuch *Internationalisierung*
   - „Parallelverarbeitung für Anwendungen" auf Seite 238
   - „Profile und Richtlinien für das Optimierungsprogramm - Übersicht" auf Seite 426
   - „Konfigurationsparameter mit Einfluss auf die Abfrageoptimierung" im Handbuch *Datenserver, Datenbanken und Datenbankobjekte*
4. Modifizieren Sie Ihre Anwendung gemäß den Überlegungen und Richtlinien. Führen Sie die empfohlenen Tasks aus, die in den Überlegungen und Richtlinien aufgeführt sind. Führen Sie insbesondere die folgenden Tasks aus:
   - „Auswählen von Optimierungsklassen" auf Seite 422
   - „Angeben der Isolationsstufe" auf Seite 182
   - „Verhindern von auf Sperren bezogenen Leistungsproblemen" auf Seite 203
5. Erfassen und aktualisieren Sie Ihre Datenbankstatistikdaten.
6. Ermitteln Sie, ob Sie Ihre Tabellen und Indizes reorganisieren müssen.
7. Testen Sie Ihre Anwendung, und vergleichen Sie die Leistung mit der vorherigen Leistung.

Kapitel 18. Hinweise zu Anwendungen

Aspekte des gemeinsamen Zugriffs

Da viele Benutzer auf Daten in einer relationalen Datenbank zugreifen und sie ändern, muss der Datenbankmanager in der Lage sein, einerseits den Benutzern diese Änderungen zu ermöglichen, und andererseits sicherzustellen, dass die Datenintegrität gewahrt bleibt. Der Begriff *gemeinsamer Zugriff* bezeichnet die Möglichkeit, dass mehrere interaktive Benutzer oder Anwendungsprogramme die Ressourcen zur gleichen Zeit verwenden können. Der Datenbankmanager steuert diesen Zugriff, um unerwünschte Folgen zu vermeiden. Solche Folgen sind zum Beispiel:

- **Verlorene Aktualisierungen**. Zwei Anwendungen A und B lesen beispielsweise dieselbe Zeile aus einer Datenbank und berechnen aufgrund der Daten, die von den Anwendungen gelesen werden, neue Werte für eine ihrer Spalten. Die Anwendung A aktualisiert die Zeile mit ihrem neuen Wert und anschließend aktualisiert B die Zeile ebenfalls, sodass die von A durchgeführte Aktualisierung verloren geht.
- **Zugriff auf nicht festgeschriebene Daten**. Anwendung A könnte zum Beispiel einen Wert in der Datenbank aktualisieren, den Anwendung B liest, bevor er festgeschrieben wurde. Wenn dann der Wert von der Anwendung A später nicht mit COMMIT festgeschrieben, sondern zurückgesetzt wird, basieren die Berechnungen der Anwendung B auf nicht festgeschriebenen (und daher wahrscheinlich ungültigen) Daten.
- **Nichtwiederholbare Lesevorgänge**. Einige Anwendungen bewirken die folgende Reihenfolge von Ereignissen: Anwendung A liest eine Zeile aus der Datenbank und verarbeitet anschließend weitere Anforderungen. In der Zwischenzeit ändert oder löscht Anwendung B die Zeile und schreibt diese Aktion fest. Später, wenn Anwendung A versucht, die ursprüngliche Zeile erneut zu lesen, empfängt sie die geänderte Zeile oder stellt fest, dass die ursprüngliche Zeile gelöscht wurde.
- **Das Phänomen der Phantomzeilen**. Phantomzeilen treten unter folgenden Umständen auf:
  1. Eine Anwendung führt eine Abfrage aus, die eine Menge von Zeilen nach einer Suchbedingung liest.
  2. Eine andere Anwendung fügt neue Daten ein oder aktualisiert vorhandene Daten, die die Abfragebedingungen der ersten Anwendung erfüllen würden.
  3. Die erste Anwendung wiederholt die Abfrage aus Schritt 1 (innerhalb derselben UOW (Unit of Work, Arbeitseinheit)).

     Einige zusätzliche Zeilen („Phantomzeilen") werden als Teil der Ergebnismenge zurückgegeben, die bei der ersten Ausführung der Abfrage (Schritt 1) nicht zurückgegeben wurden.

Anmerkung: Bei deklarierten temporären Tabellen gibt es keine Probleme bezüglich des gemeinsamen Zugriffs, da sie nur für die Anwendung verfügbar sind, die sie deklariert hat. Diese Art von Tabelle besteht nur von dem Zeitpunkt ihrer Deklaration durch die Anwendung bis zu dem Zeitpunkt, zu dem die Anwendung beendet wird oder die Verbindung trennt.

Steuerung des gemeinsamen Zugriffs in Systemen föderierter Datenbanken

Ein *System föderierter Datenbanken* unterstützt von Anwendungen und Benutzern übergebene SQL-Anweisungen, die auf zwei oder mehr Datenbankverwaltungssysteme (DBMS) bzw. Datenbanken innerhalb einer Anweisung verweisen. Als Verweise auf die Datenquellen, die aus einem DBMS und Daten bestehen, verwendet DB2 so genannte *Kurznamen*. Kurznamen sind Aliasnamen für Objekte in anderen Datenbankmanagern. In einem föderierten System verlässt sich DB2 auf die Protokolle zur Steuerung des gemeinsamen Zugriffs des Datenbankmanagers, der die angeforderten Daten bereitstellt.

In einem föderierten DB2-System besteht für Datenbankobjekte *Positionstransparenz*. Durch die Positionstransparenz ist es möglich, wenn Informationen von Tabellen und Sichten an andere Positionen versetzt werden, die Verweise auf diese Informationen über Kurznamen zu aktualisieren, ohne die Anwendungen ändern zu müssen, die diese Informationen anfordern. Wenn eine Anwendung auf Daten über Kurznamen zugreift, verlässt sich DB2 darauf, dass die Protokolle zur Steuerung des gemeinsamen Zugriffs der Datenbankmanager der Datenquellen die Isolationsstufen sicherstellen. Obwohl DB2 versucht, die angeforderte Isolationsstufe an der Datenquelle mit einem logischen Äquivalent abzugleichen, können die Ergebnisse je nach Funktionsspektrum der Datenquelle unterschiedlich sein.

Isolationsstufen und Leistung

Eine *Isolationsstufe* legt fest, wie Daten gesperrt oder von anderen Prozessen während des Zugriffs auf die Daten isoliert werden. Die Isolationsstufe ist während der Dauer der UOW (Unit of Work, Arbeitseinheit) in Kraft. Anwendungen, die einen Cursor verwenden, der mit der Anweisung DECLARE CURSOR und mit der Klausel WITH HOLD deklariert wurde, behalten die gewählte Isolationsstufe für die Dauer der UOW bei, in der die Anweisung OPEN CURSOR durchgeführt wurde. DB2 unterstützt die folgenden Isolationsstufen:

- Wiederholtes Lesen (Repeatable Read)
- Lesestabilität (Read Stability)
- Cursorstabilität (Cursor Stability)
- Nicht festgeschriebener Lesevorgang (Uncommitted Read, UR)

Anmerkung: Einige Hostdatenbankserver unterstützen die Isolationsstufe *kein Commit* (No Commit). In anderen Datenbanken verhält sich diese Isolationsstufe wie die Isolationsstufe *Nicht festgeschriebener Lesevorgang* (Uncommited Read).

Im Folgenden sind detaillierte Erläuterungen zu den Isolationsstufen aufgeführt. Diese sind in absteigender Reihenfolge bezüglich der Auswirkungen auf die Leistung, aber in aufsteigender Reihenfolge bezüglich der Vorsicht geordnet, die beim Zugriff auf und Aktualisieren von Daten erforderlich ist.

Wiederholtes Lesen (Repeatable Read)

Die Isolationsstufe *Wiederholtes Lesen* (RR - Repeatable Read) sperrt alle Zeilen, auf die eine Anwendung innerhalb einer UOW verweist. Bei Verwendung der Isolationsstufe RR liefert eine Anweisung SELECT, die von einer Anwendung zweimal innerhalb derselben UOW, in der der Cursor geöffnet wurde, ausgegeben wird, jedes Mal dasselbe Ergebnis. Bei der Isolationsstufe RR können verlorene Aktualisierungen, Zugriffe auf nicht festgeschriebene Daten und Phantomzeilen nicht auftreten.

Eine Anwendung mit der Isolationsstufe RR kann, bis die UOW beendet wird, so oft wie erforderlich Zeilen abrufen und Aktionen an ihnen vornehmen. Andere Anwendungen können jedoch keine Zeile aktualisieren, löschen oder einfügen, die sich auf die Ergebnistabelle auswirken würde, bis die UOW beendet ist. Für Anwendungen mit der Isolationsstufe RR sind nicht festgeschriebene Änderungen anderer Anwendungen nicht sichtbar.

Durch die Isolationsstufe RR werden alle Zeilen, auf die verwiesen wird, und nicht nur die Zeilen, die abgerufen werden, gesperrt. Die Sperrung wird so ausgeführt, dass eine andere Anwendung keine Zeile einfügen oder aktualisieren kann, die der Liste von Zeilen, auf die in einer Abfrage zugegriffen wird, hinzugefügt würde, wenn diese Abfrage erneut ausgeführt würde. Dadurch wird das Auftreten von Phantomzeilen verhindert. Wenn Sie zum Beispiel 10.000 Zeilen unter Verwendung von Vergleichselementen durchsuchen, werden alle 10.000 Zeilen gesperrt, auch wenn letzten Endes nur 10 Zeilen den Vergleichselementen entsprechen.

Anmerkung: Durch die Isolationsstufe RR wird sichergestellt, dass alle zurückgelieferten Daten bis zu dem Zeitpunkt, zu dem die Daten für die Anwendung sichtbar werden, unverändert bleiben, auch wenn temporäre Tabellen oder Zeilenblockung verwendet werden.

Da die Isolationsstufe RR eine beträchtliche Anzahl an Sperren anfordern kann, können diese Sperren die Anzahl von Sperren, die durch die Konfigurationsparameter *locklist* und *maxlocks* festgelegt wird, überschreiten. Um eine Sperreneskalation zu vermeiden, kann das Optimierungsprogramm eventuell von vornherein eine einzige Sperre auf Tabellenebene für eine Indexsuche anfordern, wenn das Auftreten einer Sperreneskalation wahrscheinlich ist. Dies funktioniert so, als würde der Datenbankmanager für Sie eine Anweisung LOCK TABLE absetzen. Wenn Sie keine Sperre auf Tabellenebene aktivieren lassen wollen, stellen Sie sicher, dass für die Transaktion ausreichend Sperren verfügbar sind, oder verwenden Sie die Isolationsstufe der Lesestabilität (RS).

Bei der Auswertung referenzieller Integritätsbedingungen kann DB2 unabhängig von der durch den Benutzer festgelegten Isolationsstufe in einigen Fällen die für die Suche in der untergeordneten Tabelle verwendete Isolationsstufe intern auf RR heraufsetzen. Dadurch werden zusätzliche Sperren bis zur COMMIT-Operation aktiviert, sodass die Wahrscheinlichkeit eines Deadlocks oder einer Sperrzeitlimitüberschreitung zunimmt. Um dies zu vermeiden, wird empfohlen, einen Index zu erstellen, der nur die Spalte bzw. Spalten des Fremdschlüssels enthält, sodass die RI-Suche diesen Index als Alternative nutzen kann.

Lesestabilität (RS)

Die Isolationsstufe *Lesestabilität* (RS - Read Stability) sperrt nur die Zeilen, die von einer Anwendung innerhalb einer UOW abgerufen werden. Sie stellt sicher, dass jede den Vergleichselementen entsprechende gelesene Zeile während einer UOW nicht von Prozessen anderer Anwendungen geändert wird und dass keine von einem Prozess einer anderen Anwendung geänderte Zeile gelesen wird, bis die Änderung von dem Prozess festgeschrieben wurde. Das heißt, „nichtwiederholbare Lesevorgänge" sind **nicht** möglich.

Anders als bei der Isolationsstufe RR (Wiederholtes Lesen) können bei der Isolationsstufe RS (Lesestabilität), wenn die Anwendung dieselbe Abfrage mehr als einmal absetzt, zusätzliche *Phantomzeilen* auftreten (*Phänomen der Phantomzeilen*). In dem bereits angeführten Beispiel der Suche über 10.000 Zeilen werden durch die Isolationsstufe RS nur die den Vergleichselementen entsprechenden Zeilen gesperrt.

Da durch die Abfrage nur 10 Zeilen abgerufen werden, werden unter der Isolationsstufe RS nur für diese zehn Zeilen Sperren aktiviert. Unter der Isolationsstufe RR (Wiederholtes Lesen) werden in diesem Beispiel hingegen sämtliche 10.000 Zeilen gesperrt. Bei den aktivierten Sperren kann es sich um Sperren der Modi Share, Next Share, Update oder Exclusive handeln.

Anmerkung: Durch die Isolationsstufe RS wird sichergestellt, dass alle zurückgelieferten Daten bis zu dem Zeitpunkt, zu dem die Daten für die Anwendung *sichtbar* werden, unverändert bleiben, auch wenn temporäre Tabellen oder Zeilenblockung verwendet werden.

Ein Zweck der Isolationsstufe RS besteht darin, sowohl einen hohen Grad des gemeinsamen Zugriffs als auch eine stabile Sicht der Daten zur Verfügung zu stellen. Um diesen Zweck zu erfüllen, stellt das Optimierungsprogramm sicher, dass Sperren auf Tabellenebene erst aktiviert werden, wenn eine Sperreneskalation auftritt.

Die Isolationsstufe RS eignet sich am besten für Anwendungen, die alle folgenden Merkmale aufweisen:
- Sie arbeiten in einer Umgebung mit gemeinsamem Zugriff.
- Sie fordern Zeilen an, die für die Dauer der UOW stabil bleiben müssen.
- Sie verwenden dieselbe Abfrage in einer UOW nur einmal, oder es ist nicht notwendig, dass bei mehrmaliger Verwendung der Abfrage innerhalb derselben UOW stets dieselben Abfrageergebnisse erzielt werden.

Cursorstabilität (CS)

Die Isolationsstufe *Cursorstabilität* (CS - Cursor Stability) sperrt jede Zeile, auf die von einer Transaktion einer Anwendung zugegriffen wird, während sich der Cursor auf der Zeile befindet. Diese Sperre bleibt aktiv, bis die nächste Zeile abgerufen oder die Transaktion beendet wird. Wenn jedoch Daten in einer Zeile geändert werden, muss die Sperre aktiv bleiben, bis die Änderung in der Datenbank festgeschrieben wurde.

Keine anderen Anwendungen können eine Zeile aktualisieren oder löschen, die von einer Anwendung mit der Isolationsstufe CS abgerufen wurde, während sich ein Aktualisierungscursor auf der Zeile befindet. Für Anwendungen mit der Isolationsstufe CS sind nicht festgeschriebene Änderungen anderer Anwendungen nicht sichtbar.

In dem bereits angeführten Beispiel einer Suche über 10.000 Zeilen heißt dies, dass unter der Isolationsstufe CS lediglich eine Sperre für die Zeile, die unter der aktuellen Cursorposition liegt, aktiv ist. Die Sperre wird entfernt, wenn der Cursor von der Zeile fortbewegt wird (sofern die Zeile nicht aktualisiert wurde).

Bei der Isolationsstufe CS können sowohl unterschiedliche Abfrageergebnisse innerhalb derselben UOW (d. h. nicht wiederholbare Lesevorgänge) als auch Phantomzeilen auftreten. Die Isolationsstufe CS ist die Standardisolationsstufe, die verwendet werden sollte, wenn es auf den höchstmöglichen Grad des gemeinsamen Zugriffs ankommt und nur festgeschriebene Zeilen anderer Anwendungen sichtbar sein sollen.

Nicht festgeschriebener Lesevorgang (Uncommitted Read, UR)

Die Isolationsstufe *Nicht festgeschriebener Lesevorgang* (UR - Uncommitted Read) erlaubt einer Anwendung den Zugriff auf nicht festgeschriebene Änderungen

anderer Transaktionen. Die Anwendung sperrt die Zeile, die sie liest, auch für keine andere Anwendung, sofern die andere Anwendung nicht versucht, die Tabelle zu löschen oder zu ändern. Die Isolationsstufe UR wirkt sich auf Lesecursor und Aktualisierungscursor unterschiedlich aus.

Cursor mit Lesezugriff können auf die meisten nicht festgeschriebenen Änderungen anderer Transaktionen zugreifen. Tabellen, Sichten und Indizes, die momentan von anderen Transaktionen erstellt oder gelöscht werden, sind während der Verarbeitung der Transaktion jedoch nicht verfügbar. Alle anderen Änderungen durch andere Transaktionen können gelesen werden, bevor sie mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht wurden.

Anmerkung: Aktualisierungscursor, die unter der Isolationsstufe UR arbeiten, verhalten sich genauso wie unter der Isolationsstufe CS (Cursorstabilität).

Wenn eine Anwendung ein Programm unter der Isolationsstufe UR ausführt, kann sie die Isolationsstufe CS verwenden. Dies liegt an der Mehrdeutigkeit der Cursor, die in den Anwendungsprogrammen verwendet werden. Mehrdeutige Cursor können durch eine BLOCKING-Option auf die Isolationsstufe CS eskaliert werden. Der Standardwert der BLOCKING-Option ist UNAMBIG. Das bedeutet, dass mehrdeutige Cursor als aktualisierbar behandelt werden und die Eskalation der Isolationsstufe auf CS durchgeführt wird. Zur Vermeidung dieser Eskalation haben Sie die folgenden beiden Möglichkeiten:

- Ändern Sie die Cursor im Anwendungsprogramm, sodass die Mehrdeutigkeit beseitigt wird. Ändern Sie die SELECT-Anweisungen, um die Klausel FOR READ ONLY hinzuzufügen.
- Belassen Sie mehrdeutige Cursor im Anwendungsprogramm. Kompilieren Sie jedoch das Programm mit den Optionen BLOCKING ALL und STATICREADONLY YES vor oder binden Sie es mit diesen Optionen, um zuzulassen, dass alle mehrdeutigen Cursor als reine Lesecursor behandelt werden, wenn das Programm ausgeführt wird.

Wie in dem für die Isolationsstufe RR genannten Beispiel einer Suche über 10.000 Zeilen werden bei der Isolationsstufe UR keine Zeilensperren aktiviert.

Bei der Isolationsstufe UR können sowohl unterschiedliche Leseergebnisse innerhalb derselben UOW (d. h. nicht wiederholbare Lesevorgänge) als auch Phantomzeilen auftreten. Die Isolationsstufe UR wird in der Regel für Abfragen auf Tabellen verwendet, die sich im Lesezugriff befinden, oder wenn Anweisungen SELECT ausgeführt werden und es dabei keine Rolle spielt, ob nicht festgeschriebene Daten anderer Anwendungen angezeigt werden.

Zusammenfassung der Isolationsstufen

Die folgende Tabelle fasst die verschiedenen Isolationsstufen im Hinblick auf unerwünschte Nebeneffekte zusammen.

Tabelle 4. Zusammenfassung der Isolationsstufen

Isolationsstufe	Zugriff auf nicht festgeschriebene Daten	Nichtwiederholbare Lesevorgänge	Phänomen der Phantomzeilen
Wiederholtes Lesen (RR)	Nicht möglich	Nicht möglich	Nicht möglich
Lesestabilität (RS)	Nicht möglich	Nicht möglich	Möglich
Cursorstabilität (CS)	Nicht möglich	Möglich	Möglich

Tabelle 4. Zusammenfassung der Isolationsstufen (Forts.)

Isolationsstufe	Zugriff auf nicht festgeschriebene Daten	Nichtwiederholbare Lesevorgänge	Phänomen der Phantomzeilen
Nicht festgeschriebener Lesevorgang (UR)	Möglich	Möglich	Möglich

Die folgende Tabelle gibt einfache Anhaltspunkte, die Ihnen bei der Wahl der ersten Isolationsstufe für Ihre Anwendungen helfen können. Betrachten Sie die Angaben dieser Tabelle als Ausgangspunkt, und lesen Sie die Beschreibung der verschiedenen Isolationsstufen in den vorigen Abschnitten, um Hinweise zu erhalten, die vielleicht eine geeignetere Isolationsstufe empfehlen.

Tabelle 5. Richtlinien zur Wahl einer Isolationsstufe

Anwendungsart	Hohe Datenstabilität erforderlich	Hohe Datenstabilität nicht erforderlich
Schreib-/Lese-Transaktionen	RS	CS
Transaktionen mit Lesezugriff	RR oder RS	UR

Die Wahl der geeigneten Isolationsstufe für eine Anwendung ist zur Vermeidung von Erscheinungen, die für die Anwendung nicht tolerierbar sind, äußerst wichtig. Von der Isolationsstufe sind nicht nur die verschiedenen Grade der Isolation zwischen Anwendungen, sondern auch die Leistungsmerkmale einer einzelnen Anwendung betroffen, da die CPU- und Speicherressourcen, die zur Aktivierung und Freigabe von Sperren erforderlich sind, mit der Isolationsstufe variieren. Die Möglichkeit, dass Deadlocks auftreten, ist ebenfalls je nach Isolationsstufe unterschiedlich.

Angeben der Isolationsstufe

Da die Isolationsstufe festlegt, wie Daten gesperrt und von anderen Prozessen während des Zugriffs auf die Daten isoliert werden, sollten Sie eine Isolationsstufe auswählen, die die Anforderungen des gemeinsamen Zugriffs und der Datenintegrität angemessen berücksichtigt. Die Isolationsstufe, die Sie angegeben, gilt für die Dauer der UOW (Unit of Work, Arbeitseinheit).

Anmerkung: Isolationsstufen können für XQuery-Anweisungen nicht auf der Anweisungsebene angegeben werden.

Die Isolationsstufe kann auf verschiedene Arten angegeben werden. Die folgenden heuristischen Methoden können verwendet werden, um die Isolationsstufe festzulegen, die zum Kompilieren einer SQL- oder XQuery-Anweisung verwendet wird:

Statisches SQL:

- Wenn eine Isolationsklausel in der Anweisung angegeben wird, wird der Wert dieser Klausel verwendet.
- Wird in der Anweisung keine Isolationsklausel angegeben, ist die verwendete Isolationsstufe die für das Paket zum Zeitpunkt der Bindung des Pakets an die Datenbank angegebene Klausel.

Dynamisches SQL:

- Wenn eine Isolationsklausel in der Anweisung angegeben wird, wird der Wert dieser Klausel verwendet.
- Wurde in der Anweisung keine Isolationsklausel angegeben, und wurde eine Anweisung SET CURRENT ISOLATION in der aktuellen Sitzung abgesetzt, wird der Wert des Sonderregisters CURRENT ISOLATION verwendet.
- Wird in der Anweisung keine Isolationsklausel angegeben, und wurde keine Anweisung SET CURRENT ISOLATION in der aktuellen Sitzung abgesetzt, ist die verwendete Isolationsstufe die für das Paket zum Zeitpunkt der Bindung des Pakets an die Datenbank angegebene Klausel.

Statische oder dynamische XQuery-Anweisungen:

- Die Isolationsstufe der Umgebung legt die Isolationsstufe fest, wenn der XQuery-Ausdruck ausgewertet wird.

Anmerkung: Viele kommerziell geschriebene Anwendungen ermöglichen eine Auswahl der Isolationsstufe. Informationen dazu finden Sie in der Dokumentation zur jeweiligen Anwendung.

Gehen Sie wie folgt vor, um die Isolationsstufe anzugeben:

- **Beim Vorkompilieren oder Binden:**

  Für eine Anwendung, die in einer unterstützten kompilierten Sprache geschrieben ist, wird die Option ISOLATION der Befehle PREP oder BIND des Befehlszeilenprozessors verwendet. Sie können die Isolationsstufe auch über die APIs PREP und BIND angeben.

  – Wenn Sie beim Vorkompilieren eine Bindedatei erstellen, wird die Isolationsstufe in der Bindedatei gespeichert. Wenn Sie beim Binden keine Isolationsstufe angeben, wird standardmäßig die beim Vorkompilieren verwendete Isolationsstufe verwendet.
  – Wenn Sie keine Isolationsstufe angeben, wird die Standardisolationsstufe Cursorstabilität (CS) verwendet.

  Anmerkung: Führen Sie zur Feststellung der Isolationsstufe eines Pakets die folgende Abfrage aus:

  ```
  SELECT ISOLATION FROM SYSCAT.PACKAGES
    WHERE PKGNAME = 'XXXXXXXX'
    AND PKGSCHEMA = 'YYYYYYYY'
  ```

  Dabei steht `XXXXXXXX` für den Namen des Pakets und `YYYYYYYY` für den Namen des Schemas des Pakets. Beide Namen müssen vollständig in Großbuchstaben angegeben werden.

- **Auf Datenbankservern, die REXX unterstützen:**

  Wenn eine Datenbank erstellt wird, werden mehrere Bindedateien, die die verschiedenen Isolationsstufen für SQL in REXX unterstützen, an die Datenbank gebunden. Andere Befehlszeilenprozessorpakete werden ebenfalls an die Datenbank gebunden, wenn die Datenbank erstellt wird.

  REXX und der Befehlszeilenprozessor stellen die Verbindung zu einer Datenbank mit der Standardisolationsstufe CS (Cursorstabilität) her. Durch die Änderung in eine andere Isolationsstufe wird der Status der Verbindung nicht geändert. Dies muss im Status CONNECTABLE AND UNCONNECTED oder IMPLICITLY CONNECTABLE ausgeführt werden.

Die Isolationsstufe, die von einer REXX-Anwendung verwendet wird, lässt sich durch Prüfen des Werts für die REXX-Variable SQLISL feststellen. Dieser Wert wird jedes Mal aktualisiert, wenn der Befehl CHANGE SQLISL ausgeführt wird.

- **Auf Anweisungsebene:**

  Verwenden Sie die Klausel WITH. Die Isolationsstufe auf Anweisungsebene setzt die Isolationsstufe des Pakets außer Kraft, in dem die Anweisung auftritt.

  Für die folgenden SQL-Anweisungen können Sie eine Isolationsstufe angeben:

  – SELECT
  – SELECT INTO
  – DELETE mit Suche
  – INSERT
  – UPDATE mit Suche
  – DECLARE CURSOR

  Die folgenden Bedingungen gelten für Isolationsstufen, die für Anweisungen angegeben werden:

  – Die WITH-Klausel kann nicht in Unterabfragen verwendet werden.
  – Die Option WITH UR gilt nur für Operationen mit Lesezugriff. Anderenfalls wird die Anweisung automatisch von UR in CS geändert.

- **Über CLI oder ODBC zur Laufzeit:**

  Verwenden Sie den Befehl CHANGE ISOLATION LEVEL. Für DB2 Call Level Interface (DB2 Call Level Interface) können Sie die Isolationsstufe bei der Konfiguration von DB2 Call Level Interface ändern. Verwenden Sie zur Laufzeit die Funktion *SQLSetConnectAttr* mit dem Attribut SQL_ATTR_TXN_ISOLATION, um die Transaktionsisolationsstufe für die aktuelle Verbindung zu definieren, auf die durch *ConnectionHandle* verwiesen wird.Darüber hinaus können Sie das Schlüsselwort TXNISOLATION in der Datei db2cli.ini verwenden.

- **Über JDBC oder SQLJ zur Laufzeit:**

  Anmerkung: JDBC und SQLJ sind in DB2 mit CLI implementiert. Dies bedeutet, dass die Einstellungen der Datei ″db2cli.ini″ Einfluss auf die Projekte haben können, die unter Verwendung von JDBC und SQLJ geschrieben und ausgeführt werden.

  In SQLJ verwenden Sie die SQLJ-Profilanpassungsfunktion (Befehl db2sqljcustomize), um ein Paket zu erstellen. Zu den Optionen, die Sie für dieses Paket angeben können, gehört auch die Isolationsstufe.

- **Für dynamisches SQL in der aktuellen Sitzung:**

  Verwenden Sie die Anweisung SET CURRENT ISOLATION, um die Isolationsstufe für dynamisches SQL festzulegen, die innerhalb einer Sitzung abgesetzt wird. Wird diese Anweisung abgesetzt, wird das Sonderregister CURRENT ISOLATION auf einen Wert festgelegt, der die Isolationsstufe für alle dynamischen SQL-Anweisungen angibt, die innerhalb der aktuellen Sitzung abgesetzt wurden. Sobald dieser festgelegt wurde, stellt das Sonderregister CURRENT ISOLATION die Isolationsstufe für alle nachfolgenden dynamischen SQL-Anweisungen zur Verfügung, die innerhalb der Sitzung kompiliert wurden, unabhängig davon, welches Paket die Anweisung abgesetzt hat. Diese Isolationsstufe wird angewendet bis die Sitzung beendet wird oder bis eine Anweisung SET CURRENT ISOLATION mit der Option RESET abgesetzt wird.

Sperren und Steuerung des gemeinsamen Zugriffs

Zur Steuerung des gemeinsamen Zugriffs sowie zur Verhinderung eines unkontrollierten Datenzugriffs richtet der Datenbankmanager Sperren für Pufferpools, Tabellen, Datenpartitionen, Tabellenblöcke oder Tabellenzeilen ein. Eine *Sperre* ordnet eine Ressource des Datenbankmanagers einer Anwendung zu, die als *Sperreneigner* bezeichnet wird, um die Zugriffsmöglichkeiten anderer Anwendungen auf dieselbe Ressource zu steuern.

Der Datenbankmanager verwendet Sperren entweder auf Zeilen- oder auf Tabellenebene, wobei folgende Faktoren die Grundlage für die Auswahl der jeweils geeigneten Sperre bilden:

- Die Isolationsstufe, die bei der Vorkompilierung oder beim Binden einer Anwendung an die Datenbank angegeben wird. Folgende Isolationsstufen sind möglich:
  - Nicht festgeschriebener Lesevorgang (UR)
  - Cursorstabilität (CS)
  - Lesestabilität (RS)
  - Wiederholtes Lesen (RR)

  Die unterschiedlichen Isolationsstufen werden zur Steuerung des Zugriffs auf nicht festgeschriebene Daten, zur Verhinderung verlorener Aktualisierungen, zur Ermöglichung nicht wiederholter Lesevorgänge für Daten sowie zur Verhinderung von Phantomzeilen verwendet. Verwenden Sie zur Minimierung des Leistungseinflusses die minimale Isolationsstufe, die den Erfordernissen Ihrer jeweiligen Anwendung entspricht.
- Der vom Optimierungsprogramm ausgewählte Zugriffsplan. Tabellensuchen, Indexsuchen und andere Datenzugriffsmethoden erfordern jeweils verschiedene Arten des Zugriffs auf die Daten.
- Das Attribut LOCKSIZE für die Tabelle. Die Klausel LOCKSIZE in der Anweisung ALTER TABLE gibt die Unterteilung (Granularität) der Sperren an, die beim Zugriff auf die Tabellen verwendet werden. Ausgewählt werden können ROW für Zeilensperren, TABLE für Tabellensperren oder BLOCKINSERT für lediglich Blocksperren in MDC-Tabellen. Wenn die Klausel BLOCKINSERT für eine MDC-Tabelle verwendet wird, wird das Sperren auf Zeilenebene durchgeführt, mit Ausnahme einer INSERT-Operation, bei der stattdessen das Sperren auf Blockebene durchgeführt wird. Verwenden Sie für MDC-Tabellen die Anweisung ALTER TABLE ... LOCKSIZE BLOCKINSERT, wenn Transaktionen umfangreiche Einfügeoperationen in zerlegten Zellen durchführen. Verwenden Sie die Anweisung ALTER TABLE ... LOCKSIZE TABLE für Tabellen, auf die lediglich Lesezugriff besteht. Dadurch wird die Zahl der Sperren verringert, die durch die Datenbankaktivität erforderlich werden. Bei partitionierten Tabellen werden erst Tabellensperren und anschließend Datenpartitionssperren angefordert, was die Daten, auf die zugegriffen wird, vorgeben.
- Der Speicherplatz, der für das Sperren aufgewendet wird. Der Speicherplatz, der für das Sperren aufgewendet wird, wird durch den Konfigurationsparameter `locklist` (Sperrenliste) der Datenbank gesteuert. Wenn sich die Sperrenliste füllt, kann sich die Leistung aufgrund von Sperreneskalationen und verringertem gemeinsamen Zugriff auf gemeinsam verwendete Objekte in der Datenbank verschlechtern. Wenn Sperreneskalationen häufig stattfinden, erhöhen Sie den Wert des Parameters `locklist`, des Parameters `maxlocks` oder die Werte beider Parameter. Stellen Sie des Weiteren sicher, dass zur Reduzierung von gleichzeitigen Sperren Transaktionen häufig festgeschrieben werden, damit die Sperren gelöst werden.

Die meisten Sperren werden für Tabellen erstellt. Beim Erstellen, Ändern oder Löschen von Pufferpools wird jedoch eine Pufferpoolsperre gesetzt. Mit dieser Sperre wird der Modus EXCLUSIVE (X) verwendet. Beim Erfassen von Systemüberwachungsdaten ist es möglich, dass Sie auf diesen Sperrentyp treffen. Beim Anzeigen der Momentaufnahme wird ersichtlich, dass der Name der Sperre der Kennung (ID) des Pufferpools selbst entspricht.

Im Allgemeinen werden Sperren auf Zeilenebene verwendet, sofern nicht einer der folgenden Fälle vorliegt:

- Die ausgewählte Isolationsstufe ist nicht festgeschriebenes Lesen (Uncommitted Read, UR).
- Die ausgewählte Isolationsstufe ist wiederholtes Lesen (Repeatable Read, RR) und der Zugriffsplan erfordert eine Suche ohne Vergleichselemente.
- Das Tabellenattribut ist „TABLE".
- Die Sperrenliste füllt sich und verursacht Sperreneskalation.
- Über die Anweisung LOCK TABLE erfolgt eine explizite Sperrung einer Tabelle. Die Anweisung LOCK TABLE verhindert, dass gleichzeitig ablaufende Anwendungsprozesse eine Tabelle ändern oder verwenden.

Wenn es sich hierbei um eine MDC-Tabelle handelt, gibt es verschiedene andere Fälle, in denen das Sperren auf Blockebene anstelle des Sperrens auf Zeilenebene verwendet wird, einschließlich der folgenden Fälle:

- Das Tabellenattribut LOCKSIZE lautet „BLOCKINSERT".
- Die ausgewählte Isolationsstufe ist wiederholtes Lesen (Repeatable Read, RR) und der Zugriffsplan beinhaltet Vergleichselemente.
- Eine durchsuchte Aktualisierungs- oder Löschoperation, die lediglich Vergleichselemente in Dimensionsspalten umfasst.

Sperreneskalationen verringern den gemeinsamen Zugriff. Bedingungen, die Sperreneskalationen verursachen könnten, sollten vermieden werden.

Die Dauer einer Zeilensperre variiert in Abhängigkeit von der verwendeten Isolationsstufe:

- UR-Suchen: Zeilensperren werden nicht aktiviert, sofern keine Zeilendaten geändert werden.
- CS-Suchen: Zeilensperren bleiben nur für die Zeit aktiviert, die sich der Cursor auf der Zeile befindet.
- RS-Suchen: Sperren bleiben nur für die Dauer der Transaktion für Zeilen aktiviert, die den Suchbedingungen entsprechen.
- RR-Suchen: Alle Zeilensperren bleiben für die Dauer der Transaktion aktiviert.

Sperrenattribute

Sperren des Datenbankmanagers verfügen über folgende Grundattribute:

Modus
Die Art des Zugriffs, die dem Sperreneigner gewährt wird, sowie die Art des Zugriffs, die Benutzern gewährt wird, die das gesperrte Objekt gleichzeitig verwenden. Dies wird manchmal auch als *Status* der Sperre bezeichnet.

Objekt
Die Ressource, die gesperrt ist. Der einzige Typ von Objekt, den Sie explizit sperren können, ist eine Tabelle. Der Datenbankmanager implementiert

Sperren auch für andere Arten von Ressourcen, wie Zeilen, Tabellen und Tabellenbereiche. Für MDC-Tabellen (Tabellen mit mehrdimensionalem Clustering) können außerdem Blocksperren aktiviert werden; für partitionierte Tabellen können Datenpartitionssperren aktiviert werden. Das Objekt, das gesperrt wird, bestimmt die *Granularität* der Sperre.

Dauer Der Zeitraum, für den eine Sperre aktiv ist. Die Isolationsstufe, in der die Abfrage ausgeführt wird, beeinflusst die Sperrdauer.

Die folgende Tabelle zeigt die Modi und ihre Auswirkungen in der Reihenfolge zunehmender Ressourcenbeschränkung. Detaillierte Informationen über Sperren auf verschiedenen Stufen finden Sie in den Referenztabellen zu den Sperrmodi.

Tabelle 6. Zusammenfassung der Sperrmodi

Sperrmodus	Gültiger Objekttyp	Beschreibung
IN (Intent None)	Tabellenbereiche, Blöcke, Tabellen, Datenpartitionen	Der Sperreneigner kann alle Daten im Objekt lesen, einschließlich der nicht festgeschriebenen Daten, aber er kann keine Daten aktualisieren. Andere gleichzeitig ausgeführte Anwendungen können die Tabelle lesen oder aktualisieren.
IS (Intent Share)	Tabellenbereiche, Blöcke, Tabellen, Datenpartitionen	Der Sperreneigner kann Daten in der gesperrten Tabelle lesen, aber nicht aktualisieren. Andere Anwendungen können die Tabelle lesen oder aktualisieren.
NS (Next Key Share)	Zeilen	Der Sperreneigner und jede gleichzeitig ausgeführte Anwendung kann die gesperrten Zeilen lesen, aber nicht aktualisieren. Diese Sperre wird anstatt einer Sperre im Modus S für Zeilen einer Tabelle aktiviert, wenn die Isolationsstufe der Anwendung entweder RS oder CS ist. Der Sperrmodus NS wird nicht zum Sperren der nächsten Zeile (Next-Key locking) verwendet. Er wird anstelle des Modus S bei CS- und RS-Suchen verwendet, um die Auswirkungen des Sperrens der nächsten Zeilen bei diesen Suchen zu minimieren.
S (Share)	Zeilen, Blöcke, Tabellen, Datenpartitionen	Der Sperreneigner und jede gleichzeitig ausgeführte Anwendung kann die gesperrten Daten lesen, aber nicht aktualisieren.
IX (Intent Exclusive)	Tabellenbereiche, Blöcke, Tabellen, Datenpartitionen	Der Sperreneigner und gleichzeitig ausgeführte Anwendungen können Daten lesen und aktualisieren. Andere gleichzeitig ausgeführte Anwendungen können die Tabelle sowohl lesen als auch aktualisieren.
SIX (Share with Intent Exclusive)	Tabellen, Blöcke, Datenpartitionen	Der Sperreneigner kann Daten lesen und aktualisieren. Andere gleichzeitig ausgeführte Anwendungen können die Tabelle lesen.
U (Update)	Zeilen, Blöcke, Tabellen, Datenpartitionen	Der Sperreneigner kann Daten aktualisieren. Andere UOWs (UOW = Unit of Work, Arbeitseinheit) können die Daten im gesperrten Objekt lesen, aber nicht versuchen, sie zu aktualisieren.
NW (Next Key Weak Exclusive)	Zeilen	Wenn eine Zeile in einen Index eingefügt wird, wird eine NW-Sperre für die nächste Zeile aktiviert. Für Indizes des Typs 2 findet dies nur statt, wenn die nächste Zeile momentan von einer RR-Suche gesperrt wird. Der Sperreneigner kann die gesperrte Zeile lesen, aber nicht aktualisieren. Dieser Sperrmodus ist dem Sperrmodus X ähnlich, ist jedoch auch mit den Sperrmodi W und NS kompatibel.
X (Exclusive)	Zeilen, Blöcke, Tabellen, Pufferpools, Datenpartitionen	Der Sperreneigner kann Daten im gesperrten Objekt lesen und auch aktualisieren. Nur Anwendungen mit der Isolationsstufe UR (nicht festgeschriebener Lesevorgang) können auf das gesperrte Objekt zugreifen.

Tabelle 6. Zusammenfassung der Sperrmodi (Forts.)

Sperrmodus	Gültiger Objekttyp	Beschreibung
W (Weak Exclusive)	Zeilen	Eine Sperre in diesem Modus wird für die Zeile aktiviert, wenn eine Zeile in eine Tabelle eingefügt wird, für die keine Indizes des Typs 2 definiert sind. Der Sperreneigner kann die gesperrte Zeile ändern. Dieser Sperrmodus wird auch während des Einfügens in einen eindeutigen Index verwendet, um festzustellen, ob ein doppelter Wert festgeschrieben wurde, wenn ein doppelter Wert angetroffen wird. Dieser Sperrmodus ist dem Sperrmodus X ähnlich, jedoch ist er mit dem Sperrmodus NW kompatibel. Nur Anwendungen mit der Isolationsstufe UR (nicht festgeschriebener Lesevorgang) können auf die gesperrte Zeile zugreifen.
Z (Super Exclusive)	Tabellenbereiche, Tabellen, Datenpartitionen	Diese Sperre wird für eine Tabelle unter bestimmten Umständen aktiviert, zum Beispiel, wenn die Tabelle geändert (ALTER) oder gelöscht (DROP) wird, ein Index für die Tabelle erstellt oder gelöscht wird oder bestimmte Arten von Tabellenreorganisation durchgeführt werden. Keine anderen gleichzeitig ausgeführten Anwendungen können die Tabelle lesen oder aktualisieren.

Sperrgranularität

Wenn eine Anwendung eine Sperre für ein Datenbankobjekt aktiviert hat, kann eine andere Anwendung auf dieses Objekt vielleicht nicht zugreifen. Aus diesem Grund sind Sperren auf Zeilenebene in Bezug auf einen maximalen gemeinsamen Zugriff besser als Sperren auf Blockebene, Datenpartitionsebene oder Tabellenebene; dadurch wird das gesperrte Datenvolumen, auf das nicht zugegriffen werden kann, minimiert. Jedoch benötigen Sperren Speicherplatz und Verarbeitungszeit, sodass eine einzelne Tabellensperre den Sperrenaufwand minimiert.

Die Klausel LOCKSIZE der Anweisung ALTER TABLE legt den Geltungsbereich (Granularität) von Sperren auf die Zeilen-, Datenpartitions-, Block- oder Tabellenebene fest. Standardmäßig werden Zeilensperren verwendet. Nur Sperren der Modi S (Shared) und X (Exclusive) werden durch diese definierten Tabellensperren angefordert. Die Klausel LOCKSIZE ROW in der Anweisung ALTER TABLE verhindert normale Sperreneskalationen nicht.

Eine permanente Tabellensperre, die durch eine Anweisung ALTER TABLE definiert wird, kann einer Tabellensperre für eine einzelne Transaktion durch die Anweisung LOCK TABLE in folgenden Fällen vorzuziehen sein:

- Die Tabelle ist schreibgeschützt und benötigt immer nur S-Sperren. Andere Benutzer können ebenfalls S-Sperren für die Tabelle aktivieren.
- Auf die Tabelle greifen in der Regel reine Leseanwendungen zu, jedoch greift manchmal ein einzelner Benutzer zu kurzen Pflegemaßnahmen auf sie zu, und dieser Benutzer benötigt eine X-Sperre. Während das Pflegeprogramm ausgeführt wird, werden die reinen Leseanwendungen ausgesperrt. Unter anderen Umständen können reine Leseanwendung jedoch auf die Tabelle bei minimalem Sperrenaufwand gleichzeitig zugreifen.

Bei einer MDC-Tabelle können Sie BLOCKINSERT für die Klausel LOCKSIZE angeben, um das Sperren auf Blockebene nur bei INSERT-Operationen zu verwenden. Wenn die Angabe stattgefunden hat, wird das Sperren auf Zeilenebene für alle anderen Operationen durchgeführt, jedoch in nur geringfügigem Maße für INSERT-Operationen. D. h., das Sperren auf Blockebene wird beim Einfügen von Zeilen verwendet, das Sperren auf Zeilenebene wird jedoch für das Sperren des

nächsten Schlüssels verwendet, wenn bei der Aktualisierung in den Indizes RR-Suchen aufgefunden werden. Das Sperren mit BLOCKINSERT kann in den folgenden Fällen nützlich sein:

- Es gibt Mehrfachtransaktionen, bei denen umfangreiche Einfügungen in getrennte Zellen stattfinden.
- Normalerweise werden vom System keine Einfügeoperationen ausgeführt, bei denen mehrere Transaktionen gleichzeitig Daten in eine einzelne Zelle einfügen. Sind derartige Einfügeoperationen doch erforderlich, so werden pro Zelle von jeder der beteiligten Transaktionen so viele Daten eingefügt, dass der Benutzer nicht überprüfen muss, ob die Daten durch die einzelnen Transaktionen in separate Blöcke eingefügt werden.

Die Anweisung ALTER TABLE gibt Sperren global an, wodurch alle Anwendungen und Benutzer, die auf die Tabelle zugreifen, betroffen werden. Einzelne Anwendungen können die Anweisung LOCK TABLE verwenden, um stattdessen Tabellensperren auf Anwendungsebene zu definieren.

Wartestatus und Zeitlimitüberschreitungen für Sperren

Die Erkennung von Zeitlimitüberschreitungen für Sperren ist eine Datenbankmanagerfunktion, die verhindert, dass Anwendungen unendlich lange auf die Freigabe einer Sperre in einer abnormalen Situation warten. Zum Beispiel könnte eine Transaktion auf eine Sperre warten, die von der Anwendung eines anderen Benutzers aktiviert wurde. Der andere Benutzer hat jedoch seine Workstation verlassen, ohne seine Anwendung die Transaktion festschreiben zu lassen, wodurch die Sperre freigegeben würde. Um die Blockierung einer Anwendung in einem solchen Fall zu vermeiden, setzen Sie den Konfigurationsparameter **locktimeout** auf die maximale Zeitdauer, die eine beliebige Anwendung auf eine Sperre warten sollte.

Durch die Einstellung dieses Parameters können globale Deadlocks besser vermieden werden, insbesondere in Anwendungen mit verteilten Arbeitseinheiten (DUOW). Wenn die Zeit, die eine Sperrenanforderung ansteht, länger ist als die im Wert des Parameters **locktimeout** definierte Zeit, empfängt die anfordernde Anwendung einen Fehler und ihre Transaktion wird mit ROLLBACK rückgängig gemacht. Beispiel: Wenn Anwendung 1 versucht, eine Sperre zu erhalten, die bereits für Anwendung 2 aktiv ist, gibt Anwendung 1 den SQLCODE-Wert -911 (SQLSTATE-Wert 40001) mit dem Ursachencode 68 zurück, wenn das Zeitlimit abläuft. Der Standardwert für den Parameter **locktimeout** ist -1, d. h. die Erkennung der Überschreitung des Sperrzeitlimits ist ausgeschaltet.

Anmerkung: Für Tabellen-, Zeilen-, Datenpartitions- und MDC-Blocksperren kann eine Anwendung die Einstellung für **locktimeout** auf der Datenbankebene mit der Anweisung SET CURRENT LOCK TIMEOUT überschreiben.

Wenn eine Berichtsdatei über Sperrzeitlimitüberschreitungen generiert werden soll, setzen Sie die Registrierdatenbankvariable **DB2_CAPTURE_LOCKTIMEOUT** auf den Wert ON. Der Bericht zu einer Zeitlimitüberschreitung enthält Informationen zu den Hauptanwendungen, die an dem Sperrenkonflikt beteiligt sind, der zu der Sperrzeitlimitüberschreitung führte. Darüber hinaus enthält er Details über die Sperre selbst, wie zum Beispiel den Namen und den Typ der Sperre, die Zeilen-ID, die Tabellenbereichs-ID und die Tabellen-ID.

Wenn Sie mehr Informationen zu Zeitlimitüberschreitungen bei Sperrenanforderungen in der Datei db2diag.log protokollieren wollen, setzen Sie den Konfigurationsparameter **diaglevel** des Datenbankmanagers auf den Wert 4. Die protokollierten Informationen geben das gesperrte Objekt, den Sperrmodus und

die Anwendung an, die die Sperre aktiviert hat. Der Name der aktuellen dynamischen SQL- oder XQuery-Anweisung oder des statischen Pakets kann ebenfalls protokolliert werden. Eine dynamische SQL- oder XQuery-Anweisung wird nur protokolliert, wenn der Parameter **diaglevel** auf den Wert 4 gesetzt ist.

Informationen über Wartestatus für Sperren und Zeitlimitüberschreitungen durch Sperren können Sie mithilfe der Systemmonitorelemente zum Wartestatus von Sperren ('lock_wait') oder über den Diagnoseanzeiger **db.apps_waiting_locks** abrufen.

Berichterstellung für Überschreitungen des Zeitlimits für Sperren

Die Funktion zur Berichterstellung für Überschreitungen des Zeitlimits für Sperren erfasst Informationen zu Ereignissen von Zeitlimitüberschreitungen bei Sperren. Diese enthalten Informationen zu den wichtigsten Anwendungen, die an dem Sperrenkonflikt beteiligt sind, der zur Überschreitung des Sperrzeitlimits geführt hat. Diese Funktion wird durch die Registrierdatenbankvariable **DB2_CAPTURE_LOCKTIMEOUT** gesteuert.

Informationen werden über den Anforderer der Sperre (die Anwendung, die den Zeitlimitfehler empfangen hat) und den aktuellen Sperreneigner erfasst. Der Bericht wird in einem Verzeichnis gespeichert, das durch einen Datenbankkonfigurationsparameter bestimmt wird.

Wenn eine Überschreitung des Sperrzeitlimits auftritt und die Berichtsfunktion für Sperrzeitlimitüberschreitungen aktiv ist, werden die folgenden Informationen erfasst und in eine Berichtsdatei für Sperrzeitlimitüberschreitung geschrieben.

Sperre des Zugriffskonflikts

- Name und Typ der Sperre.
- Sperrenspezifikationen wie Zeilen-ID, Tabellenbereichs-ID und Tabellen-ID. Anhand dieser Informationen können Sie die Systemkatalogsicht SYSCAT.TABLES abfragen, um den Namen der Tabelle zu ermitteln.

Anforderer der Sperre

- Informationen zur Identifikation der Anwendung, wie zum Beispiel der Name und die Koordinatorpartition der Anwendung
- Wert des Zeitlimits
- Angeforderter Sperrmodus
- SQL-Kontext der Anforderung, falls zutreffend
- Paket-ID
- Abschnittseintragsnummer
- SQL-Informationen, zum Beispiel dynamische oder statische Anweisung, Typ der Anweisung
- Effektive Isolationsstufe der Sperre
- Relevanter Anweisungstext, falls verfügbar (siehe „Funktionseinschränkungen" auf Seite 192)
- Anwendungsstatus
- Aktuelle Operation
- Sperreneskalation

Eigner oder Vertreter der Sperre

Es kann mehr als einen Sperreneigner geben. Wenn eine Sperre zum Beispiel in gemeinsam genutzten Modus (SHARE) von mehreren

Anwendungen aktiviert wurde, werden nur die Informationen des ersten ermittelten Sperreneigners berichtet. Der erste Sperreneigner ist Vertreter für weitere Sperreneigner.

- Informationen zur Identifikation der Anwendung, wie zum Beispiel der Name und die Koordinatorpartition der Anwendung
- Aktivierter Sperrmodus
- Liste der momentan aktiven SQL-Anweisungen in dieser Partition
  1. Paket-ID
  2. Abschnittseintragsnummer
  3. SQL-Informationen, zum Beispiel dynamische oder statische Anweisung, Typ der Anweisung
  4. Effektive Isolationsstufe der Sperre
  5. Relevanter Anweisungstext, wenn verfügbar
- Die Liste inaktiver SQL-Anweisungen aus der aktuellen UOW (Unit of Work, Arbeitseinheit) in dieser Datenbankpartition (nur verfügbar, wenn ein Deadlock-Ereignismonitor mit Anweisungsverlaufsprotokollierung aktiv ist)
  1. Paket-ID
  2. Abschnittseintragsnummer
  3. SQL-Informationen, zum Beispiel dynamische oder statische Anweisung, Typ der Anweisung
  4. Effektive Isolationsstufe der Sperre
  5. Relevanter Anweisungstext, wenn verfügbar

Zur Erfassung zusätzlicher Informationen, wie zum Beispiel Informationen zum Betriebssystem oder andere zugehörige Umgebungsinformationen, verwenden Sie das angepasste Script db2cos.

Verwendung der Funktion

Die Registrierdatenbankvariable **DB2_CAPTURE_LOCKTIMEOUT** steuert die Berichtsfunktion für Sperrzeitlimitüberschreitungen.

Wenn diese Registrierdatenbankvariable auf den Wert ON gesetzt wird, erfasst die Funktion grundlegende Informationen zu jeder Überschreitung eines Sperrzeitlimits, die innerhalb der DB2-Instanz auftritt. Außerdem wird ein Bericht zu dieser Sperrzeitlimitüberschreitung erstellt. Wenn die Registrierdatenbankvariable auf den leeren Wert gesetzt wird, wird die Funktion inaktiviert.

Zur Aktivierung der Berichterstellung für Sperrzeitlimitüberschreitungen führen Sie den folgenden Befehl aus:

```
db2set DB2_CAPTURE_LOCKTIMEOUT=ON
```

Zur Inaktivierung der Berichterstellung für Sperrzeitlimitüberschreitungen führen Sie den folgenden Befehl aus:

```
db2set DB2_CAPTURE_LOCKTIMEOUT=
```

In einigen Fällen kann es sinnvoll sein, alle Anweisungen zu erfassen, die in der UOW des Sperreneigners ausgeführt wurden, bevor die Sperrzeitlimitüberschreitung aufgetreten ist. Dies ist zum Beispiel nützlich, wenn die Sperrzeitlimitüberschreitung durch eine zuvor ausgeführte SQL-Anweisung verursacht wurde. Wenn der vorhergehende Verlauf einer UOW in die Informationen zu einer

Sperrzeitlimitüberschreitung für den aktuellen Sperreneigner aufgenommen werden soll, aktivieren Sie einen Deadlock-Ereignismonitor mithilfe der Anweisungsklausel HISTORY. Verwenden Sie zum Beispiel eine der folgenden Anweisungen:

```
create event monitor testit for deadlocks with details history write to file pfad global
create event monitor testit for deadlocks with details history write to table
```

Die Anweisung CREATE EVENT MONITOR besitzt noch weitere Optionen, wie zum Beispiel die Möglichkeit, den Namen des Tabellenbereichs und der Tabelle anzugeben, in die Daten geschrieben werden sollen. Details finden Sie in der Beschreibung der Anweisung CREATE EVENT MONITOR.

Der Ereignismonitor mit der Funktion zum Überwachen des Anweisungsverlaufs bezieht sich auf alle Anwendungen und erhöht die Monitorspeicherbelegung durch den DB2-Datenbankmanager. Verwenden Sie ihn nur in Fällen, in denen die Ursache des Sperrenkonflikts nicht die aktuelle Anweisung ist, die von den beteiligten Anwendungen ausgeführt wird, oder wenn der Zweck des Berichts mehr als eine einfache Ermittlung der an der Sperrzeitlimitüberschreitung beteiligten Anwendungen ist.

Betrachten Sie ein Beispiel: Die Anwendung 1 führt die folgenden Anweisungen aus:

```
INSERT INTO T1 VALUES (1)
SELECT * FROM T5
```

Nachfolgend führt die Anwendung 2 die folgende Anweisung aus:

```
SELECT * FROM T1
```

In diesem Beispiel wartet die Anweisung SELECT der Anwendung 2 auf die Zeilensperre, die von der zuvor von Anwendung 1 ausgeführten Anweisung INSERT aktiviert wurde. In diesem Beispiel würde der Bericht zur Anweisung SELECT, die von Anwendung 1 ausgeführt wird, nicht helfen, die genaue Ursache des Sperrenkonflikts zu erkennen. Der Anweisungsverlauf für den Sperreneigner würde jedoch die zuvor ausgeführte Anweisung INSERT und die Ursache für den Sperrenkonflikt offenbaren.

Die Erstellung eines Ereignismonitors mit der Funktion zum Überwachen des Anweisungsverlaufs würde auch helfen, wenn die Anwendung, die die Sperre aktiviert hat, zum Zeitpunkt der Sperrzeitlimitüberschreitung keine SQL-Anweisung ausführt. Wenn ein Deadlock-Ereignismonitor mit Verlaufsüberwachungsfunktion aktiv ist, werden die vorherigen SQL-Anweisungen der Anwendung in der UOW in den Bericht geschrieben.

Funktionseinschränkungen

Der Bericht zu Sperrzeitlimitüberschreitungen erfasst möglicherweise nicht immer die erforderlichen Details zu Ereignissen von Sperrenkonflikten und Sperrzeitlimitüberschreitungen. In einigen Fällen, zum Beispiel in den nachfolgend aufgeführten, stehen der Berichtsfunktion für Sperrzeitlimitüberschreitungen nicht alle Informationen zur Verfügung:

- Der Text der SQL-Anweisung ist möglicherweise nicht in allen Fällen verfügbar, zum Beispiel wenn statische SQL-Anweisungen an der Sperrzeitlimitüberschreitung beteiligt sind.
- DB2-Dienstprogramme und interne Funktionen können Sperren aktivieren, ohne SQL-Anweisungen auszuführen. Zum Beispiel fordert das Dienstprogramm für Online-Backup eine Tabellenbereichssperre und eine Tabellensperre an, wenn

diese Objekte während des Online-Backups verarbeitet werden. Eine Anwendung könnte das Zeitlimit beim Warten auf eine Sperre überschreiten, während eine Tabelle durch ein Backup gesichert wird. Wenn der Befehl BACKUP über den Befehlszeilenprozessor (CLP) aufgeführt wurde, würde der Anwendungsname als "db2bp" (CLP) gemeldet, und es gäbe keine aktive SQL-Anweisung für den Sperreneigner. Ein DB2-Dienstprogramm oder eine interne Funktion kann ebenfalls das Zeitlimit überschreiten, wenn es bzw. sie darauf wartet, dass eine Anwendung eine Sperre freigibt. In diesem Fall gäbe es keine aktive SQL-Anweisung für den Sperrenanforderer.

- Subagenten, die für eine Anwendung in fernen Partitionen aktiv sind, haben möglicherweise unvollständige Informationen zu den SQL-Anwendungen und anderen Informationen der Anwendung. Insbesondere verfügt der Sperreneigner, wenn er ein ferner Subagent ist, über keinen vollständigen Anweisungsverlauf für die Anwendung.
- Die interne Katalogtabellenverarbeitung, die während des regulären DB2-Betriebs ausgeführt wird, verwendet kein SQL. Die Funktion wird lediglich die beteiligte Tabelle, jedoch nicht die Komponente von DB2 angeben, die entweder den Konflikt verursacht hat oder von ihm betroffen ist.
- Einige Sperren werden von bestimmten SQL-Anweisungen nicht aktiviert. Sie können andere Objekte als Tabellen (z. B. Pakete) darstellen oder sie werden für andere Zwecke der internen DB2-Verarbeitung verwendet.

Berichtsdateien für Überschreitungen des Zeitlimits für Sperren

Wenn die Funktion zur Berichterstellung bei Überschreitungen des Sperrzeitlimits aktiv ist und eine Überschreitung auftritt, generiert der Agent, der den Sperrzeitlimitfehler empfängt, eine Berichtsdatei. Diese Datei wird in dem Verzeichnispfad für Diagnosedaten in der Datenbankpartition abgelegt, in der die Zeitlimitüberschreitung für Sperren aufgetreten ist.

Der Verzeichnispfad für die Diagnosedaten kann mithilfe des Konfigurationsparameters des Datenbankmanagers **diagpath** definiert werden. Wird dieser Parameter nicht definiert, kann die Speicherposition der Datei variieren. Weitere Informationen zur Speicherposition der Berichtsdatei finden Sie in der Beschreibung zum Parameter **diagpath**.

In einer Umgebung mit partitionierten Datenbanken kann eine Anwendung gleichzeitig Verarbeitungsoperationen für eine oder auch mehrere Datenbankpartitionen ausführen und auch Sperren für diese anfordern. Die Berichtsfunktion für Zeitlimitüberschreitungen für Sperren erleichtert die Isolation der Datenbankpartition, in der die Fehler wegen einer Zeitlimitüberschreitung für Sperren aufgetreten sind, indem ein Bericht für diese Datenbankpartition generiert wird. Der Bericht umfasst Informationen zum eindeutigen Kontext des Fehlers wegen einer Zeitlimitüberschreitung für Sperren in dieser Partition.

Der Bericht wird in einer Datei mit dem folgenden Namensformat gespeichert: db2locktimeout.*par*.*AGENTID*.*jjjj-mm-tt-hh-mm-ss*. Dabei gilt Folgendes:

- *par* ist die Datenbankpartitionsnummer. In nicht partitionierten Datenbankumgebungen ist *par* auf 0 gesetzt.
- *AGENTID* ist die Agenten-ID.
- *jjjj-mm-tt-hh-mm-ss* ist die aus der Angabe für Jahr, Monat, Tag, Stunde, Minute und Sekunde bestehende Zeitmarke.

Ein Beispiel für den Dateinamen eines Zeitlimitüberschreitungsberichts ist /home/juntang/sqllib/db2dump/db2locktimeout.000.4944050.2006-08-11-11-09-43.

Anmerkung: Wenn keine Berichtsdateien für Zeitlimitüberschreitungen für Sperren mehr benötigt werden, sollten Sie diese aus dem Verzeichnis löschen. Da sich die Berichtsdateien an derselben Speicherposition wie andere Diagnoseprotokolle befinden, wird das DB2-System möglicherweise beendet, wenn das Verzeichnis vollständig belegt ist. Wenn Sie bestimmte Berichtsdateien für Zeitlimitüberschreitungen für Sperren über einen längeren Zeitraum aufbewahren müssen, sollten Sie diese in ein anderes Verzeichnis versetzen.

Beschreibung des Berichts für Zeitlimitüberschreitungen für Sperren

Die Arbeitsschritte in diesem Abschnitt dienen zum Generieren eines typischen Berichts für Zeitlimitüberschreitungen für Sperren. Im vorliegenden Beispiel ist die Datenbank MYDB2 bereits vorhanden.

1. Stellen Sie eine Verbindung zur Datenbank MYDB2 her:
   ```
   db2 connect to mydb2
   ```
2. Erstellen Sie eine Tabelle mit dem Namen T1:
   ```
   db2 create table T1 ( c1 int )
   ```
3. Fügen Sie wie folgt Werte in die Tabelle ein:
   ```
   db2 insert into T1 values 0,1,2,3,4,5,6,7,8,9
   ```
4. Erstellen Sie einen Ereignismonitor mit dem Namen TESTIT:
   ```
   db2 create event monitor testit for deadlocks with details history write to table autostart
   ```
5. Aktivieren Sie den Ereignismonitor:
   ```
   db2 set event monitor testit state 1
   ```
6. Schreiben Sie die durchgeführten Änderungen fest:
   ```
   db2 commit
   ```
7. Aktivieren Sie die Berichtsfunktion für Zeitlimitüberschreitungen für Sperren:
   ```
   db2set DB2_CAPTURE_LOCKTIMEOUT=ON
   ```

Führen Sie die folgenden Arbeitsschritte aus, um eine Zeitlimitüberschreitung für Sperren zu erzeugen, wenn zwei (in separaten CLP-Fenstern ausgeführte) Anwendungen auf die Datenbank MYDB2 zugreifen. Das erste CLP-Fenster wird verwendet, um die Anwendung Appl1 auszuführen, das zweite CLP-Fenster zur Ausführung der Anwendung Appl2.

1. Im ersten CLP-Fenster (in dem Appl1 ausgeführt wird) müssen Sie die folgenden Befehle eingeben:
   ```
   db2 connect to mydb2
   db2 commit
   ```
2. Im zweiten CLP-Fenster (in dem Appl2 ausgeführt wird) müssen Sie die folgenden Befehle eingeben:
   ```
   db2 connect to mydb2
   db2 set lock timeout 1
   ```

   Anmerkung: Die Intervalleinstellung von 1 Sekunde für die Zeitlimitüberschreitung für Sperren wird für Produktionsumgebungen nicht empfohlen.
3. Geben Sie im ersten CLP-Fenster die folgenden Befehle ein:
   ```
   db2 +c declare c1 cursor for select tabname from syscat.tables
   db2 +c open c1
   db2 +c fetch c1
   db2 +c insert into T1 values 8
   ```

   Anmerkung: Mit der Option **+c** wird die Funktion für das automatische Commit inaktiviert.

4. Geben Sie im zweiten CLP-Fenster den folgenden Befehl ein:
   ```
   db2 "select * from T1"
   ```

Appl1 hält eine exklusive Satzsperre für Tabelle T1. Eine Zeitlimitüberschreitung für Sperren tritt auf, wenn Appl2 versucht, eine sequenzielle Suche über die gesamte Tabelle T1 durchzuführen. Von einen Agenten von Appl2 wird in der Datei *db2locktimeout.000.1376548.2007-01-07-15-22-13* der folgende Bericht für eine Zeitlimitüberschreitung für Sperren erstellt.

Anmerkung: Die in dem folgenden Beispiel aufgeführten Zeilennummern sind nicht Bestandteil des Berichts für die Zeitlimitüberschreitung für Sperren. Das Format der in Ihren Umgebungen erstellten Berichte kann in unterschiedlichen Releases von DB2 möglicherweise leichte Unterschiede aufweisen, und es werden eventuell neue Felder hinzugefügt, wenn die Berichtsfunktion erweitert wird.

Tabelle 7. Inhalt des Berichts für Zeitlimitüberschreitungen für Sperren

Abschnitt des Berichts für Zeitlimitüberschreitungen für Sperren	**Beschreibung**

`1.LOCK TIMEOUT REPORT`
 	
`2.Date:  07/01/2007`
`3.Time:  15:22:13`
`4.Instance: juntang`
`5.Database: MYDB2`
`6.Database Partition: 000`
	Die Zeilen 2 bis 6 geben die Uhrzeit und die Position der Zeitlimitüberschreitung für Sperren an:
	• Das lokale Datum der Datenbankpartition: 07. Januar 2007.
	• Die lokale Uhrzeit der Datenbankpartition: 15:22:13.
	• Der Name der Datenbankinstanz: juntang.
	• Der Datenbankname: MYDB2.
	• Die Partitionsnummer: 0.

`7.Lock Information:`
 	
`8.Lock Name:  0002000A000000000300000B52`
`9.Lock Type:  Row`
`10.Lock Specifics: Table space ID=2, Table ID=10,`
`Row ID=x000000000300000B`
	Die Zeilen 7 bis 10 enthalten die Sperreninformationen für den Zeitpunkt, zu dem die Zeitlimitüberschreitung aufgetreten ist:
	• Der Name der Sperre: 0002000A000000000300000B52.
	• Der Sperrentyp: Zeile. Dieser Typ gibt an, dass die Sperre sich auf eine bestimmte Zeile bezieht.
	• Die Kennungen für den Tabellenbereich, die Tabelle und die Zeile, für die die Sperre gehalten wird: 2, 10 bzw. x000000000300000B. Anhand dieser Informationen können Sie den Namen der Tabelle feststellen, für die die Sperre gesetzt wurde. Fragen Sie hierzu die Systemkatalogsicht SYSCAT.TABLES ab. Geben Sie beispielsweise `SELECT TABNAME, TABSCHEMA, TBSPACE FROM SYSCAT.TABLES WHERE TABLEID = 10 AND TBSPACEID = 2` ein, um den Tabellennamen, das Schema und den Namen des Tabellenbereichs zurückzugeben.

Tabelle 7. Inhalt des Berichts für Zeitlimitüberschreitungen für Sperren (Forts.)

Abschnitt des Berichts für Zeitlimitüberschreitungen für Sperren	Beschreibung
11.Lock Requestor: 12.System Auth ID: JUNTANG 13. Application Handle:[0-32] 14. Application ID: *LOCAL.juntang.070107170213 15. Application Name: db2bp 16. Requesting Agent ID: 1376548 17. Coordinator Agent ID: 1376548 18. Coordinator Partition: 0 19. Lock timeout Value: 1000 milliseconds 20. Lock mode requested: .NS 21. Application status: (SQLM_UOWEXEC) 22. Current Operation: (SQLM_FETCH) 23. Lock Escalation: No	Die Zeilen 11 bis 23 enthalten detaillierte Informationen zu der Anwendung, die auf die Zuteilung der Sperre wartet: • Die Systemauthentifizierungs-ID: JUNTANG. • Die Anwendungskennung: [0-32]. • Die Anwendungs-ID: *LOCAL.juntang.070107170213. Informationen zur Interpretation der Anwendungs-ID finden Sie in „appl_id - Anwendungs-ID (Monitorelement)" in *Systemmonitor Handbuch und Referenz*. • Der Name der Anwendung, die die Zeitlimitüberschreitung für Sperren verursacht hat: db2bp - der Name des CLP-Back-End-Prozesses. Dieser Wert kann in bestimmten Fällen den Namen einer Anwendung angeben. • Die ID des Agenten, der die Anwendung bedient: 1376548. • Die ID des Koordinatoragenten: 1376548. • Die Nummer der Koordinatorpartition: 0. • Der Wert der Zeitlimitüberschreitung für Sperren für den Sperrenanforderer: 1000 Millisekunden (weil in dem CLP-Fenster, in dem Appl2 ausgeführt wird, db2 set lock timeout 1 eingegeben wurde). • Der angeforderte Sperrmodus: NS. • Der Anwendungsstatus: SQLM_UOWEXEC. Dies bedeutet, dass eine UOW (Unit of Work) ausgeführt wird. Eine Liste der gültigen Anwendungsstatus und der zugehörigen Definitionen finden Sie in „appl_status - Anwendungsstatus (Monitorelement)" in *Systemmonitor Handbuch und Referenz*. • Die Anweisungsoperation, die momentan ausgeführt wird oder gerade abgeschlossen wurde: SQLM_FETCH. Eine Liste der gültigen Operationen finden Sie in „stmt_operation/operation - Anweisungsoperation (Monitorelement)" in *Systemmonitor Handbuch und Referenz*. • Sperreneskalation: No (Nein). Dies bedeutet, dass die vom Sperrenanforderer verwendete Sperre nicht eskaliert wurde.

Tabelle 7. Inhalt des Berichts für Zeitlimitüberschreitungen für Sperren (Forts.)

Abschnitt des Berichts für Zeitlimitüberschreitungen für Sperren	**Beschreibung**

24.Context of Lock Request:
25. Identification: UOW ID (28); Activity ID (1)
26. Activity Information:
27. Package Schema: (NULLID)
28. Package Name: (SQLC2F0A)
29. Package Version: ()
30. Section Entry Number: 201
31. SQL Type: Dynamic
32. Statement Type: DML, Select (blockable)
33. Effective Isolation: Cursor Stability
34. Statement Unicode Flag: No
35. Statement: select * from t1
	Die Zeilen 24 bis 35 enthalten detaillierte Informationen zu der SQL-Anweisung, die vom Sperrenanforderer ausgeführt wurde:
	• Die ID der aktuellen UOW des Sperrenanforderers (28) und die Aktivitäts-ID in der UOW (1).
	• Das Paketschema: NULLID.
	• Der Paketname: SQLC2F0A.
	• Die Abschnittsnummer der SQL-Anweisung, die vom Sperrenanforderer ausgeführt wird: 201.
	• Der SQL-Typ: dynamic. Der andere gültige Wert lautet 'static'.
	• Der Typ der Anweisungsoperation: DML, Select (blockable). Die anderen gültigen Werte umfassen DDL-Operationen.
	• Der für den Sperrenanforderer gesetzte Isolationsmodus: Cursor Stability (CS).
	• Die SQL-Anweisung: SELECT * FROM T1.
	Anmerkung: Bei statischen SQL-Anweisungen ist das Feld für die Anweisung möglicherweise leer. Um den Text der statischen SQL-Anweisung festzustellen, können Sie die Werte in den Feldern für den Paketnamen, das Paketschema, die Paketversion und die Abschnittseintragsnummer verwenden, um die Systemkatalogsicht SYSCAT.STATEMENTS abzufragen. Dieser Arbeitsschritt wird im folgenden Beispiel dargestellt:
	`SELECT SEQNO, SUBSTR(TEXT,1,120)`
	`FROM SYSCAT.STATEMENTS`
	`WHERE PKGNAME = '`*`paketnamenwert`*`'`
	`AND PKGSCHEMA = '`*`paketschemawert`*`'`
	`AND VERSION = '`*`paketversionswert`*`'`
	`AND SECTNO =` *`abschnittseintragsnummernwert`*
	Da es bei dieser Abfrage zu Konkurrenzsituationen mit anderen Anwendungen kommen kann, die versuchen, die Katalogsicht SYSCAT.STATEMENTS zu aktualisieren, sollten Sie die Isolationsstufe UR (Uncommitted Read) verwenden, wenn Sie diese Abfrage ausführen. Wenn die Isolationsstufe nicht geändert werden kann, sollten Sie diese Abfrage nur ausführen, wenn in der Datenbank nur wenig Aktivität besteht. Die Isolationsstufe kann gesetzt werden, indem Sie den Befehl SET CURRENT ISOLATION=UR im aktuellen CLP-Fenster ausführen oder indem Sie die Klausel WITH UR zur SQL-Anweisung hinzufügen.

36.Lock Owner (Representative): 37. System Auth ID: JUNTANG 38. Application Handle: [0-27] 39. Application ID: *LOCAL.juntang.070107170027 40. Application Name: db2bp 41. Requesting Agent ID: 1589358 42. Coordinator ID: 1589358 43. Coordinator Partition: 0 44. Lock mode held: ..X	Die Zeilen 36 bis 44 enthalten detaillierte Informationen zum Sperreneigentümer, bei dem es sich um die Anwendung handelt, die die Sperre hält. Die Informationen stimmen mit den Informationen zum Sperrenanforderer überein, die in den Zeilen 11 bis 23 aufgeführt sind.

Tabelle 7. Inhalt des Berichts für Zeitlimitüberschreitungen für Sperren (Forts.)

Abschnitt des Berichts für Zeitlimitüberschreitungen für Sperren	Beschreibung
45.List of Active SQL Statements: 46. Entry #1 47. Identification: UOW ID (7); Activity ID (1) 48. Package Schema: (NULLID) 49. Package Name: (SQLC2F0A) 50. Package Version: () 51. Section Entry Number: 1 52. SQL Type: Dynamic 53. Statement Type: DML, Select (blockable) 54. Effective Isolation: Cursor Stability 55. Statement Unicode Flag: No 56. Statement: select tabname from syscat.tables 57. List of Inactive SQL Statements from current UOW: 58. Entry: #1 59. Identification: UOW ID (7); Activity ID (2) 60. Package Schema: (NULLID) 61. Package Name: (SQLC2F0A) 62. Package Version: () 63. Section Entry Number: 203 64. SQL Type: Dynamic 65. Statement Type: DML, Insert/Update/Delete 66. Effective Isolation: Cursor Stability 67. Statement Unicode Flag: No 68. Statement: insert into t1 values 8	Die Zeilen 45 bis 68 enthalten das SQL-Anweisungsprotokoll des Sperreneigentümers. Die Informationen ähneln den Aktivitätsinformationen für den Sperrenanforderer in den Zeilen 24 bis 35: • Die Zeilen 46 bis 56 enthalten Informationen zur aktuellen SQL-Anweisung des Sperreneigentümers. Die Anweisung für den Sperreninhaber sind möglicherweise nicht verfügbar. Dieser Fall tritt ein, wenn die vom Sperreninhaber zuletzt verarbeitete Anweisung abgeschlossen wurde. • Die Zeilen 57 bis 68 enthalten Informationen zu den inaktiven SQL-Anweisungen, die zuvor in der UOW des Sperreneigentümers ausgeführt wurden und noch nicht festgeschrieben wurden. Im vorliegenden Beispiel gibt es nur eine inaktive SQL-Anweisung. In anderen Umgebungen können auch mehrere inaktive SQL-Anweisungen vorhanden sein.

Beispielberichte für Zeitlimitüberschreitungen für Sperren

In diesem Abschnitt werden Berichte für Zeitlimitüberschreitungen für Sperren dargestellt, die in unterschiedlichen Szenarios generiert werden:

- Dynamisches SQL verwenden (Sperreneigentümer führt keine Verarbeitungsoperationen aus)
- Dynamisches SQL verwenden (Sperreneigentümer führt bestimmte Verarbeitungsoperationen aus)
- Statisches SQL verwenden

Szenario 1: Dynamisches SQL verwenden (Sperreneigentümer führt keine Verarbeitungsoperationen aus)

Dieses Beispiel zeigt den Inhalt eines Berichts für Zeitlimitüberschreitungen für Sperren, wobei kein Anweisungsprotokoll erfasst wird und der Sperreneigentümer momentan keine Verarbeitungsoperationen ausführt. Das Anweisungsprotokoll für einen Sperreninhaber wird nicht erfasst, wenn kein Deadlock-Ereignismonitor mit einer Anweisungsprotokollklausel ausgeführt wird oder wenn der Sperrenkoordinator sich in einer anderen Partition befindet. Im vorliegenden Beispiel führt der Sperreninhaber momentan keine SQL-Anweisung aus. Aus diesem Grund ist keine aktive Anweisung vorhanden, die für den Sperreninhaber gemeldet werden könnte.

```
LOCK TIMEOUT REPORT

Date:     06/11/2006
Time:     18:36:43
Instance: juntang
Database: MYDB2
Database Partition: 000
```

```
Lock Information:

   Lock Name:      0002000200000070000000052
   Lock Type:      Row
   Lock Specifics: Table space ID=2, Table ID=2, Row ID=00000007


Lock Requestor:
   System Auth ID:          JUNTANG
   Application Handle:      [1-53]
   Application ID:          *N1.juntang.061106233631
   Application Name:        db2bp
   Requesting Agent ID:     1024114
   Coordinator Agent ID:    0
   Coordinator Partition:   1
   Lock timeout Value:      1000 milliseconds
   Lock mode requested:     .NS
   Application status:      (SQLM_UOWEXEC)
   Current Operation:       (SQLM_FETCH)
   Lock Escalation:          No

   Context of Lock Request:
   Identification:            UOW ID (2); Activity ID (1)
   Activity Information:
      Package Schema:         (NULLID  )
      Package Name:           (SQLC2E07)
      Package Version:        ()
      Section Entry Number:  201
      SQL Type:                Dynamic
      Statement Type:         DML, Select (blockable)
      Effective Isolation:   Cursor Stability
      Statement Unicode Flag: No
      Statement:              select * from T1


Lock Owner (Representative):
   System Auth ID:          JUNTANG
   Application Handle:      [0-53]
   Application ID:          *N0.juntang.061106233613
   Application Name:        db2bp
   Requesting Agent ID:    1429550
   Coordinator Agent ID:   1429550
   Coordinator Partition:  0
   Lock mode held:          ..X

   List of Active SQL Statements:  Not available

   List of Inactive SQL Statements from current UOW : Not available
```

Szenario 2: Dynamisches SQL verwenden (Sperreneigentümer führt bestimmte Verarbeitungsoperationen aus)

Dieses Beispiel zeigt den Inhalt eines Berichts für Zeitlimitüberschreitungen für Sperren, wobei ein Anweisungsprotokoll erfasst wird und der Sperreneigentümer momentan bestimmte Verarbeitungsoperationen ausführt. Das Anweisungsprotokoll für den Sperreninhaber wurde erfasst, weil in der Partition, mit der die Anwendung verbunden war, ein Deadlock-Ereignismonitor mit der Protokollklausel ausgeführt wurde.

```
LOCK TIMEOUT REPORT

Date:     06/11/2006
Time:     18:44:34
Instance: juntang
Database: MYDB2
Database Partition:000
```

```
Lock Information:

   Lock Name:      0002000200000070000000052
   Lock Type:      Row
   Lock Specifics: Table space ID=2, Table ID=2, Row ID=00000007


Lock Requestor:
   System Auth ID:          JUNTANG
   Application Handle:      [1-53]
   Application ID:          *N1.juntang.061106234422
   Application Name:        db2bp
   Requesting Agent ID:    1024116
   Coordinator Agent ID:   0
   Coordinator Partition:  1
   Lock timeout Value:      2000 milliseconds
   Lock mode requested:     .NS
   Application status:      (SQLM_UOWEXEC)
   Current Operation:       (SQLM_FETCH)
   Lock Escalation:          No


   Context of Lock Request:
   Identification:            UOW ID (2); Activity ID (1)
   Activity Information:
      Package Schema:         (NULLID  )
      Package Name:           (SQLC2E07)
      Package Version:        ()
      Section Entry Number: 201
      SQL Type:               Dynamic
      Statement Type:         DML, Select (blockable)
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:              select * from T1

Lock Owner (Representative):
   System Auth ID:          JUNTANG
   Application Handle:      [0-55]
   Application ID:          *N0.juntang.061106234256
   Application Name:        db2bp
   Requesting Agent ID: 1102052
   Coordinator AgentID:      1102052
   Coordinator Partition:  0
   Lock mode held:          ..X

   List of Active SQL Statements:

      Entry                   #1
      Identification:         UOW ID (2); Activity ID (2)
      Package Schema:         (NULLID  )
      Package Name:           (SQLC2E07)
      Package Version:        ()
      Section Entry Number: 1
      SQL Type:               Dynamic
      Statement Type:         DML, Select (blockable)
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:              select tabname from syscat.tables

   List of Inactive SQL Statements from current UOW:

      Entry                   #1
      Identification:         UOW ID (2); Activity ID (4)
      Package Schema:         (NULLID  )
      Package Name:           (SQLC2E07)
```

```
      Package Version:        ()
      Section Entry Number: 201
      SQL Type:               Dynamic
      Statement Type:         DML, Select (blockable)
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:              select * from T1

      Entry                   #2
      Identification:         UOW ID (2); Activity ID (3)
      Package Schema:         (NULLID  )
      Package Name:           (SQLC2E07)
      Package Version:        ()
      Section Entry Number: 203
      SQL Type:               Dynamic
      Statement Type:         DML, Insert/Update/Delete
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:              insert into T1 values 7

      Entry                   #3
      Identification:         UOW ID (2); Activity ID (1)
      Package Schema:         (NULLID  )
      Package Name:           (SQLC2E07)
      Package Version:        ()
      Section Entry Number: 203
      SQL Type:               Dynamic
      Statement Type:         DML, Insert/Update/Delete
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:              insert into T1 values 8
```

Szenario 3: Statisches SQL verwenden

Dieses Beispiel zeigt den Inhalt eines Berichts für Zeitlimitüberschreitungen für Sperren, wobei ein Anweisungsprotokoll erfasst wird. In diesem Fall haben sowohl der Sperrenanforderer als auch der Sperreninhaber statische SQL-Anweisungen ausgeführt. Zur Feststellung der ausgeführten SQL-Anweisungen können Sie die Katalogsicht SYSCAT.STATEMENTS mit einer ähnlichen Abfrage abfragen, die für Zeile 35 im Abschnitt **Beschreibung des Berichts für Zeitlimitüberschreitungen für Sperren** aufgeführt ist.

```
LOCK TIMEOUT REPORT

Date:     05/11/2006
Time:     14:22:27
Instance: svtdbm9
Database: ABSOLUT
Database Partition: 000


Lock Information:

   Lock Name:      0007000B000481070000000052
   Lock Type:      Row
   Lock Specifics: Table space ID=7, Table ID=11, Row ID=00048107


Lock Requestor:
   System Auth ID:          SVTDBM9
   Application Handle:      [0-77]
   Application ID:          *N0.svtdbm9.061105192016
   Application Name:        am171c
   Requesting Agent ID:    1179836
   Coordinator Agent ID:   7131156
   Coordinator Partition:  0
```

```
   Lock timeout Value:      5000 milliseconds
   Lock mode requested:     ..U
   Application status:      (SQLM_UOWEXEC)
   Current Operation:       (SQLM_FETCH)
   Lock Escalation:          No

   Context of Lock Request:
   Identification:            UOW ID (1); Activity ID (3)
   Activity Information:
      Package Schema:         (ABSOLUT )
      Package Name:           (AM4RC_L0)
      Package Version:        ()
      Section Entry Number: 3
      SQL Type:               Static
      Statement Type:         DML, Select
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:              Not Available

Lock Owner (Representative):
   System Auth ID:          SVTDBM9
   Application Handle:      [0-78]
   Application ID:          *N0.svtdbm9.061105192017
   Application Name:        am25c
   Requesting Agent ID:    6111474
   Coordinator Agent ID:   6111474
   Coordinator Partition:  0
   Lock mode held:          ..X

   List of Active SQL Statements:

      Entry                   #1
      Identification:         UOW ID (1); Activity ID (4)
      Package Schema:         (ABSOLUT )
      Package Name:           (AM0PC_L0)
      Package Version:        ()
      Section Entry Number: 3
      SQL Type:               Static
      Statement Type:         DML, Insert/Update/Delete
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:

   List of Inactive SQL Statements from current UOW:

      Entry                   #1
      Identification:         UOW ID (1); Activity ID (3)
      Package Schema:         (ABSOLUT )
      Package Name:           (AM0PC_L0)
      Package Version:        ()
      Section Entry Number: 4
      SQL Type:                Static
      Statement Type:         DML, Insert/Update/Delete
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
      Statement:

      Entry                   #2
      Identification:         UOW ID (1); Activity ID (2)
      Package Schema:         (ABSOLUT )
      Package Name:           (AM0PC_L0)
      Package Version:        ()
      Section Entry Number: 1
      SQL Type:                Static
      Statement Type:         DML, Insert/Update/Delete
      Effective Isolation:    Cursor Stability
      Statement Unicode Flag: No
```

```
Statement:

Entry                   #3
Identification:         UOW ID (1); Activity ID (1)
Package Schema:         (ABSOLUT )
Package Name:           (AM0PC_L0)
Package Version:        ()
Section Entry Number:  2
SQL Type:                Static
Statement Type:         DML, Insert/Update/Delete
Effective Isolation:    Cursor Stability
Statement Unicode Flag: No
Statement:
```

Sperrenumwandlung

Die Änderung des Modus einer bereits aktivierten Sperre wird als *Umwandlung* bezeichnet. Die Sperrenumwandlung erfolgt, wenn ein Prozess auf ein Datenobjekt zugreift, für das er bereits eine Sperre aktiviert hat, und der Zugriffsmodus eine noch stärker einschränkende als die aktuelle Sperre erfordert. Für einen Prozess kann immer nur eine Sperre für ein Datenobjekt aktiv sein, obwohl er mehrfach eine Sperre für dasselbe Datenobjekt indirekt durch eine Abfrage anfordern kann.

Einige Sperrmodi gelten nur für Tabellen, andere nur für Zeilen oder Blöcke. Für Zeilen oder Blöcke findet in der Regel eine Umwandlung statt, wenn eine X-Sperre benötigt wird und eine S- oder U-Sperre (Update) zurzeit aktiviert ist.

Sperren der Modi IX (Intent Exclusive) und S (Shared) stellen jedoch in Bezug auf die Sperrenumwandlung einen Sonderfall dar. Weder der Modus S noch der Modus IX wird als stärker einschränkend angesehen als der jeweils andere Modus. Wenn einer dieser Modi aktiv ist und der andere angefordert wird, erfolgt eine Sperrenumwandlung in den Modus SIX (Share with Intent Exclusive). Alle anderen Umwandlungen werden so ausgeführt, dass der angeforderte Sperrmodus zum Modus der aktiven Sperre wird, wenn der angeforderte Modus einen höheren Grad der Einschränkung bewirkt.

Es kann auch eine doppelte Umwandlung stattfinden, wenn eine Abfrage eine Zeile aktualisiert. Wenn die Zeile über einen Indexzugriff gelesen wird und im Modus S gesperrt ist, hat die Tabelle, die diese Zeile enthält, eine abdeckende Intent-Sperre. Wenn der Sperrentyp jedoch IS und nicht IX ist und die Zeile nachfolgend geändert wird, wird die Tabellensperre in eine IX-Sperre und die Zeilensperre in eine X-Sperre umgewandelt.

Eine Sperrenumwandlung findet normalerweise implizit bei der Ausführung einer Abfrage statt. Das Verständnis der verschiedenen Arten von Sperren für verschiedene Abfragen sowie Tabellen- und Indexkombinationen kann beim Entwurf und bei der Optimierung von Anwendungen hilfreich sein.

Die Systemmonitorelemente *lock_current_mode* und *lock_mode* können Informationen zu Sperrenumwandlungen bereitstellen, die in Ihrer Datenbank stattfinden.

Verhindern von auf Sperren bezogenen Leistungsproblemen

Berücksichtigen Sie die folgenden Richtlinien, wenn Sie das Sperren im Hinblick auf den gemeinsamen Zugriff und die Datenintegrität optimieren:

- Erstellen Sie kleine UOWs (UOW = Unit of Work, Arbeitseinheit) mit häufigen Anweisungen COMMIT, um den gemeinsamen Zugriff auf Daten durch zahlreiche Benutzer zu fördern.

Nehmen Sie Anweisungen COMMIT in die Anwendung an den Stellen auf, an denen sich die Anwendung logisch in einem Konsistenzzustand befindet, das heißt, wenn die geänderten Daten konsistent sind. Wenn eine Anweisung COMMIT abgesetzt wird, werden Sperren freigegeben, abgesehen von Tabellensperren, die mit der Klausel WITH HOLD deklarierten Cursorn zugeordnet sind.

- Schließen Sie einen mit WITH HOLD deklarierten Cursor, indem Sie eine COMMIT-Anweisung absetzen.

  In manchen Situationen bleiben Sperren bestehen, nachdem die Ergebnismenge geschlossen und die Transaktion mit COMMIT festgeschrieben wurde. Durch das Schließen eines mit WITH HOLD definierten Cursors vor dem Absetzen der COMMIT-Anweisung wird sichergestellt, dass Sperren freigegeben werden.

- Führen Sie INSERT-Anweisungen als separate UOWs aus.

  In manchen Situationen bleiben Sperren bestehen, nachdem die Ergebnismenge geschlossen und die Transaktion mit COMMIT festgeschrieben wurde. Durch die Ausführung von INSERT-Anweisungen als separate UOWs wird sichergestellt, dass Sperren freigegeben werden.

- Geben Sie eine geeignete Isolationsstufe an.

  Sperren werden aktiviert, auch wenn die Anwendung Zeilen lediglich liest, sodass auch UOWs im Lesezugriff mit COMMIT festgeschrieben werden müssen. Der Grund dafür ist der, dass in den Isolationsstufen RR, RS und CS in Anwendungen mit Lesezugriff gemeinsame Sperren (Shared) aktiviert werden. Bei den Isolationsstufen RR und RS werden alle Sperren beibehalten, bis eine Anweisung COMMIT ausgeführt wird, sodass keine anderen Prozesse die gesperrten Daten aktualisieren können, es sei denn, Sie schließen den Cursor mit der Klausel WITH RELEASE. Außerdem werden auch Katalogsperren in Anwendungen mit dynamischen SQL- oder XQuery-Anweisungen unter der Isolationsstufe UR aktiviert.

  Der Datenbankmanager stellt sicher, dass die Anwendung keine noch nicht festgeschriebenen Daten (Zeilen, die von anderen Anwendungen aktualisiert, aber noch nicht mit der Anweisung COMMIT festgeschrieben wurden) abruft, sofern Sie nicht mit der Isolationsstufe UR (Uncommitted Read) arbeiten.

- Verwenden Sie die Anweisung LOCK TABLE in geeigneter Weise.

  Diese Anweisung sperrt eine ganze Tabelle. Es wird nur die in der Anweisung LOCK TABLE angegebene Tabelle gesperrt. Übergeordnete und abhängige Tabellen der angegebenen Tabelle werden nicht gesperrt. Sie müssen selbst entscheiden, ob das Sperren anderer Tabellen, auf die möglicherweise zugegriffen wird, erforderlich ist, um das gewünschte Ergebnis hinsichtlich des gemeinsamen Zugriffs und der Leistung zu erzielen. Die Sperre wird erst freigegeben, wenn für die UOW ein Commit oder ein Rollback durchgeführt wurde.

  LOCK TABLE IN SHARE MODE

  Sie wollen auf Daten zugreifen, die *zeitlich konsistent* sind, d. h. Daten, die für eine Tabelle zu einem bestimmten Zeitpunkt aktuell sind. Wenn auf die Tabelle häufig zugegriffen wird, liegt die einzige Möglichkeit, die Tabelle in einem stabilen Zustand zu erhalten, darin, die Tabelle zu sperren. Die Anwendung soll zum Beispiel eine Momentaufnahme der Tabelle machen. Während der Zeit jedoch, die die Anwendung zur Verarbeitung einiger Zeilen der Tabelle benötigt, aktualisieren andere Anwendungen Zeilen, die die Anwendung noch nicht verarbeitet hat. Dies ist unter der Isolationsstufe RR (Wiederholtes Lesen) zulässig, jedoch nicht beabsichtigt.

  Als alternative Methode kann Ihre Anwendung die Anweisung LOCK TABLE IN SHARE MODE absetzen: Es können keine Zeilen geändert werden, egal ob sie von Ihnen abgerufen wurden oder nicht. Anschlie-

ßend kann die Anwendung die benötigte Anzahl von Zeilen abrufen und gleichzeitig davon ausgehen, dass die abgerufenen Zeilen nicht kurz vor dem Abrufen geändert wurden.

Nach Ausführung der Anweisung LOCK TABLE IN SHARE MODE können andere Benutzer Daten aus der Tabelle abrufen, jedoch keine Zeilen in der Tabelle aktualisieren, löschen oder einfügen.

LOCK TABLE IN EXCLUSIVE MODE
Sie beabsichtigen, einen großen Teil der Tabelle zu aktualisieren. Es ist weniger aufwendig und daher effektiver, alle anderen Benutzer vom Zugriff auf die Tabelle auszuschließen, als jede Zeile zur Aktualisierung zu sperren und anschließend die Sperren wieder freizugeben, wenn alle Änderungen festgeschrieben sind.

Durch LOCK TABLE IN EXCLUSIVE MODE werden alle anderen Benutzer ausgeschlossen, d. h. keine anderen Anwendungen können auf die Tabelle zugreifen, außer wenn es sich um Anwendungen mit der Isolationsstufe UR (nicht festgeschriebener Lesevorgang) handelt.

- Verwenden Sie Anweisungen ALTER TABLE in Anwendungen.

  Die Anweisung ALTER TABLE mit dem Parameter LOCKSIZE stellt eine Alternative zur Anweisung LOCK TABLE dar. Über den Parameter LOCKSIZE können Sie eine Sperrgranularität für den nächsten Tabellenzugriff angeben. Gültige Werte sind ROW für Zeilensperre oder TABLE für Tabellensperre. Für MDC-Tabellen können Sie eine Sperrgranularität der Klausel BLOCKINSERT angeben.

  Die Auswahl von Zeilensperren (ROW) entspricht der Auswahl der Standardsperrgranularität bei der Erstellung einer Tabelle. Die Auswahl von Tabellensperren (TABLE) verbessert eventuell die Leistung von Abfragen durch die dadurch verringerte Anzahl benötigter Sperren. Allerdings könnte der gemeinsame Zugriff eingeschränkt werden, weil alle Sperren für die gesamte Tabelle aktiviert werden. Bei MDC-Tabellen kann die Auswahl der Klausel BLOCKINSERT die Leistung von INSERT-Operationen durch Sperren auf Blockebene und Vermeiden von Zeilensperren für Einfügungen verbessern. Sperren auf Zeilenebene werden immer noch für alle anderen Operationen durchgeführt; die Durchführung findet für Einfügungen von Schlüsseln statt, um RR-Scanner zu schützen. Die Option BLOCKINSERT ist für große Einfügungen in Zellen durch einzelne Transaktionen nützlich. Keine der LOCKSIZE-Auswahlmöglichkeiten verhindert eine normale Sperreneskalation.

- Schließen Sie Cursor, um die von ihnen aktivierten Sperren freizugeben.

  Wenn Sie einen Cursor mit der Anweisung CLOSE CURSOR schließen, die die Klausel WITH RELEASE enthält, versucht der Datenbankmanager alle vorhandenen Lesesperren freizugeben, die für diesen Cursor aktiviert sind. Tabellenlesesperren sind IS-, S- und U-Tabellensperren. Zeilenlesesperren sind S-, NS- und U-Zeilensperren. Blocklesesperren sind IS-, S- und U-Blocksperren.

  Die Klausel WITH RELEASE hat keine Auswirkung auf Cursor, die unter den Isolationsstufen CS oder UR verwendet werden. Wenn die Klausel WITH RELEASE für Cursor angegeben wird, die unter der Isolationsstufe RS oder RR verwendet werden, hebt sie einige Merkmale dieser Isolationsstufen auf. Dies heißt insbesondere, dass beim einem RS-Cursor das Phänomen *nicht wiederholbarer Lesevorgänge* und bei einem RR-Cursor das Phänomen *nicht wiederholbarer Lesevorgänge* bzw. das Phänomen von *Phantomzeilen* auftreten können.

  Wird ein Cursor, der ursprünglich ein RR- oder RS-Cursor ist, nachdem er mit der Klausel WITH RELEASE geschlossen wurde, erneut geöffnet, werden neue Lesesperren aktiviert.

In CLI-Anwendungen kann das DB2-CLI-Verbindungsattribut SQL_ATTR_CLOSE_BEHAVIOR verwendet werden, um die gleichen Ergebnisse wie mit CLOSE CURSOR WITH RELEASE zu erzielen.

- Wenn Sie in einer Umgebung mit partitionierten Datenbanken die Konfigurationsparameter ändern, die sich auf Sperren auswirken, stellen Sie sicher, dass die Änderungen in allen Datenbankpartitionen durchgeführt werden.

Korrigieren von Problemen der Sperreneskalation

Der Datenbankmanager kann Sperren automatisch von Zeilen- oder Blockebene auf Tabellenebene eskalieren (hochstufen). Bei partitionierten Tabellen kann Sperren automatisch von Zeilen- oder Blockebene auf Datenpartitionsebene eskalieren. Der Datenbankkonfigurationsparameter *maxlocks* gibt an, wann die Sperreneskalation ausgelöst wird. Die Tabelle, für die die Sperre aktiviert wird, die die Sperreneskalation auslöst, kann selbst unberührt bleiben. Sperren werden zuerst für die Tabelle mit den meisten Sperren eskaliert, angefangen bei Tabellen, für die LOB- und LONG VARCHAR-Deskriptoren gesperrt sind, dann für die Tabelle mit der nächsthöchsten Anzahl von Sperren usw., bis die Anzahl der aktiven Sperren auf ungefähr die Hälfte des durch *maxlocks* definierten Werts gesenkt wurde.

In einer adäquat entwickelten Datenbank kommt eine Sperreneskalation nur selten vor. Wenn die Sperreneskalation jedoch den gemeinsamen Zugriff auf ein nicht akzeptables Maß reduziert (worauf das Monitorelement *lock_escalation* bzw. der Diagnoseanzeiger *db.lock_escal_rate* hinweist), müssen Sie das Problem analysieren und entscheiden, wie es zu lösen ist.

Stellen Sie sicher, dass Informationen zur Sperreneskalation aufgezeichnet werden. Setzen Sie den Konfigurationsparameter *notifylevel* (Benachrichtigungsstufe) des Datenbankmanagers auf den Wert 3 (Standardwert) oder 4. Wenn *notifylevel* den Wert 2 hat, wird nur der SQLCODE-Wert des Fehlers protokolliert. Wenn bei der Benachrichtigungsstufe 3 oder 4 eine Sperreneskalation fehlschlägt, werden Informationen zum SQLCODE-Wert des Fehlers sowie zur Tabelle aufgezeichnet, für die die Eskalation fehlgeschlagen ist. Die aktuelle Abfrageanweisung wird nur protokolliert, wenn es sich um eine momentan ausgeführte dynamische Abfrageanweisung handelt und der Parameter *notifylevel* auf den Wert 4 gesetzt ist.

Befolgen Sie zur Diagnose der Ursache für nicht akzeptable Sperreneskalationen sowie zur Anwendung von Abhilfemaßnahmen die folgenden Schritte:

1. Analysieren Sie das Benachrichtigungsprotokoll für die Systemverwaltung für alle Tabellen, für die Sperren eskaliert werden. Diese Protokolldatei enthält die folgenden Informationen:
   - Die Anzahl der momentan aktiven Sperren.
   - Die Anzahl der bis zum Ende der Sperreneskalation erforderlichen Sperren.
   - Die Tabellenkennung und der Tabellenname jeder eskalierten Tabelle.
   - Die Anzahl der momentan aktiven Sperren für andere Objekte als Tabellen.
   - Die neue Sperre auf Tabellenebene, die als Teil der Eskalation aktiviert werden soll. Dabei handelt es sich in der Regel um eine Sperre im Modus „S," (Share) oder „X," (eXclusive).
   - Der interne Rückkehrcode für das Ergebnis des Aktivierens der neuen Tabellensperrenebene.
2. Verwenden Sie die Informationen im Benachrichtigungsprotokoll für die Systemverwaltung, um zu entscheiden, wie das Eskalationsproblem zu lösen ist. Ziehen Sie die folgenden Möglichkeiten in Betracht:

- Erhöhen der Anzahl der global zulässigen Sperren durch Heraufsetzen des Werts des Parameters *maxlocks*, des Parameters *locklist* oder beider Parameter in der Konfigurationsdatei der Datenbank. In einer partitionierten Datenbank führen Sie diese Änderungen in allen Datenbankpartitionen aus.

  Sie könnten diese Methode wählen, wenn der gleichzeitige Zugriff auf die Tabelle durch andere Prozesse von höchster Wichtigkeit ist. Jedoch kann der Systemaufwand, der mit der Aktivierung von Sperren auf Datensatzebene verbunden ist, zu größeren Verzögerungen für andere Prozesse führen, als durch den gleichzeitigen Zugriff auf eine Tabelle an Zeit eingespart werden kann.
- Anpassen des Prozesses bzw. der Prozesse, die die Eskalation verursacht haben. Für diese Prozesse könnten Sie die Anweisung LOCK TABLE explizit absetzen.
- Ändern der Isolationsstufe. Beachten Sie, dass diese Maßnahme jedoch zu einer Verringerung des gemeinsamen Zugriffs führen kann.
- Erhöhen der Häufigkeit von Commits zur Verringerung der Anzahl von Sperren, die zu einem gegebenen Zeitpunkt aktiv sind.
- Verwenden häufiger COMMIT-Anweisungen für Transaktionen, die LONG VARCHAR-Daten und verschiedene Arten von LOB-Daten anfordern. Diese Art von Daten wird zwar erst vom Datenträger gelesen, wenn die Ergebnismenge im Speicher erstellt wird, jedoch wird der Deskriptor bereits beim ersten Verweis auf die Daten gesperrt. Infolgedessen werden möglicherweise wesentlich mehr Sperren aktiviert als für Zeilen, die gängigere Arten von Daten enthalten.

Auswerten nicht festgeschriebener Daten durch Sperrenverzögerung

Zur Verbesserung des gemeinsamen Zugriffs lässt DB2 jetzt die Verzögerung von Zeilensperren für Suchen mit der Isolationsstufe CS oder RS in einigen Fällen zu, bis ein Datensatz festgestellt wird, der die Vergleichselemente einer Abfrage erfüllt. Standardmäßig sperrt DB2 bei Verwendung der Zeilensperrung während einer Tabellen- oder Indexsuche jede durchsuchte Zeile, deren Festschreibestatus (COMMIT-Status) unbekannt ist, bevor festgestellt wird, ob die Zeile der Abfrage entspricht. Zur Verbesserung des gemeinsamen Zugriffs durch Suchoperationen ist es möglich, die Zeilensperre zu verzögern, bis festgestellt wurde, dass eine Zeile einer Abfrage entspricht.

Zur Nutzung dieser Funktion aktivieren Sie die Registrierdatenbankvariable DB2_EVALUNCOMMITTED.

Wenn diese Variable aktiviert ist, kann eine Auswertung von Vergleichselementen für nicht festgeschriebene Daten stattfinden. Das bedeutet, dass eine Zeile, die eine nicht festgeschriebene Aktualisierung enthält, die Abfrage vielleicht nicht erfüllt, während sie der Abfrage möglicherweise entsprechen würde, wenn mit der Auswertung der Vergleichselemente gewartet würde, bis die aktualisierte Transaktion beendet ist. Darüber hinaus werden nicht festgeschriebene gelöschte Zeilen bei Tabellensuchen übersprungen. DB2 überspringt gelöschte Schlüssel beim Durchsuchen von Indizes des Typs 2, wenn die Registrierdatenbankvariable DB2_SKIPDELETED aktiviert ist.

Diese Registrierdatenbankvariableneinstellungen gelten bei der Kompilierung für dynamische SQL- oder XQuery-Anweisungen sowie beim Binden für statische SQL- oder XQuery-Anweisungen. Dies bedeutet, dass selbst wenn die Registrierdatenbankvariable bei der Ausführung aktiviert ist, die Sperrenvermeidungsstrategie

nicht angewendet wird, wenn die Registrierdatenbankvariable DB2_EVALUNCOMMITTED beim Binden nicht aktiviert war. Wenn die Registrierdatenbankvariable beim Binden aktiviert, bei der Ausführung jedoch nicht aktiviert ist, wird die Sperrenvermeidungsstrategie trotzdem angewendet. Für statische SQL- oder XQuery-Anweisungen gilt beim Rebind eines Pakets die Einstellung der Registrierdatenbankvariablen zum Zeitpunkt des Bindens. Ein impliziter Rebind von statischen SQL- oder XQuery-Anweisungen verwendet die aktuelle Einstellung der Registrierdatenbankvariablen DB2_EVALUNCOMMITTED.

Anwendbarkeit der Auswertung nicht festgeschriebener Daten für verschiedene Zugriffspläne

Tabelle 8. Zugriff nur über Satz-ID-Index

Vergleichselemente	Nicht festgeschriebene Daten auswerten
Keine	Nein
Als Suchargument verwendbare Vergleichselemente	Ja

Tabelle 9. Reiner Datenzugriff (relational oder mit verzögerter Satz-ID-Liste)

Vergleichselemente	Nicht festgeschriebene Daten auswerten
Keine	Nein
Als Suchargument verwendbare Vergleichselemente	Ja

Tabelle 10. Satz-ID-Index und Datenzugriff

Vergleichselemente		Nicht festgeschriebene Daten auswerten	
Index	**Daten**	**Indexzugriff**	**Datenzugriff**
Keine	Keine	Nein	Nein
Keine	Als Suchargument verwendbare Vergleichselemente	Nein	Nein
Als Suchargument verwendbare Vergleichselemente	Keine	Ja	Nein
Als Suchargument verwendbare Vergleichselemente	Als Suchargument verwendbare Vergleichselemente	Ja	Nein

Tabelle 11. Blockindex und Datenzugriff

Vergleichselemente		Nicht festgeschriebene Daten auswerten	
Index	**Daten**	**Indexzugriff**	**Datenzugriff**
Keine	Keine	Nein	Nein
Keine	Als Suchargument verwendbare Vergleichselemente	Nein	Ja
Als Suchargument verwendbare Vergleichselemente	Keine	Ja	Nein
Als Suchargument verwendbare Vergleichselemente	Als Suchargument verwendbare Vergleichselemente	Ja	Ja

Beispiel

Das folgende Beispiel zeigt einen Vergleich der Standardfunktionsweise von Sperren und der neuen Funktionsweise mit Auswertung nicht festgeschriebener Daten.

Die folgende Tabelle ist die Tabelle ORG aus der Beispieldatenbank SAMPLE.

```
DEPTNUMB DEPTNAME       MANAGER DIVISION   LOCATION
-------- -------------- ------- ---------- -------------
      10 Head Office        160 Corporate  New York
      15 New England         50 Eastern    Boston
      20 Mid Atlantic        10 Eastern    Washington
      38 South Atlantic      30 Eastern    Atlanta
      42 Great Lakes        100 Midwest    Chicago
      51 Plains             140 Midwest    Dallas
      66 Pacific            270 Western    San Francisco
      84 Mountain           290 Western    Denver
```

Die folgenden Transaktionen werden an dieser Tabelle mit der Standardisolationsstufe der Cursorstabilität (CS) ausgeführt.

Tabelle 12. Transaktionen an der Tabelle ORG mit der Isolationsstufe CS

SITZUNG 1	SITZUNG 2
`connect to SAMPLE`	`connect to SAMPLE`
`+c update org set  deptnumb=5 where manager=160`	
	`select * from org where deptnumb >= 10`

Die nicht festgeschriebene UPDATE-Operation in Sitzung 1 aktiviert eine exklusive Satzsperre für die erste Zeile in der Tabelle, welche die SELECT-Abfrage in Sitzung 2 daran hindert, zurückzukehren, obwohl die in Sitzung 1 zu aktualisierende Zeile momentan die Abfrage in Sitzung 2 nicht erfüllt. Das liegt daran, dass die Isolationsstufe CS festlegt, dass jede Zeile, auf die durch eine Abfrage zugegriffen wird, gesperrt werden muss, während sich der Cursor auf dieser Zeile befindet. Sitzung 2 kann solange keine Sperre für die erste Zeile aktivieren, bis Sitzung 1 ihre Sperre freigibt.

Beim Durchsuchen der Tabelle kann das Warten auf die Sperre in Sitzung 2 durch die Verwendung der Funktion zur Auswertung nicht festgeschriebener Daten vermieden werden. Dabei wird zunächst das Vergleichselement ausgewertet und anschließend eine Sperre für die Zeile aktiviert, um das Vergleichselement endgültig auszuwerten. In diesem Fall würde die Abfrage in Sitzung 2 nicht versuchen, die erste Zeile in der Tabelle zu sperren, sodass der gemeinsame Zugriff durch Anwendungen verbessert wird. Beachten Sie, dass dies auch bedeuten würde, dass die Auswertung des Vergleichselements in Sitzung 2 im Hinblick auf den nicht festgeschriebenen Wert mit deptnumb=5 in Sitzung 1 erfolgen würde. Die Abfrage in Sitzung 2 würde die erste Zeile aus ihrer Ergebnismenge herauslassen, obwohl ein Rollback der UPDATE-Operation in Sitzung 1 die Abfrage in Sitzung 2 erfüllen würde.

Wenn die Reihenfolge der Operationen umgekehrt wäre, ergäbe sich mit der Funktion zur Auswertung nicht festgeschriebener Daten trotzdem eine Verbesserung für den gemeinsamen Zugriff. Bei der Standardfunktionsweise von Sperren würde Sitzung 2 zunächst eine Zeilensperre anfordern, die die Ausführung der Aktualisierung mit Suche in Sitzung 1 verhindern würde, obwohl die Aktualisierung in Sitzung 1 die von der Abfrage in Sitzung 2 gesperrte Zeile nicht ändern würde. Wenn die Aktualisierung mit Suche in Sitzung 1 versuchen würde, die Zeilen zunächst

zu untersuchen und sie nur zu sperren, wenn sie die Vergleichselemente erfüllen, würde die Abfrage in Sitzung 1 nicht blockiert.

Einschränkungen

Die folgenden externen Einschränkungen gelten für diese neue Funktionalität:

- Die Registrierdatenbankvariable DB2_EVALUNCOMMITTED muss aktiviert sein.
- Die Isolationsstufe muss CS oder RS sein.
- Zeilensperren müssen verwendet werden.
- Als Suchargument verwendbare Vergleichselemente sind vorhanden.
- Die Auswertung nicht festgeschriebener Daten gilt nicht für Suchen in den Katalogtabellen.
- Für MDC-Tabellen können Blocksperren für eine Indexsuche verzögert werden. Bei Tabellensuchen werden Blocksperren jedoch nicht verzögert.
- Das verzögerte Sperren tritt nicht für eine Tabelle auf, die eine Inplace-Tabellenreorganisation ausführt.
- Das verzögerte Sperren findet nicht bei einer Indexsuche statt, wenn es sich um einen Index des Typs 1 handelt.
- Für Iscan-Fetch-Pläne werden Zeilensperren nicht bis zum Datenzugriff verzögert, sondern die Zeile wird beim Indexzugriff gesperrt, bevor auf die Zeile in der Tabelle zugegriffen wird.
- Gelöschte Zeilen werden bei Tabellensuchen bedingungslos übersprungen, während gelöschte Schlüssel von Indizes des Typs 2 nur übersprungen werden, wenn die Registrierdatenbankvariable DB2_SKIPDELETED aktiviert ist.

Option zum Ignorieren nicht festgeschriebener Einfügungen

Die Registrierdatenbankvariable DB2_SKIPINSERTED steuert, ob nicht festgeschriebene Einfügungen für Cursor ignoriert werden können, die die Isolationsstufe Cursorstabilität oder Lesestabilität verwenden. Das DB2-Datenbanksystem kann nicht festgeschriebene Einfügungen wie folgt handhaben:

- Das DB2-Datenbanksystem kann so lange warten, bis die INSERT-Transaktion durchgeführt (d. h. mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht) wurde, und die Daten dementsprechend verarbeiten. Die Standardoption ist OFF.

  In den folgenden Beispielen sehen Sie Instanzen, bei denen die Standardoption OFF bevorzugt wird:

  – Angenommen, zwei Anwendungen verwenden eine Tabelle, um Daten untereinander auszutauschen, wobei die erste Anwendung Daten in die Tabelle einfügt und die zweite Anwendung die Daten liest. Die Daten müssen von der zweiten Anwendung in der Reihenfolge in der Tabelle so verarbeitet werden, damit, falls die nächste zu lesende Zeile von der ersten Anwendung eingefügt wurde, die zweite Anwendung warten muss, bis die eingefügten Daten festgeschrieben werden. In einem solchen Fall muss der Standardwert für DB2_SKIPINSERTED verwendet werden.

  – Angenommen, eine Anwendung nimmt Änderungen an Daten vor, indem die Daten gelöscht und das neue Datenimage eingefügt wird. In den Fällen, in denen UPDATE-Anweisungen vermieden werden, muss der Standardwert für DB2_SKIPINSERTED verwendet werden.

- Das DB2-Datenbanksystem kann nicht festgeschriebene Einfügungen ignorieren, was in vielen Fällen den gemeinsamen Zugriff verbessern kann. Wenn Sie dieses Verhalten wünschen, muss die Registrierdatenbankvariable auf ON gesetzt sein.

Im Allgemeinen wird durch ON der gemeinsame Zugriff verbessert für die meisten Anwendungen bevorzugt verwendet. Wenn die Registrierdatenbankvariable aktiviert ist, werden nicht festgeschriebene eingefügte so Zeilen betrachtet, als wären sie noch nicht eingefügt worden.

Sperrtypenkompatibilität

Die Sperrenkompatibilität wird zu einem Problem, wenn eine Anwendung eine Sperre für ein Objekt aktiviert hat und eine weitere Anwendung eine Sperre für dasselbe Objekt anfordert. Wenn die beiden Sperrmodi kompatibel sind, kann die Anforderung für eine zweite Sperre für das Objekt zugelassen werden.

Wenn der Sperrmodus der angeforderten Sperre nicht mit der bereits aktiven Sperre kompatibel ist, kann die Sperrenanforderung nicht erfüllt werden. Stattdessen muss die Anforderung warten, bis die erste Anwendung die Sperre freigegeben hat und alle weiteren bestehenden inkompatiblen Sperren freigegeben wurden.

Die folgende Tabelle enthält Informationen zu den Umständen, in denen eine Sperrenanforderungen erfüllt werden kann, wenn ein anderer Prozess eine Sperre für dieselbe Ressource in einem bestimmten Status bereits aktiviert hat oder gerade anfordert. Ein **Nein** bedeutet, dass der Anforderer warten muss, bis alle inkompatiblen Sperren von anderen Prozessen freigegeben werden. Beachten Sie, dass es zu einer Zeitlimitüberschreitung kommen kann, wenn ein Anforderer auf eine Sperre wartet. Ein **Ja** bedeutet, dass die Sperre zugelassen wird, sofern kein früherer Anforderer auf die Ressource wartet.

Tabelle 13. Kompatibilität der Sperrmodi

	Status der aktiven Sperre											
Angeforderter Status	**Keiner**	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**X**	**Z**	**NW**	**W**
Keiner	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
IN	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Ja	Ja
IS	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Nein
NS	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Ja	Nein
S	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Nein	Nein
IX	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein
SIX	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
U	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein
X	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
Z	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
NW	Ja	Ja	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja
W	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein

Tabelle 13. Kompatibilität der Sperrmodi (Forts.)

	Status der aktiven Sperre											
Angeforderter Status	**Keiner**	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**X**	**Z**	**NW**	**W**

Anmerkung:

I	Intent
N	None
NS	Next Key Share
S	Share
X	Exclusive
U	Update
Z	Super Exclusive
NW	Next Key Weak Exclusive
W	Weak Exclusive

Anmerkung:

- Ja - Angeforderte Sperre wird sofort **erteilt**.
- Nein - Anforderer muss **warten**, bis aktive die Sperre freigegeben oder das Zeitlimit überschritten wird.

Sperrmodi und Zugriffspfade für Standardtabellen

Dieser Abschnitt enthält Referenzinformationen zu Sperrmethoden für Standardtabellen für verschiedene Datenzugiffspläne.

In den folgenden Tabellen werden die Typen von Sperren aufgelistet, die für Standardtabellen in der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag besteht aus zwei Teilen: Tabellensperre und Zeilensperre. Ein Strich zeigt an, dass für eine bestimmte Stufe keine Sperrung erfolgt.

Anmerkung:

1. In einer MDC-Umgebung (Multidimensional Clustering) wird eine zusätzliche Sperrstufe verwendet: der Block.
2. Sperrmodi können explizit mit einer Sperranforderungsklausel (LOCK REQUEST) in einer SELECT-Anweisung geändert werden.

Tabelle 14. Sperrmodi für Tabellensuchen ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	S/-	U/-	SIX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X

Tabelle 15. Sperrmodi für Tabellensuchen mit Vergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	S/-	U/-	SIX/X	U/-	SIX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

Anmerkung: Wenn in der Isolationsstufe UR mit IN-Sperren für Indizes des Typs 1 Vergleichselemente für INCLUDE-Spalten im Index angewendet werden, werden die Isolationsstufe auf CS erhöht und die Sperren in eine IS-Tabellensperre und in NS-Zeilensperren umgewandelt.

Tabelle 16. Sperrmodi für Satz-ID-Indexsuchen (RID) ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	S/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X

Tabelle 17. Sperrmodi für Satz-ID-Indexsuchen (RID) mit einer einzigen Qualifikationszeile

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IS/S	IX/U	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X

Tabelle 18. Sperrmodi für Satz-ID-Indexsuchen (RID) nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IS/S	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X

Tabelle 18. Sperrmodi für Satz-ID-Indexsuchen (RID) nur mit Start- und Stoppvergleichselementen (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
UR	IN/-	IX/U	IX/X	IX/X	IX/X

Tabelle 19. Sperrmodi für Satz-ID-Indexsuchen (RID) nur mit Index und anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IS/S	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

Die folgenden Tabellen zeigen die Sperrmodi für Fälle, in denen der Lesezugriff auf die Datenseiten verzögert erfolgt, um folgende Operationen an der Zeilenliste durchführen zu können:

- Eine weitere Qualifizierung mithilfe mehrerer Indizes
- Eine Sortierung für einen effizienten Vorablesezugriff

Tabelle 20. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IS/S	IX/S		X/-	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

Tabelle 21. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten: nach einer Satz-ID-Indexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IN/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X

Tabelle 21. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten: nach einer Satz-ID-Indexsuche ohne Vergleichselemente (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X

Tabelle 22. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IS/S	IX/S		IX/S	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

Tabelle 23. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IS/S	IX/S		IX/X	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

Tabelle 24. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IN/-	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IS/-	IX/U	IX/X	IX/U	IX/X

Tabelle 25. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten: nach einer Satz-ID-Indexsuche mit Vergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IN/-	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen

In einer MDC-Umgebung (Multidimensional Clustering) wird eine zusätzliche Sperrstufe verwendet: der Block. In den folgenden Tabellen werden die Typen von Sperren aufgelistet, die in der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag besteht aus drei Teilen: Tabellensperre, Blocksperre und Zeilensperre. Ein Strich zeigt an, dass für eine bestimmte Stufe keine Sperrung erfolgt.

Anmerkung: Sperrmodi können explizit mit einer Sperranforderungsklausel (LOCK REQUEST) in einer SELECT-Anweisung geändert werden.

Tabelle 26. Sperrmodi für Tabellensuchen ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

Tabelle 27. Sperrmodi für Tabellensuchen mit Vergleichselementen nur mit Dimensionsspalten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-

Tabelle 28. Sperrmodi für Tabellensuchen mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Die beiden folgenden Tabellen zeigen Sperrmodi für Satz-ID-Indizes für MDC-Tabellen.

Tabelle 29. Sperrmodi für Satz-ID-Indexsuchen (RID) ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

Tabelle 30. Sperrmodi für Satz-ID-Indexsuchen (RID) mit einer einzigen Qualifikationszeile

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

Tabelle 31. Sperrmodi für Satz-ID-Indexsuchen (RID) nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

Tabelle 32. Sperrmodi für Satz-ID-Indexsuchen (RID) nur mit Indexvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Tabelle 33. Sperrmodi für Satz-ID-Indexsuchen (RID) mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Anmerkung: In den folgenden Tabellen, die Sperrmodi für Satz-ID-Indexsuchen für verzögerte Datenseitenzugriffe anzeigen, wird die Isolationsstufe auf CS erhöht und die Sperren in eine IS-Tabellensperre, eine IS-Blocksperre und NS-Zeilensperren hochgestuft, wenn die Isolationsstufe UR mit einer IN-Sperre für Indizes des Typs 1 vorliegt oder wenn es Vergleichselemente für INCLUDE-Spalten im Index gibt.

Tabelle 34. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 35. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Satz-ID-Indexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

Tabelle 36. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 37. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Tabelle 38. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 38. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
UR	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 39. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Sperren für Blockindexsuchen für MDC-Tabellen

In den folgenden Tabellen werden die Typen von Sperren aufgelistet, die in der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag besteht aus drei Teilen: Tabellensperre, Blocksperre und Zeilensperre. Ein Strich zeigt an, dass für eine bestimmte Stufe keine Sperrung erfolgt.

Anmerkung: Sperrmodi können explizit mit einer Sperranforderungsklausel (LOCK REQUEST) in einer SELECT-Anweisung geändert werden.

Tabelle 40. Sperrmodi für Indexsuchen ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--

Tabelle 41. Sperrmodi für Indexsuchen nur mit Dimensionsvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

Tabelle 42. Sperrmodi für Indexsuchen nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-

Tabelle 43. Sperrmodi für Indexsuchen mit Vergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Die folgende Tabelle listet Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten auf:

Tabelle 44. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 45. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--

Tabelle 46. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche nur mit Dimensionsvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

Tabelle 47. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche nur mit Dimensionsvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--

Tabelle 48. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 48. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche nur mit Start- und Stoppvergleichselementen (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
UR	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 49. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	

Tabelle 50. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 51. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Sperrverhalten für partitionierte Tabellen

Zusätzlich zu einer Sperre für die gesamte Tabelle wird eine Sperre für jede Datenpartition einer partitionierten Tabelle aktiviert. Auf diese Weise können die zu sperrenden Bereiche besser differenziert und der gemeinsame Zugriff im Vergleich zu einer nicht partitionierten Tabelle erhöht werden. Die neue Datenpartitionssperre wird in der Ausgabe des Befehls 'db2pd', des Ereignismonitors, der Verwaltungssichten und der Tabellenfunktionen ausgewiesen.

Beim Zugriff auf eine Tabelle wird von der Sperrfunktion zuerst die Tabellensperre gesetzt. Anschließend werden die erforderlichen Datenpartitionssperren gesetzt, die sich nach den Daten richten, auf die zugegriffen werden soll. Aufgrund der verwendeten Zugriffsmethoden und Isolationsstufen ist es möglich, dass Datenpartitionen gesperrt werden müssen, die sich nicht in der Ergebnismenge befinden. Wenn diese Datenpartitionssperren aktiviert wurden, können sie möglicherweise ebenso lange wie die Tabellensperre beibehalten werden. Bei einem Indexsuchvorgang mit der Isolationsstufe CS (Cursorstabilität) werden die Sperren für zuvor verwendete Datenpartitionen vieleicht beibehalten, um den Aufwand für das erneute Sperren einer Datenpartition zu vermeiden, wenn diese Datenpartition in nachfolgenden Schlüsseln referenziert wird. Die Datenpartitionssperre umfasst auch den Aufwand für die Sicherstellung des Zugriffs auf die Tabellenbereiche. Bei nicht partitionierten Tabellen wird der Zugriff auf Tabellenbereiche durch die Tabellensperre gesteuert. Aus diesem Grund werden Datenpartitionen auch dann gesperrt, wenn eine exklusive oder den gemeinsamen Zugriff zulassende (SHARE) Sperre auf Tabellenebene für eine partitionierte Tabelle aktiviert wurde.

Durch die feinere Differenzierung kann eine Transaktion exklusiven Zugriff auf eine bestimmte Datenpartition haben und Zeilensperren vermeiden, während andere Transaktionen auf die anderen verfügbaren Datenpartitionen zugreifen können. Dieser Fall kann infolge eines für eine Massenaktualisierung ausgewählten Zugriffsplans oder durch eine Eskalation von Sperren auf die Datenpartitionsebene eintreten. Als Tabellensperre für zahlreiche Zugriffsmethoden wird normalerweise eine Intent-Sperre verwendet. Dies gilt auch dann, wenn für die Datenpartitionen eine den gemeinsamen Zugriff zulassende oder eine exklusive Sperre aktiviert wird. Auf diese Weise kann der gemeinsame Zugriff verbessert werden. Wenn jedoch auf Datenpartitionsebene Nicht-Intent-Sperren erforderlich sind und im Zugriffsplan angegeben ist, dass unter Umständen auf alle Datenpartitionen zugegriffen wird, wird möglicherweise auf Tabellenebene eine Nicht-Intent-Sperre ausgewählt, um Deadlocks zwischen Datenpartitionssperren bei gleichzeitig zugreifenden Transaktionen zu vermeiden.

Sperren für die SQL-Anweisung LOCK TABLE

Bei partitionierten Tabellen werden für die Anweisung LOCK TABLE ausschließlich Sperren auf Tabellenebene aktiviert. Datenpartitionssperren werden hingegen nicht aktiviert. Auf diese Weise wird sichergestellt, dass bei nachfolgenden DML-Anweisungen keine Zeilensperren für die Tabelle aktiviert werden. Außerdem werden so Deadlocks auf Zeilen-, Block- oder Datenpartitionsebene vermieden. Mit der Anweisung LOCK TABLE IN EXCLUSIVE MODE kann beim Aktualisieren von Indizes ebenfalls ein exklusiver Zugriff sichergestellt werden. Dies ist sinnvoll, um das Anwachsen von Indizes des Typs 2 während umfangreicher Aktualisierungsoperationen zu begrenzen.

Auswirkung des Parameters LOCKSIZE TABLE der Anweisung ALTER TABLE

Die Anweisung ALTER TABLE verfügt über die Option zum Einstellen von LOCKSIZE TABLE, durch die sichergestellt werden kann, dass für die Tabelle eine den gemeinsamen Zugriff zulassende (SHARE) oder eine exklusive Sperre ohne Intent-Sperren definiert wird. Bei einer partitionierten Tabelle wird diese Sperrenstrategie sowohl bei der Tabellensperre als auch bei den Datenpartitionssperren für alle betroffenen Datenpartitionen angewendet.

Zeilen- und Blocksperreneskalation

Bei partitionierten Tabellen werden durch eine Sperreneskalation Zeilen- und Blocksperren in Datenpartitionssperren umgewandelt. Dies bedeutet wiederum, dass andere Transaktionen besser auf die Tabelle zugreifen können, selbst wenn für eine Datenpartition eine Eskalation auf Sperren mit gemeinsamem Zugriff oder exklusive Sperren (Share, Exclusive oder Super Exclusive) erfolgt, andere Datenpartitionen, die nicht eskaliert werden, davon jedoch unberührt bleiben. Nach einer Eskalation in einer bestimmten Datenpartition setzt die Transaktion die Aktivierung von Zeilensperren in anderen Datenpartitionen möglicherweise fort. Die Benachrichtigungsprotokollnachricht für Eskalationen umfasst Informationen zu der Datenpartition, in der die Eskalation aufgetreten ist, sowie zum Namen der betroffenen partitionierten Tabelle. Aus diesem Grund kann der exklusive Zugriff auf einen Index durch die Sperreneskalation nicht sichergestellt werden. Entweder muss in der Anweisung eine exklusive Sperre auf Tabellenebene verwendet, eine explizite Anweisung LOCK TABLE IN EXCLUSIVE MODE ausgeführt oder von der Tabelle das Attribut LOCKSIZE TABLE verwendet werden. Die Gesamttabellensperre für die Zugriffsmethode wird vom Optimierungsprogramm ausgewählt und ist vom Ausschluss von Datenpartitionen abhängig. Eine umfangreiche Aktualisierung an einer Tabelle kann zur Auswahl einer exklusiven Tabellensperre führen, wenn kein Ausschluss von Datenpartitionen erfolgt.

Interpretieren von Informationen über Sperren

Das folgende Beispiel aus der Verwaltungssicht SNAPLOCK kann Sie bei der Interpretation der für eine partitionierte Tabelle zurückgegebenen Informationen über Sperren unterstützen.

Beispiel 1:

Die folgende Verwaltungssicht SNAPLOCK wurde während einer Offline-Indexreorganisation erfasst.

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE,
LOCK_MODE, LOCK_ESCALATION FROM SYSIBMADM.SNAPLOCK where TABNAME like 'TP1' and LOCK_OBJECT_TYPE like 'TABLE_%'
ORDER BY TABNAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, TAB_FILE_ID, LOCK_MODE
```

```
TABNAME         TAB_FILE_ID          TBSP_NAME       DATA_PARTITION_ID LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_ESCALATION
--------------- -------------------- --------------- ----------------- ------------------ ---------- ---------------
TP1                           32768 -                              -1 TABLE_LOCK         Z                       0
TP1                               4 USERSPACE1                     0 TABLE_PART_LOCK    Z                       0
TP1                               5 USERSPACE1                     1 TABLE_PART_LOCK    Z                       0
TP1                               6 USERSPACE1                     2 TABLE_PART_LOCK    Z                       0
TP1                               7 USERSPACE1                     3 TABLE_PART_LOCK    Z                       0
TP1                               8 USERSPACE1                     4 TABLE_PART_LOCK    Z                       0
TP1                               9 USERSPACE1                     5 TABLE_PART_LOCK    Z                       0
TP1                              10 USERSPACE1                     6 TABLE_PART_LOCK    Z                       0
TP1                              11 USERSPACE1                     7 TABLE_PART_LOCK    Z                       0
TP1                              12 USERSPACE1                     8 TABLE_PART_LOCK    Z                       0
TP1                              13 USERSPACE1                     9 TABLE_PART_LOCK    Z                       0
TP1                              14 USERSPACE1                    10 TABLE_PART_LOCK    Z                       0
```

```
TP1                             15 USERSPACE1                  11 TABLE_PART_LOCK    Z                       0
TP1                              4 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                              5 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                              6 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                              7 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                              8 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                              9 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                             10 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                             11 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                             12 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                             13 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                             14 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                             15 USERSPACE1                   - TABLE_LOCK         Z                       0
TP1                             16 USERSPACE1                   - TABLE_LOCK         Z                       0

  26 Satz/Sätze ausgewählt.
```

In diesem Beispiel werden der Sperrobjekttyp TABLE_LOCK und die Datenpartitions-ID (DATA_PARTITION_ID) mit dem Wert -1 verwendet, um den Zugriff auf die partitionierte Tabelle TP1 und die gleichzeitige Verwendung dieser Tabelle zu steuern. Die Sperrobjekte des Typs TABLE_PART_LOCK dienen zur Steuerung des größtmöglichen Zugriffs auf jede Datenpartition und ihrer größtmöglichen gemeinsamen Nutzung.

Darüber hinaus werden in dieser Ausgabe weitere Sperrobjekte des Typs TABLE_LOCK erfasst (mit TAB_FILE_ID 4 bis 16), die keinen Wert für DATA_PARTITION_ID haben. Eine Sperre dieses Typs, bei der ein Objekt mit einer TAB_FILE_ID und einem TBSP_NAME einer Datenpartition oder einem Index für die partitionierte Tabelle entsprechen, kann verwendet werden, um den gleichzeitigen Zugriff mit dem Online-Backup-Dienstprogramm zu steuern.

Faktoren mit Auswirkungen auf Sperren

Die folgenden Faktoren beeinflussen den Modus und die Granularität von Sperren des Datenbankmanagers:

- Die Art der Verarbeitung, die von der Anwendung ausgeführt wird
- Die Datenzugriffsmethode
- Der Typ der Indizes: 1 oder 2
- Verschiedene Konfigurationsparameter

Sperren und Arten der Anwendungsverarbeitung

Zum Zweck der Bestimmung von Sperrenattributen kann die Verarbeitung durch Anwendungen einem der folgenden Typen zugeordnet werden:

- Lesezugriff

  Dieser Typ umfasst alle SELECT-Anweisungen, die von sich aus nur Lesezugriff haben, eine explizite Klausel FOR READ ONLY enthalten oder mehrdeutig sind, jedoch vom Abfragecompiler aufgrund der im Befehl PREP oder BIND angegebenen Option BLOCKING als Anweisungen mit Lesezugriff eingestuft werden. Für diesen Verarbeitungstyp sind nur S-Sperren (S, NS oder IS) erforderlich.
- Änderungsabsicht

  Dieser Typ umfasst alle SELECT-Anweisungen mit der Klausel FOR UPDATE, der Klausel USE AND KEEP UPDATE LOCKS, der Klausel USE AND KEEP EXCLUSIVE LOCKS bzw. Anweisungen, die der Abfragecompiler als mehrdeutige Anweisung interpretiert, um anzunehmen, dass eine Änderung beabsichtigt wird. Für diesen Typ werden Share- und Update-Sperren (S, U und X für Zeilen; IX, U, X und S für Blöcke; IX, U und X für Tabellen) verwendet.
- Änderung

Dieser Typ umfasst Anweisungen mit UPDATE, INSERT und DELETE, aber keine Anweisungen mit UPDATE WHERE CURRENT OF oder DELETE WHERE CURRENT OF. Für diesen Typ sind exklusive Sperren (X oder IX) erforderlich.

- Cursorgesteuert

  Dieser Typ umfasst Anweisungen mit UPDATE WHERE CURRENT OF und DELETE WHERE CURRENT OF. Für diesen Typ sind ebenfalls Sperren des exklusiven Modus (X oder IX) erforderlich.

Eine Anweisung, die an einer Zieltabelle Einfüge-, Aktualisierungs- oder Löschoperationen (INSERT, UPDATE oder DELETE) auf der Grundlage des Ergebnisses eines Subselects vornimmt, führt zwei Typen der Verarbeitung aus. Die Regeln für Lesezugriffsverarbeitung bestimmen die Sperren für die Tabellen, die in der Anweisung des Subselects zurückgegeben werden. Die Regeln für die Änderungsverarbeitung bestimmen die Sperren für die Zieltabelle.

Sperren und Datenzugriffsmethoden

Ein *Zugriffsplan* ist die Methode, die das Optimierungsprogramm zum Abrufen von Daten aus einer bestimmten Tabelle auswählt. Der Zugriffsplan kann erhebliche Auswirkungen auf Sperrmodi haben. Wenn zum Beispiel eine Indexsuche zum Auffinden einer bestimmten Zeile verwendet wird, wählt das Optimierungsprogramm wahrscheinlich Sperren auf Zeilenebene (IS) für die Tabelle aus. Wenn die Tabelle EMPLOYEE zum Beispiel einen Index für die Spalte EMPNO hat, könnte ein Zugriff über einen Index ausgeführt werden, um Informationen zu einem einzelnen Mitarbeiter mithilfe einer Anweisung auszuwählen, die die folgende SELECT-Klausel enthält:

```
SELECT *
  FROM EMPLOYEE
  WHERE EMPNO = '000310';
```

Wenn kein Index verwendet wird, muss die gesamte Tabelle der Reihe nach durchsucht werden, um die ausgewählten Zeilen zu finden. Dafür ist möglicherweise nur eine Sperre auf Tabellenebene (S) erforderlich. Wenn zum Beispiel kein Index für die Spalte SEX vorhanden ist, könnte eine Tabellensuche verwendet werden, um alle männlichen Mitarbeiter mit einer Anweisung auszuwählen, die folgende SELECT-Klausel enthält:

```
SELECT *
  FROM EMPLOYEE
  WHERE SEX = 'M';
```

Anmerkung: Bei der cursorgesteuerten Verarbeitung wird der Sperrmodus des zugrunde liegenden Cursors verwendet, bis die Anwendung eine Zeile findet, die zu aktualisieren oder zu löschen ist. Für diesen Verarbeitungstyp wird unabhängig vom Sperrmodus eines Cursors immer eine Sperre des Modus Exclusive aktiviert, um die Aktualisierung oder Löschung durchzuführen.

Das Sperren in Bereichsclustertabellen funktioniert etwas anders als beim Sperren von Standardschlüsseln und nächsten Schlüsseln. Beim Zugriff auf einen Bereich von Zeilen in einer Bereichsclustertabelle werden alle Zeilen in diesem Bereich gesperrt, selbst wenn einige dieser Zeilen leer sind. Beim Sperren von Standardschlüsseln oder nächsten Schlüsseln werden nur Zeilen mit vorhandenen Datensätzen gesperrt.

Referenztabellen enthalten detaillierte Informationen darüber, welche Sperren für welche Art von Zugriffsplan aktiviert werden.

Beim verzögerten Zugriff auf die Datenseiten erfolgt der Zugriff auf die Zeile in zwei Schritten, wodurch komplexere Sperrsituationen auftreten. Die Zeitpunkte der Aktivierung von Sperren und die Dauer der Sperren hängen von der Isolationsstufe ab. Da bei der Isolationsstufe RR (*Wiederholtes Lesen*) alle Sperren bis zum Ende der Transaktion beibehalten werden, bleiben die im ersten Schritt aktivierten Sperren bestehen, und im zweiten Schritt müssen keine weiteren Sperren aktiviert werden. Für die Isolationsstufen Lesestabilität und Cursorstabilität müssen Sperren im zweiten Schritt aktiviert werden. Um den gemeinsamen Zugriff zu maximieren, werden im ersten Schritt keine Sperren aktiviert und alle Vergleichselemente erneut angewandt, sodass nur den Auswahlbedingungen entsprechende Zeilen zurückgegeben werden.

Indextypen und Sperren des nächsten Schlüssels

Wenn Transaktionen Änderungen an Indizes des Typs 1 verursachen, treten Sperren der jeweils nächsten Schlüssel (Next-Key Locking) auf. Für Indizes des Typs 2 findet das Sperren nächster Schlüssel nur in minimalem Ausmaß statt.

- Sperren der nächsten Schlüssel für Indizes des Typs 2:

  Das Sperren eines nächsten Schlüssels findet statt, wenn ein Schlüssel in einen Index eingefügt wird.

  Während der Einfügung eines Schlüssels in einen Index, wird die Zeile, die dem Schlüssel entspricht, der auf den neuen Schlüssel im Index als Nächstes folgt, nur gesperrt, wenn diese Zeile momentan durch eine RR-Indexsuche gesperrt ist. Der Sperrmodus, der für die Sperre des nächsten Schlüssels verwendet wird, ist NW. Diese Sperre des nächsten Schlüssels wird wieder freigegeben, bevor die eigentliche Einfügung des Schlüssels erfolgt. Die Einfügung des Schlüssels erfolgt, wenn eine Zeile in eine Tabelle eingefügt wird.

  Wenn Aktualisierungen an einer Zeile zu einer Änderung am Wert des Indexschlüssels für diese Zeile führen, erfolgt die Einfügung des Schlüssels außerdem deshalb, weil der ursprüngliche Schlüsselwert als gelöscht markiert wird und der neue Schlüsselwert in den Index eingefügt wird. Bei Aktualisierungen, die nur die INCLUDE-Spalten eines Index betreffen, kann der Schlüssel an seinem Platz aktualisiert werden, und es findet keine Sperrung des nächsten Schlüssels statt.

  Bei RR-Suchen wird die Zeile, die dem Schlüssel entspricht, der auf das Ende des Suchbereichs folgt, im S-Modus gesperrt. Wenn auf das Ende des Suchbereichs keine Schlüssel folgen, wird eine Tabellenende-Sperre aktiviert, um das Ende des Index zu sperren. Wenn der Schlüssel, der auf das Ende des Suchbereichs folgt, als gelöscht markiert ist, fährt die Suche mit dem Sperren der entsprechenden Zeilen so lange fort, bis ein Schlüssel angetroffen wird, der nicht als gelöscht markiert ist, wenn die Suche die entsprechende Zeile für diesen Schlüssel sperrt, oder so lange, bis das Ende des Index gesperrt ist.

- Sperren der nächsten Schlüssel für Indizes des Typs 1:

  Sperren der nächsten Schlüssel werden bei Lösch- und Einfügeoperationen an Indizes und bei Indexsuchen aktiviert. Wenn eine Zeile in einer Tabelle aktualisiert, aus ihr gelöscht oder in sie eingefügt wird, wird für diese Zeile eine X-Sperre aktiviert. Bei Einfügungen kann der Sperrmodus auf eine W-Sperre herabgesetzt werden.

  Wenn der Schlüssel aus dem Tabellenindex gelöscht oder in ihn eingefügt wird, wird die Tabellenzeile gesperrt, die dem Schlüssel entspricht, der auf den gelöschten oder eingefügten Schlüssel im Index folgt. Bei Aktualisierungen, die den Wert des Schlüssels betreffen, wird der ursprüngliche Schlüsselwert zuerst gelöscht und der neue Wert eingefügt, sodass zwei Sperren für nächste Schlüssel aktiviert werden. Die Dauer dieser Sperren ist wie folgt festgelegt:

– Beim Löschen eines Indexschlüssels ist der Sperrmodus für den nächsten Schlüssel X und die Sperre bleibt bis zum Commit aktiv.
– Beim Einfügen eines Indexschlüssels ist der Sperrmodus für den nächsten Schlüssel NW. Diese Sperre wird nur aktiviert, wenn es eine Konkurrenzsituation für die Sperre gibt. In diesem Fall wird die Sperre freigegeben, bevor der Schlüssel tatsächlich in den Index eingefügt wird.
– Bei RR-Suchen wird die Tabellenzeile, die dem Schlüssel direkt nach dem Ende des Indexsuchbereichs entspricht, im Modus S gesperrt und die Sperre bis zum Commit beibehalten.
– Bei CS/RS-Suchen wird die Zeile, die dem Schlüssel direkt nach dem Ende des Indexsuchbereichs entspricht, im Modus NS gesperrt, wenn eine Konkurrenzsituation für die Sperre vorliegt. Diese Sperre wird freigegeben, wenn das Ende des Suchbereichs einmal geprüft ist.

Das Sperren nächster Schlüssel für Indizes des Typs 1 bei Einfügungen und Löschungen von Schlüsseln kann zu Deadlocks führen. Das folgende Beispiel zeigt, wie sich zwei Transaktionen gegenseitig sperren könnten. Bei Indizes des Typs 2 treten solche Deadlocks nicht auf.

Betrachten Sie das folgende Beispiel eines Index, der sechs Zeilen mit den folgenden Werten enthält: `1 5 6 7 8 12`.

1. Transaktion 1 löscht die Zeile mit dem Schlüsselwert 8. Die Zeile mit dem Wert 8 wird im Modus X gesperrt. Wenn der entsprechende Schlüssel aus dem Index gelöscht wird, wird die Zeile mit dem Wert 12 im Modus X gesperrt.
2. Transaktion 2 löscht die Zeile mit dem Schlüsselwert 5. Die Zeile mit dem Wert 5 wird im Modus X gesperrt. Wenn der entsprechende Schlüssel aus dem Index gelöscht wird, wird die Zeile mit dem Wert 6 im Modus X gesperrt.
3. Transaktion 1 fügt eine Zeile mit dem Schlüsselwert 4 ein. Diese Zeile wird im Modus W gesperrt. Wenn versucht wird, den neuen Schlüssel in den Index einzufügen, wird die Zeile mit dem Wert 6 im Modus NW gesperrt. Dieser Sperrversuch muss auf die X-Sperre warten, die von Transaktion 2 für diese Zeile aktiviert wurde.
4. Transaktion 2 fügt eine Zeile mit dem Schlüsselwert 9 ein. Diese Zeile wird im Modus W gesperrt. Wenn versucht wird, den neuen Schlüssel in den Index einzufügen, wird die Zeile mit dem Schlüsselwert 12 im Modus NW gesperrt. Dieser Sperrversuch muss auf die X-Sperre warten, die von Transaktion 1 für diese Zeile aktiviert wurde.

Bei der Verwendung von Indizes des Typs 1 führt ein solches Szenario zu einem Deadlock, und für eine der Transaktionen wird ein Rollback durchgeführt.

Angabe einer Strategie für den Modus 'Wartestatus für Sperre'

Eine einzelne Sitzung kann nun eine Strategie für den Modus 'Wartestatus für Sperre' angeben, die verwendet wird, wenn für die Sitzung eine Sperre erforderlich ist, die nicht unmittelbar abgerufen werden kann. Die Strategie gibt an, ob die Sitzung folgende Bedingungen erfüllt:

- Zurückgeben von SQLCODE und SQLSTATE, wenn keine Sperre abgerufen werden kann
- Andauerndes Warten auf eine Sperre
- Warten auf eine Sperre über einen angegebenen Zeitraum
- Verwenden des Werts für den Datenbankkonfigurationsparameter *locktimeout* beim Warten auf eine Sperre

Die Strategie für den Modus 'Wartestatus für Sperre' wird über die neue Anweisung SET CURRENT LOCK TIMEOUT angegeben, durch die der Wert des Sonderregisters CURRENT LOCK TIMEOUT geändert wird. Das Sonderregister CURRENT LOCK TIMEOUT gibt die Anzahl an Sekunden an, die auf eine Sperre gewartet werden soll, bevor ein Fehler zurückgegeben wird, der darauf hinweist, dass keine Sperre abgerufen werden kann.

Traditionelle Sperrmethoden können Anwendungen zur Folge haben, die sich gegenseitig blockieren. Dies geschieht dann, wenn eine Anwendung auf eine andere Anwendung warten muss, um die Sperre freizugeben. Strategien zum Umgang mit Auswirkungen auf solche Blockungen stellen in der Regel einen Mechanismus bereit, um die annehmbare Höchstdauer des Blocks anzugeben. Dies ist die Menge an Zeit, die eine Anwendung vor der Rückgabe ohne Sperre wartet. Vorher war dies nur auf der Datenbankebene durch Ändern des Werts des Datenbankkonfigurationsparameters *locktimeout* möglich.

Der Wert des Parameters *locktimeout* gilt für alle Sperren; die Sperrtypen, die diese neue Funktion enthält, umfassen Blocksperren für Zeilen, Tabellen, Indexschlüssel und MDC.

Optimieren von Anwendungen

Richtlinien zur Begrenzung von SELECT-Anweisungen

Das Optimierungsprogramm geht von der Annahme aus, dass eine Anwendung sämtliche Zeilen abrufen muss, die in der SELECT-Anweisung angegeben sind. Diese Annahme ist in OLTP-Umgebungen und bei Stapelbetrieb in der Regel zutreffend. Bei reinen Such- bzw. Anzeigeanwendungen („Browse") hingegen werden durch die Abfragen häufig potenziell sehr umfangreiche Antwortmengen definiert, von denen jedoch normalerweise nur so viele Zeilen abgerufen werden, wie auf eine Bildschirmanzeige passen.

Zur Verbesserung der Leistung solcher Anwendungen können Sie die SELECT-Anweisung auf folgende Weisen modifizieren:

- Verwenden Sie die Klausel FOR UPDATE, um die Spalten anzugeben, die von einer nachfolgenden positionierten Anweisung UPDATE aktualisiert werden könnten.
- Verwenden Sie die Klausel FOR READ/FETCH ONLY, um die angegebenen Spalten im Lesezugriff zurückzuliefern.
- Verwenden Sie die Klausel OPTIMIZE FOR *n* ROWS, um dem Abrufen der ersten *n* Zeilen der Gesamtergebnismenge Priorität geben.
- Verwenden Sie die Klausel FETCH FIRST *n* ROWS ONLY, um nur eine angegebene Anzahl von Zeilen abzurufen.
- Verwenden Sie die Anweisung DECLARE CURSOR WITH HOLD, um jeweils nur eine Zeile pro Mal abzurufen.

Anmerkung: Die Verwendung der Klausel FOR UPDATE, FETCH FIRST *n* ROWS ONLY oder OPTIMIZE FOR *n* ROWS bzw. die Deklaration des Cursors mit SCROLL hat Auswirkung auf die Zeilenblockung.

In den folgenden Abschnitten werden die Leistungsvorteile der einzelnen Methoden erläutert.

Klausel FOR UPDATE

Die Klausel FOR UPDATE begrenzt die Ergebnismenge ein, indem sie nur die Spalten berücksichtigt, die durch eine nachfolgende positionierte UPDATE-Anweisung aktualisiert werden können. Bei der Angabe der Klausel FOR UPDATE ohne Spaltennamen werden alle Spalten, die aktualisiert werden können, in der Tabelle oder Sicht mit eingeschlossen. Bei Angabe von Spaltennamen muss jeder Name unqualifiziert angegeben werden und eine Spalte der Tabelle oder Sicht bezeichnen.

Unter folgenden Umständen können Sie die Klausel FOR UPDATE nicht verwenden:

- Wenn der Cursor, der der der SELECT-Anweisung zugeordnet ist, nicht gelöscht werden kann.
- Wenn mindestens eine der durch SELECT ausgewählten Spalten eine Spalte ist, die nicht in einer Katalogtabelle aktualisiert werden kann und in der Klausel FOR UPDATE nicht ausgeschlossen wurde.

Verwenden Sie in CLI-Anwendungen für gleiche Zwecke das DB2 CLI-Verbindungsattribut SQL_ATTR_ACCESS_MODE.

Klausel FOR READ oder FETCH ONLY

Die Klausel FOR READ ONLY oder FOR FETCH ONLY stellt sicher, dass Ergebnisse nur im Lesezugriff zugeliefert werden. Da die Ergebnistabelle aus einer SELECT-Operation für eine Sicht, die als schreibgeschützt definiert ist, ebenfalls schreibgeschützt ist, ist diese Klausel zulässig, jedoch hat sie keinen Effekt.

Bei Tabellen, für die Aktualisierungen und Löschungen zulässig sind, kann die Angabe der Klausel FOR READ ONLY die Leistung von Abrufoperationen (FETCH) verbessern, wenn der Datenbankmanager Blöcke von Daten abrufen kann, statt exklusive Sperren zu aktivieren. Verwenden Sie die Klausel FOR READ ONLY nicht für Abfragen, in denen positionierte UPDATE- oder DELETE-Anweisungen verwendet werden.

Das DB2 CLI-Verbindungsattribut SQL_ATTR_ACCESS_MODE kann in CLI-Anwendungen für dieselben Zwecke verwendet werden.

Klausel OPTIMIZE FOR n ROWS

Mit der Klausel OPTIMIZE FOR wird die Absicht deklariert, nur eine Teilmenge des Ergebnisses abzurufen oder dem Abruf nur der ersten wenigen Zeilen Priorität zu geben. Das Optimierungsprogramm kann in diesem Fall Zugriffspläne bevorzugen, die die Antwortzeiten für den Abruf der ersten wenigen Zeilen minimieren. Darüber hinaus wird die Anzahl der Zeilen, die als ein Block an den Client gesendet werden, durch den Wert „n" der Klausel OPTIMIZE FOR begrenzt. Daher beeinflusst die Klausel OPTIMIZE FOR sowohl die Art und Weise, wie der Server die den Bedingungen entsprechenden Zeilen aus der Datenbank abruft, als auch die Art und Weise, wie er diese Zeilen an den Client zurückgibt.

Nehmen Sie zum Beispiel an, Sie führen regelmäßig eine Abfrage nach den Mitarbeitern mit den höchsten Gehältern auf die Tabelle EMPLOYEE aus.

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
```

Sie haben einen absteigenden Index für die Spalte SALARY definiert. Da jedoch die Mitarbeiter nach der Personalnummer (Spalte EMPNO) geordnet sind, weist der Index für die Spalte SALARY wahrscheinlich eine geringe Clusterbildung auf. Zur Vermeidung zahlreicher synchroner E/A-Operationen wählt das Optimierungsprogramm wahrscheinlich die Methode des Vorablesezugriffs über Listen, für die eine Sortierung der Satz-IDs (RIDs) aller qualifizierten Zeilen erforderlich ist. Diese Sortierung führt zu einer Verzögerung, bevor die ersten Ergebniszeilen an die Anwendung zurückgegeben werden. Zur Vermeidung dieser Verzögerung können Sie der Anweisung die Klausel OPTIMIZE FOR wie folgt hinzufügen:

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
OPTIMIZE FOR 20 ROWS
```

In diesem Fall geht das Optimierungsprogramm wahrscheinlich so vor, dass es den Index SALARY direkt verwendet, weil nur die 20 Mitarbeiter mit den höchsten Gehältern abgerufen werden. Unabhängig davon, wie viele Zeilen geblockt werden könnten, wird nun alle zwanzig Zeilen ein Zeilenblock an den Client zurückgegeben.

Mit der Klausel OPTIMIZE FOR bevorzugt das Optimierungsprogramm Zugriffspläne, die Massenoperationen oder die Unterbrechung des Zeilenflusses, zum Beispiel durch Sortierungen, vermeiden. Am wahrscheinlichsten wird ein Zugriffspfad durch die Klausel OPTIMIZE FOR 1 ROW beeinflusst. Die Verwendung dieser Klausel kann folgende Auswirkungen haben:

- Joinsequenzen mit zusammengesetzten inneren Tabellen treten mit geringerer Wahrscheinlichkeit auf, da für sie eine temporäre Tabelle erforderlich ist.
- Die Joinmethode könnte sich ändern. Ein Join mit Verschachtelungsschleife (Nested Loop Join) wird mit größter Wahrscheinlichkeit gewählt, da diese Methode relativ geringen Aufwand verursacht und in der Regel zum Abrufen einiger weniger Zeilen effizienter ist.
- Ein Index, der der Klausel ORDER BY entspricht, wird mit größerer Wahrscheinlichkeit genutzt, weil dadurch die Sortierung für die Klausel ORDER BY entfällt.
- Der Vorablesezugriff über Listen wird mit geringerer Wahrscheinlichkeit verwendet, da diese Zugriffsmethode eine Sortierung erforderlich macht.
- Ein sequenzieller Vorablesezugriff ist weniger wahrscheinlich, da bekannt ist, dass nur eine kleine Anzahl von Zeilen erforderlich ist.
- Bei einer Joinabfrage wird wahrscheinlich die Tabelle mit den Spalten in der Klausel ORDER BY als äußere Tabelle gewählt, wenn ein Index für die äußere Tabelle die Reihenfolge bietet, die für die Klausel ORDER BY erforderlich ist.

Obwohl die Klausel OPTIMIZE FOR für alle Optimierungsklassen gültig ist, zeigt sie für Klassen ab Optimierungsklasse 3 die besten Ergebnisse, weil die Klassen unter 3 mit der Methode der schnellen Joinaufzählung (Greedy Join Enumeration) arbeiten. Diese Methode führt manchmal zu Zugriffsplänen für Joins mehrerer Tabellen, die für ein schnelles Abrufen der ersten Zeilen nicht geeignet sind.

Die Klausel OPTIMIZE FOR bewirkt nicht, dass das Abrufen aller Ergebniszeilen unmöglich wird. Wenn Sie doch alle Ergebniszeilen abrufen, kann die Gesamtdauer wesentlich länger sein, als wenn das Optimierungsprogramm für die gesamte Antwortmenge optimiert hätte.

Wenn eine Paketanwendung die CLI (DB2 CLI oder ODBC) verwendet, können Sie das Schlüsselwort OPTIMIZEFORNROWS in der Konfigurationsdatei `db2cli.ini`

verwenden, um DB2 CLI zu veranlassen, automatisch eine Klausel OPTIMIZE FOR an das Ende jeder Abfrageanweisung anzuhängen.

Beim Auswählen von Daten aus Kurznamen können die Ergebnisse abhängig von der Unterstützung durch die Datenquelle unterschiedlich ausfallen. Wenn die durch den Kurznamen angegebene Datenquelle die Klausel OPTIMIZE FOR unterstützt und das DB2-Optimierungsprogramm die gesamte Abfrage im Pushdown-Modus an die Datenquelle sendet, wird die Klausel im fernen SQL generiert, das an die Datenquelle gesendet wird. Wenn die Datenquelle diese Klausel nicht unterstützt oder das Optimierungsprogramm entscheidet, das der Plan des geringsten Aufwands eine lokale Ausführung ist, wird die Klausel OPTIMIZE FOR lokal in DB2 angewandt. In diesem Fall bevorzugt das DB2-Optimierungsprogramm Zugriffspläne, die die Antwortzeit für das Abrufen der ersten wenigen Zeilen einer Abfrage minimieren; allerdings sind die dem Optimierungsprogramm zur Verfügung stehenden Optionen zur Generierung von Plänen etwas begrenzt, und die Leistungsgewinne aus der Klausel OPTIMIZE FOR sind möglicherweise zu vernachlässigen.

Wenn sowohl die Klausel FETCH FIRST als auch die Klausel OPTIMIZE FOR angegeben werden, wirkt sich der niedrigere der beiden Werte auf die Größe des Kommunikationspuffers aus. Die beiden Werte werden hinsichtlich der Optimierung als unabhängig voneinander betrachtet.

Klausel FETCH FIRST n ROWS ONLY

Die Klausel FETCH FIRST *n* ROWS ONLY legt die maximale Anzahl von Zeilen fest, die abgerufen werden können. Die Beschränkung der Ergebnistabelle auf die ersten wenigen Zeilen kann die Leistung erhöhen. Es werden nur *n* Zeilen abgerufen, ungeachtet der Anzahl von Zeilen, die ansonsten in der Ergebnismenge enthalten wären.

Wenn Sie sowohl die Klausel FETCH FIRST als auch die Klausel OPTIMIZE FOR angeben, wirkt sich der niedrigere der beiden Werte auf die Größe des Kommunikationspuffers aus. Hinsichtlich der Optimierung werden die beiden Werte als unabhängig voneinander betrachtet.

Anweisung DECLARE CURSOR WITH HOLD

Wenn Sie einen Cursor mit der Anweisung DECLARE CURSOR deklarieren, die die Klausel WITH HOLD enthält, bleiben alle geöffneten Cursor nach dem Commit der Transaktion geöffnet, und alle Sperren werden freigegeben, mit Ausnahme von Sperren, die die aktuelle Cursorposition geöffneter WITH HOLD-Cursor schützen.

Wenn die Transaktion mit ROLLBACK rückgängig gemacht wird, werden alle geöffneten Cursor geschlossen und alle Sperren und LOB-Querverweise freigegeben.

Durch die Verwendung des DB2 CLI-Verbindungsattributs SQL_ATTR_CURSOR-_HOLD in CLI-Anwendungen können Sie die gleichen Ergebnisse erzielen. Wenn eine Paketanwendung die CLI (DB2 CLI oder ODBC) verwendet, können Sie das Schlüsselwort CURSORHOLD in der Konfigurationsdatei `db2cli.ini` verwenden, damit DB2 CLI automatisch die Klausel WITH HOLD für jeden deklarierten Cursor annimmt.

Angeben von Zeilenblockung zur Verringerung des Systemaufwands

Zeilenblockung wird für alle Anweisungen und Datentypen, einschließlich LOB-Datentypen, unterstützt. Zeilenblockung verringert den Systemaufwand des Datenbankmanagers für Cursor durch Abrufen eines *Blocks* von Zeilen in einer einzigen Operation.

Anmerkung: Der Zeilenblock, den Sie angeben, ist eine Anzahl von Seiten im Speicher. Dabei handelt es sich nicht um einen Block einer mehrdimensionalen MDC-Tabelle (MDC - Mehrdimensionales Clustering), der wiederum einem EXTENTSIZE-Speicherbereich auf dem Datenträger zugeordnet ist.

Die Zeilenblockung wird in den folgenden Argumenten für die Befehle BIND oder PREP angegeben:

UNAMBIG
Für Cursor, die mit der Klausel FOR READ ONLY angegeben werden, wird Zeilenblockung ausgeführt.

Cursor, die nicht mit der Klausel FOR READ ONLY oder FOR UPDATE deklariert werden und die nicht *mehrdeutig* sind und *Nur-Lese-Cursor* sind, werden mit Zeilenblockung ausgeführt. *Mehrdeutige* Cursor werden nicht mit Zeilenblockung ausgeführt.

ALL Für Cursor, die mit der Klausel FOR READ ONLY angegeben werden oder nicht als FOR UPDATE deklariert werden, wird Zeilenblockung ausgeführt.

NO Es wird keine Blockung für Cursor verwendet.

Informationen zur Definition eines Nur-Lese-Cursors und eines mehrdeutigen Cursors finden Sie in der Beschreibung der Anweisung DECLARE CURSOR.

Anmerkung: Bei Verwendung der Klausel FETCH FIRST *n* ROWS ONLY oder OPTIMIZE FOR *n* ROWS in einer SELECT-Anweisung entspricht die Anzahl der Zeilen pro Block dem kleinsten der folgenden Werte:

- Der mit der oben angegebenen Formel berechnete Wert
- Der Wert von *n* in der Klausel FETCH FIRST
- Der Wert von *n* in der Klausel OPTIMIZE FOR

Zwei Konfigurationsparameter des Datenbankmanagers müssen entsprechend definiert werden. Beide Werte werden als Anzahl von Seiten im Speicher definiert. Notieren Sie sich die Werte dieser Parameter zur Verwendung in Blockgrößenberechnungen.

- Der Konfigurationsparameter *aslheapsz* des Datenbankmanagers gibt die Größe des Zwischenspeichers der Anwendungsunterstützungsebene an.
- Der Konfigurationsparameter *rqrioblk* des Datenbankmanagers gibt die Größe des Kommunikationspuffers zwischen fernen Anwendungen und ihren Datenbankagenten auf dem Datenbankserver an.

Bevor Sie die Blockung von Zeilendaten für LOB-Datentypen aktivieren, müssen Sie verstehen, welche Auswirkungen dies auf die Systemressourcen hat. Es wird mehr gemeinsam genutzter Speicher auf dem Server benötigt, um die Verweise auf die LOB-Werte in jedem Block von Daten zu speichern, wenn LOB-Spalten zurück-

gegeben werden. Die Anzahl solcher Verweise variiert entsprechend dem Wert des Datenbankkonfigurationsparameters *rqrioblk*.

Zur Erhöhung der Speicherkapazität, die dem Zwischenspeicher zugeordnet wird, ändern Sie den Datenbankkonfigurationsparameter *database_memory* wie folgt:

- Setzen Sie den Parameter auf den Wert AUTOMATIC, wenn der Datenbankmanager den Datenbankspeicher automatisch verwalten soll.
- Erhöhen Sie den Wert um 256 Seiten, wenn der Parameter momentan auf einen benutzerdefinierten Wert eingestellt ist.

Zur Erhöhung der Leistung vorhandener Anwendungen mit eingebettetem SQL, die auf LOB-Werte zugreifen, kann die Anwendung erneut gebunden werden, indem im Befehl BIND entweder die Klausel BLOCKING ALL oder die Klausel BLOCKING UNAMBIGUOUS zur Anforderung der Zeilenblockung angegeben wird. Anwendungen mit eingebettetem SQL rufen LOB-Werte vom Server zeilenweise ab, wenn ein Block von Zeilen vom Server abgerufen wurde. Benutzerdefinierte Funktionen (UDFs), die große LOB-Ergebnisse zurückgeben, können DB2 veranlassen, zum einzeiligen Abruf von LOB-Daten zurückzukehren, wenn große Mengen Speicher auf dem Server belegt werden.

Gehen Sie wie folgt vor, um die Zeilenblockung anzugeben:

1. Schätzen Sie mithilfe der Werte der Konfigurationsparameter *aslheapsz* und *rqrioblk* die Anzahl der Zeilen ab, die für jeden Block zurückgegeben werden. In den beiden folgenden Formeln steht *azl* jeweils für die Ausgabezeilenlänge.
   - Verwenden Sie die folgende Formel für lokale Anwendungen:
     ```
     Zeilen pro Block = aslheapsz * 4096 / azl
     ```
     Die Anzahl von Byte pro Seite beträgt 4.096.
   - Verwenden Sie die folgende Formel für ferne Anwendungen:
     ```
     Zeilen pro Block = rqrioblk / azl
     ```
2. Zur Aktivierung der Zeilenblockung geben Sie ein entsprechendes Argument für die Option BLOCKING in den Befehlen PREP oder BIND an.

   Wenn Sie keine Option BLOCKING angeben, wird standardmäßig der Zeilenblockungstyp UNAMBIG verwendet. Bei Verwendung des Befehlszeilenprozessors und von Call Level Interface ist der Standardtyp der Zeilenblockung ALL.

Richtlinien zur Abfrageoptimierung

Befolgen Sie die Richtlinien zur Abfrageoptimierung, um eine optimale Feinabstimmung der SQL- und XQuery-Anweisungen in einem Anwendungsprogramm zu erzielen. Die Richtlinien sollen Ihnen Hilfestellung bei der Minimierung des Bedarfs an Systemressourcen und der Zeit geben, die zur Rückgabe von Ergebnissen aus großen Tabellen und komplexen Abfragen erforderlich ist.

Anmerkung: Die vom Optimierungsprogramm verwendete Optimierungsklasse macht eine Feinabstimmung möglicherweise überflüssig, weil der Abfragecompiler den SQL- und XQuery-Code in effizientere Formate umschreiben kann.

Beachten Sie, dass die Auswahl eines Zugriffsplans durch das Optimierungsprogramm auch den Auswirkungen anderer Faktoren, wie zum Beispiel Umgebungsaspekte und Systemkatalogstatistiken, unterliegt. Durch Leistungsvergleichstests können Sie ermitteln, welche Anpassungen möglicherweise die Auswahl eines besseren Zugriffsplan bewirken.

Abfrageoptimierung mit der Bindeoption REOPT

Damit eine Abfrageoptimierung (bzw. Reoptimierung) für statische und dynamische SQL- und XQuery-Anweisungen, die Hostvariablen, Sonderregister, globale Variablen oder Parametermarken enthalten, ausgeführt werden kann, binden Sie das Paket mit der Bindeoption REOPT. Wenn diese Option verwendet wird, wird der Zugriffspfad für eine SQL- bzw. XQuery-Anweisung, die sowohl zu dem Paket gehört als auch Hostvariablen, Parametermarken, globale Variablen oder Sonderregister enthält, mit den Werten dieser Variablen, und nicht mit Standardschätzwerten, die der Compiler auswählt, optimiert. Die Optimierung erfolgt bei der Ausführung der Abfrage, wenn die Werte verfügbar sind.

Verbessern der Leistung durch Binden mit REOPT

SQL- oder XQuery-Abfragen können bei der Ausführung eine schlechte Leistung zeigen, wenn die Werte, die für die Eingabevariablen, wie zum Beispiel Parametermarken, Hostvariablen, globale Variablen und Sonderregister, verwendet werden, außerhalb des vorhersagbaren Bereichs der Schätzwerte für Standardfilterfaktoren liegen. Standardfilterfaktoren, die in Szenarios verwendet werden, in denen der tatsächliche Datenwert unbekannt ist, sind Schätzwerte für die Anzahl von Zeilen, die tatsächlich zur Laufzeit zurückgegeben werden, wenn der tatsächliche Datenwert verwendet wird.

Die Bindeoption REOPT gibt an, ob DB2 einen Zugriffspfad unter Verwendung von Werten für Hostvariablen, Parametermarken, globale Variablen und Sonderregistern optimieren soll. REOPT-Werte werden durch die folgenden Argumente in den Befehlen BIND, PREP und REBIND angegeben:

REOPT NONE
Der Zugriffspfad für eine gegebene SQL- bzw. XQuery-Anweisung, in der Hostvariablen, Parametermarken, globale Variablen oder Sonderregister enthalten sind, wird nicht mit realen Werten für diese Variablen optimiert. Stattdessen werden die Standardschätzwerte für diese Variablen verwendet. Ein solcher Plan wird in den Cache gestellt und nachfolgend verwendet. Dies ist das Standardverhalten.

REOPT ONCE
Der Zugriffspfad für eine gegebene SQL- bzw. XQuery-Anweisung wird mit den realen Werten der Hostvariablen, Parametermarken, globalen Variablen oder Sonderregister optimiert, wenn die Abfrage zum ersten Mal ausgeführt wird. Ein solcher Plan wird in den Cache gestellt und nachfolgend verwendet.

REOPT ALWAYS
Der Zugriffspfad für eine gegebene SQL- bzw. XQuery-Anweisung wird immer kompiliert und mit den bei der jeweiligen Ausführung bekannten Werten der Hostvariablen, Parametermarken, globalen Variablen oder Sonderregister optimiert.

Stichprobendaten in SQL- und XQuery-Abfragen

Datenbanken wachsen so stark an und die Abfragen für diese Datenbanken werden so komplex, dass es häufig unpraktisch und manchmal auch unnötig ist, auf sämtliche für eine Abfrage relevanten Daten zuzugreifen. In einigen Fällen interessiert sich ein Benutzer für allgemeine Trends oder Muster, bei denen angenäherte Antworten innerhalb gewisser Fehlertoleranzen vollkommen ausreichen. Eine Möglichkeit, solche Abfragen zu beschleunigen, besteht darin, die Abfrage an einer zufälligen Stichprobe der Datenbank auszuführen. Mit DB2 können Sie eine effiziente Datenstichprobe in SQL- und XQuery-Abfragen erstellen, wodurch sich poten-

zielle Leistungsverbesserungen für umfangreiche Abfragen in mehreren Größenordnungen ergeben und gleichzeitig ein hoher Grad an Genauigkeit gewahrt werden kann.

Die gängigste Anwendung von Stichprobendaten betrifft Abfragen mit Spaltenfunktionen wie AVG, SUM und COUNT, bei denen vernünftig genaue Ergebniswerte aus einer Stichprobe der Daten gewonnen werden können. Die Sichprobenerstellung kann auch dazu dienen, eine zufällige Teilmenge der tatsächlichen Zeilen einer Tabelle zu Prüfungszwecken abzurufen oder Data Mining- und Analyseoperationen zu beschleunigen.

DB2 stellt zwei Methoden zur Stichprobenerstellung bereit: Stichproben auf Zeilenebene und Stichproben auf Blockebene.

Stichprobenentnahme auf Zeilenebene nach Bernoulli

Die Bernoulli-Stichprobe auf Zeilenebene ruft eine Stichprobe von P Prozent der Tabellenzeilen mithilfe eines als Suchargument verwendbaren Vergleichselements ab, das jede Zeile mit einer Wahrscheinlichkeit von P/100 einschließt und sie mit einer Wahrscheinlichkeit von 1-P/100 ausschließt.

Die Bernoulli-Stichprobe auf Zeilenebene generiert unabhängig von den Datenwerthäufungen (Clusterbildung) immer eine gültige, zufällige Stichprobe. Allerdings ist die Leistung dieser Stichprobenmethode sehr gering, wenn kein Index verfügbar ist, da jede Zeile abgerufen und das Vergleichselement der Stichprobe auf sie angewendet werden muss. Wenn kein Index vorhanden ist, ergeben sich keine E/A-Einsparungen gegenüber der Ausführung einer Abfrage ohne Stichprobenerstellung. Wenn ein Index vorhanden ist, wird eine bessere Leistung mit dieser Art der Stichprobe erzielt, weil das Vergleichselement der Stichprobe auf die Satz-IDs (Zeilen-ID, RIDs) innerhalb der Indexblattseiten angewendet wird. Im Normalfall erfordert dies nur eine E/A-Operation pro ausgewählter RID und eine E/A-Operation pro Indexblattseite.

Stichprobenentnahme auf Systemseitenebene

Stichprobe auf Systemseitenebene ist der Stichprobe auf Zeilenebene ähnlich, jedoch werden bei ihr Stichproben der Seiten und nicht der Zeilen erstellt. Eine Seite wird mit einer Wahrscheinlichkeit von P/100 in die Stichprobe eingeschlossen. Das Einschließen einer Seite bedeutet, dass alle Zeilen dieser Seite mit eingeschlossen werden.

Die Leistung der Stichprobenentnahme auf Systemseitenebene ist hervorragend, da nur eine E/A-Operation für jede Seite erforderlich ist, die in die Stichprobe eingeschlossen wird. Im Vergleich zu keiner Stichprobenentnahme verbessert die Stichprobe auf Seitenebene die Leistung um Größenordnungen. Allerdings ist die Genauigkeit kumulierter Schätzung bei Seitenstichproben tendenziell geringer als bei Zeilenstichproben. Diese Diskrepanz in der Genauigkeit wird am deutlichsten, wenn es viele Zeilen pro Block gibt oder die in der Abfrage angegebenen Zeilen einen hohen Grad der Clusterbildung innerhalb der Seiten aufweisen.

Die beste Stichprobenmethode für einen bestimmten Zweck wird durch die Zeitvorgaben eines Benutzers sowie durch den gewünschten Grad an Genauigkeit festgelegt.

Angeben der Stichprobenmethode

Zur Ausführung einer Abfrage an einer zufälligen Stichprobe von Daten aus einer Tabelle können Sie die Klausel TABLESAMPLE der Tabellenverweisklausel in einer SQL-Anweisung verwenden. Zur Angabe der Stichprobenmethode können Sie die Schlüsselwörter BERNOULLI oder SYSTEM verwenden.

Das Schlüsselwort BERNOULLI gibt an, dass eine Bernoulli-Stichprobe auf Zeilenebene auszuführen ist.

Das Schlüsselwort SYSTEM gibt an, dass eine Stichprobe auf Systemseitenebene auszuführen ist, sofern nicht das Optimierungsprogramm feststellt, dass es effizienter ist, stattdessen eine Bernoulli-Stichprobe auf Zeilenebene auszuführen.

Parallelverarbeitung für Anwendungen

DB2 unterstützt parallele Umgebungen in erster Linie auf symmetrischen Multiprozessormaschinen (SMP-Maschinen), jedoch im begrenzten Maß auch auf Einzelprozessormaschinen. In SMP-Maschinen kann mehr als ein Prozessor auf die Datenbank zugreifen, sodass komplexe SQL-Anforderungen zur parallelen Ausführung unter den Prozessoren aufgeteilt werden können.

Verwenden Sie zur Angabe des Grades der Parallelität, die bei der Kompilierung einer Anwendung implementiert werden soll, das Sonderregister CURRENT DEGREE oder die Bindeoption DEGREE. *Grad* bezieht sich in diesem Zusammenhang auf die Anzahl der Teile einer Abfrage, die gleichzeitig ausgeführt werden. Es gibt keine strenge Beziehung zwischen der Anzahl der Prozessoren und dem Wert, den Sie für den Grad der Parallelität auswählen. Sie können einen Wert angeben, der größer oder kleiner ist als die Anzahl von Prozessoren auf der Maschine. Selbst für Einzelprozessormaschinen können Sie einen höheren Grad als 1 definieren, um die Leistung auf die eine oder andere Art zu verbessern. Beachten Sie jedoch, das jeder weitere Grad an Parallelität den Hauptspeicherbedarf und die CPU-Belastung im System erhöht.

Einige Konfigurationsparameter müssen modifiziert werden, um die Leistung zu optimieren, wenn Abfragen parallel ausgeführt werden. Insbesondere für eine Umgebung mit einem hohen Grad an Parallelität sollten Sie die Konfigurationsparameter prüfen und modifizieren, über die die Größen des gemeinsamen Speichers und des Vorablesezugriffs gesteuert werden.

Die folgenden drei Konfigurationsparameter steuern und verwalten die partitionsinterne Parallelität:

- Der Konfigurationsparameter *intra_parallel* des Datenbankmanagers aktiviert oder inaktiviert die Unterstützung für Parallelverarbeitung.
- Der Datenbankkonfigurationsparameter *max_querydegree* definiert eine obere Grenze für den Grad der Parallelität für alle Abfragen in der Datenbank. Dieser Wert hat Vorrang vor dem Wert des Sonderregisters CURRENT DEGREE und der Bindeoption DEGREE.
- Der Datenbankkonfigurationsparameter *dft_degree* definiert den Standardwert für das Sonderregister CURRENT DEGREE und die Bindeoption DEGREE.

Wenn eine Abfrage mit der Definition DEGREE = ANY kompiliert wird, wählt der Datenbankmanager den Grad der partitionsinternen Parallelität anhand einer Reihe von Faktoren aus, zu denen die Anzahl der Prozessoren und die Merkmale der Abfrage gehören. Der tatsächlich bei der Ausführung verwendete Grad kann aufgrund dieser Faktoren und des Aktivitätenaufkommens im System niedriger als die

Anzahl der Prozessoren sein. Der Grad der Parallelität kann vor der Abfrageausführung gesenkt werden, wenn das System sehr ausgelastet ist. Der Grund hierfür liegt in der intensiven Nutzung der Systemressourcen durch die partitionsinterne Parallelität, die einerseits die abgelaufene Zeit für eine Abfrage verringert, sich andererseits jedoch auch negativ auf die Leistung für andere Datenbankbenutzer auswirken kann.

Verwenden Sie die SQL-EXPLAIN-Einrichtung zum Anzeigen des Zugriffsplans, wenn Sie Informationen über den vom Optimierungsprogramm ausgewählten Grad der Parallelität erhalten wollen. Zum Anzeigen von Informationen über den tatsächlich bei der Ausführung genutzten Grad der Parallelität verwenden Sie den Datenbanksystemmonitor.

Parallelität in Umgebungen ohne SMP-Maschinen

Sie können einen Grad an Parallelität angeben, ohne eine SMP-Maschine zu haben. Zum Beispiel können ein-/ausgabegebundene Abfragen auf einer Einzelprozessormaschine von der Deklaration eines Grades 2 oder höher profitieren. In diesem Fall braucht der Prozessor vielleicht nicht auf die Beendigung von Eingabe- bzw. Ausgabefunktionen zu warten, bevor er mit der Verarbeitung einer neuen Abfrage beginnt. Die Deklaration eines Grades 2 oder höher steuert die E/A-Parallelität auf einer Einzelprozessormaschine jedoch nicht direkt. Dienstprogramme wie Load können die E/A-Parallelität unabhängig von einer solchen Deklaration steuern. Das Schlüsselwort `ANY` kann ebenfalls dazu verwendet werden, den Konfigurationsparameter *dft_degree* des Datenbankmanagers zu definieren. Das Schlüsselwort `ANY` gibt dem Optimierungsprogramm die Möglichkeit, den Grad der partitionsinternen Parallelität zu bestimmen.

Kapitel 19. Hinweise zu Umgebungen

Auswirkung von Tabellenbereichen auf die Abfrageoptimierung

Bestimmte Merkmale der verwendeten Tabellenbereiche können die Auswahl des Zugriffsplans durch den Abfragecompiler beeinflussen:

- Kenndaten der Container

  Containerkenndaten können sich wesentlich auf den Ein-/Ausgabeaufwand auswirken, der mit der Abfrageausführung verbunden ist. Bei der Auswahl eines Zugriffsplans berücksichtigt das Abfrageoptimierungsprogramm diesen Ein-/Ausgabeaufwand, einschließlich aller Unterschiede in den Aufwänden für die Zugriffe auf Daten verschiedener Tabellenbereiche. Zwei Spalten im Systemkatalog SYSCAT.TABLESPACES werden vom Optimierungsprogramm zur Abschätzung der E/A-Aufwände für den Zugriff auf Daten in einem Tabellenbereich herangezogen:

  – Die Spalte OVERHEAD, die einen Schätzwert in Millisekunden für die Zeit enthält, die der Container benötigt, bevor irgendwelche Daten in den Speicher gelesen werden. In diesen Wert fließen der Aufwand für den E/A-Controller des Containers und die Latenzzeit der Platte, zur der auch die Suchzeit der Platte gehört, mit ein.

    Mithilfe der folgenden Formel lässt sich der Systemaufwand (OVERHEAD) abschätzen:

    ```
    OVERHEAD = durchschnittliche Suchzeit in Millisekunden
               + (0,5 * rotationsbedingte Latenzzeit)
    ```

    Dabei gilt Folgendes:

    - 0,5 stellt den durchschnittlichen Aufwand für eine halbe Umdrehung (Rotation) dar.
    - Die rotationsbedingte Latenzzeit für jede vollständige Umdrehung wird wie folgt in Millisekunden berechnet:

      ```
      (1 / Umdrehung pro Minute) * 60 * 1000
      ```

      Dabei sind die Berechnungsschritte wie folgt:

      • Dividieren durch die Umdrehungen pro Minute, um die Minuten pro Umdrehung zu erhalten
      • Multiplizieren mit 60, um die Sekunden pro Umdrehung zu erhalten
      • Multiplizieren mit 1000, um die Millisekunden pro Umdrehung zu erhalten

    Als Beispiel sei eine Plattengeschwindigkeit von 7.200 Umdrehungen pro Minute angenommen. Mithilfe der Umdrehungs-Latenz-Formel ergäbe sich folgender Wert:

    ```
    (1 / 7200) * 60 * 1000 = 8,328 Millisekunden
    ```

    Das Ergebnis kann anschließend in der Berechnung des Schätzwerts für OVERHEAD bei einer angenommenen durchschnittlichen Suchzeit von 11 Millisekunden verwendet werden:

    ```
    OVERHEAD = 11 + (0,5 * 8,328)
             = 15,164
    ```

Der geschätzte OVERHEAD-Wert läge in diesem Fall bei ca. 15 Millisekunden.

– Die Spalte TRANSFERRATE, die einen Schätzwert in Millisekunden für die Zeit enthält, die zum Einlesen einer Datenseite in den Hauptspeicher benötigt wird.

  Wenn jeder Tabellenbereichscontainer eine einzelne physische Platte ist, können Sie mithilfe der folgenden Formel den Übertragungsaufwand pro Seite in Millisekunden abschätzen:

  ```
  TRANSFERRATE = (1 / spec_rate) * 1000 / 1024000 * Seitengröße
  ```

  Dabei gilt Folgendes:

  - `spec_rate` ist die Übertragungsgeschwindigkeit in MB pro Sekunde, die für die Platte angegeben wird.
  - Dividieren durch `spec_rate`, um die Sekunden pro MB zu erhalten
  - Multiplizieren mit 1000, um die Millisekunden pro MB zu erhalten
  - Dividieren durch 1.024.000 Byte pro MB
  - Multiplizieren mit der Seitengröße in Byte (z. B. 4.096 Byte für eine 4-KB-Seite)

  Als Beispiel sei eine angegebene Übertragungsgeschwindigkeit (spec_rate) von 3 MB pro Sekunde angenommen. Daraus ergäbe sich folgende Berechnung:

  ```
  TRANSFERRATE = (1 / 3) * 1000 / 1024000 * 4096
               = 1,333248
  ```

  Der geschätzte TRANSFERRATE-Wert läge hier bei ca. 1,3 Millisekunden pro Seite.

  Wenn es sich bei den Tabellenbereichscontainern nicht um einzelne physische Platten, sondern um Platteneinheiten (Disk-Arrays, z. B. RAID) handelt, sind zusätzliche Punkte bei der Abschätzung des zu verwendenden Werts für TRANSFERRATE zu beachten. Wenn die Platteneinheit relativ klein ist, können Sie den Wert für `spec_rate` mit der Anzahl Platten multiplizieren, da anzunehmen ist, dass der Engpass auf Plattenebene liegt.

  Ist die Anzahl der Platten in der Einheit, die den Container bildet, jedoch groß, liegt der Engpass vielleicht nicht auf Plattenebene, sondern bei einer der E/A-Subsystemkomponenten wie Einheitencontroller, E/A-Busse oder dem Systembus. In diesem Fall kann nicht angenommen werden, dass der E/A-Durchsatz das Produkt aus `spec_rate` und der Anzahl der Platten ist. Stattdessen muss die tatsächliche E/A-Geschwindigkeit während einer sequenziellen Tabellensuche in MB gemessen werden. Zum Beispiel könnte eine Tabellensuche mit einer Anweisung wie `select count(*) from große_tabelle` durchgeführt werden, die mehrere MB umfasst. Dividieren Sie das Ergebnis durch die Anzahl der Container, die den Tabellenbereich bilden, in dem `große_tabelle` gespeichert ist. Setzen Sie das Ergebnis für `spec_rate` in die oben angegebene Formel ein. Zum Beispiel würde eine gemessene sequenzielle E/A-Geschwindigkeit von 100 MB beim Durchsuchen einer Tabelle in einem Tabellenbereich mit vier Containern einen Wert von 25 MB pro Container bzw. einen TRANSFERRATE-Wert von (1/25) * 1000 / 1024000 * 4096 = 0,16 Millisekunden pro Seite bedeuten.

Jeder der einem Tabellenbereich zugeordneten Container kann sich auf einer anderen physischen Platte befinden. Um die besten Ergebnisse erzielen zu können, sollten alle physischen Platten, die für einen bestimmten Tabellenbereich verwendet werden, über die gleichen Werte für OVERHEAD und TRANSFER-

RATE verfügen. Wenn diese Merkmale nicht übereinstimmen, sollten Sie bei der Einstellung der Werte für OVERHEAD und TRANSFERRATE jeweils den Mittelwert verwenden.

Medienspezifische Werte für diese Spalten können Sie den technischen Daten zur Hardware entnehmen oder durch Experimentieren ermitteln. Diese Werte können in den Anweisungen CREATE TABLESPACE und ALTER TABLESPACE angegeben werden.

Experimentieren wird in der oben erwähnten Umgebung, in der eine Plateneinheit als Container eingesetzt wird, besonders wichtig. Es empfiehlt sich, eine einfache Abfrage, die Daten versetzt, zu erstellen und sie in Verbindung mit einem plattformspezifischen Messprogramm auszuführen. Anschließend können Sie die Abfrage mit anderen Containerkonfigurationen innerhalb des Tabellenbereichs wiederholen. Mithilfe der Anweisungen CREATE und ALTER TABLESPACE können Sie die Art und Weise ändern, wie Daten in der Umgebung übertragen werden.

Die durch diese beiden Werte bereitgestellten Informationen zum Ein-/Ausgabeaufwand können das Optimierungsprogramm in mehrfacher Weise beeinflussen, zum Beispiel dahin gehend, ob ein Index für den Zugriff auf die Daten zu verwenden ist und welche Tabellen als innere und äußere Tabellen in einem Join auszuwählen sind.

- Vorablesezugriff

  Bei der Kalkulation des Ein-/Ausgabeaufwands für den Zugriff auf Daten in einem Tabellenbereich berücksichtigt das Optimierungsprogramm auch die potenziellen Auswirkungen, die das Vorablesen von Daten- und Indexseiten von Platten auf die Leistung der Abfrage haben kann. Der Vorablesezugriff auf Daten- und Indexseiten kann den Aufwand und die Wartezeit verringern, die mit dem Einlesen der Daten in den Pufferpool verbunden sind.

  Das Optimierungsprogramm verwendet die Informationen der Spalten PREFETCHSIZE und EXTENTSIZE im Systemkatalog SYSCAT.TABLESPACES, um die Menge der durch den Vorablesezugriff gelesenen Daten für einen Tabellenbereich abzuschätzen.

  – Der Wert für EXTENTSIZE kann nur bei der Erstellung eines Tabellenbereichs (z. B. mit der Anweisung CREATE TABLESPACE) festgelegt werden. Der Standardwert für EXTENTSIZE beträgt 32 Seiten (von jeweils 4 KB) und ist in der Regel ausreichend.
  – Der Wert für PREFETCHSIZE kann sowohl bei der Erstellung als auch bei der Änderung des Tabellenbereichs mit der Anweisung ALTER TABLESPACE festgelegt werden. Der Standardwert für PREFETCHSIZE wird durch den Wert des Datenbankkonfigurationsparameters DFT_PREFETCH_SZ festgelegt. Lesen Sie die Empfehlungen zur Einstellung dieses Parameters und nehmen Sie Änderungen nach Bedarf vor, um das Versetzen von Daten zu verbessern.

Das folgende Beispiel zeigt die Syntax zur Änderung der Merkmale des Tabellenbereichs RESOURCE:

```
ALTER TABLESPACE RESOURCE
  PREFETCHSIZE 64
  OVERHEAD     19.3
  TRANSFERRATE 0.9
```

Ziehen Sie nach dem Vornehmen von Änderungen an den Tabellenbereichen in Betracht, einen Rebind für Ihre Anwendungen durchzuführen und das Dienstprogramm RUNSTATS zum Erfassen der neuesten Statistikdaten zu Indizes auszuführen, um sicherzustellen, dass die besten Zugriffspläne verwendet werden.

Serveroptionen mit Einfluss auf föderierte Datenbanken

Ein föderiertes System besteht aus einem DB2-Datenbankverwaltungssystem (DBMS), d. h. der föderierten Datenbank, und mindestens einer Datenquelle. Sie geben die Datenquellen für die föderierte Datenbank an, wenn Sie Anweisungen CREATE SERVER absetzen. Beim Absetzen dieser Anweisungen können Sie Serveroptionen angeben, die bestimmte Aspekte des Betriebs des Systems mit der föderierten Datenbank bezüglich DB2 und der angegebenen Datenquelle eingrenzen und steuern. Wenn Sie später Serveroptionen ändern wollen, verwenden Sie Anweisungen ALTER SERVER.

Anmerkung: Sie müssen die Installationsoption für den verteilten Join installieren und den Parameter *federated* des Datenbankmanagers auf den Wert YES setzen, bevor Sie Server erstellen und Serveroptionen angeben können.

Die Werte der Serveroptionen, die Sie angeben, haben Einfluss auf die Pushdown-Analyse von Abfragen, die globale Optimierung und andere Aspekte von Operationen mit föderierten Datenbanken. Zum Beispiel können Sie in der Anweisung CREATE SERVER Leistungsstatistikdaten als Serveroptionswerte angeben, wie zum Beispiel die Option *cpu_ratio*, mit der die relativen Geschwindigkeiten der CPUs an der Datenquelle und auf dem Server der föderierten Datenbank angegeben werden. Sie könnten auch die Option *io_ratio* auf einen Wert setzen, der die relativen Übertragungsgeschwindigkeiten der E/A-Einheiten der Quelle und des Servers der föderierten Datenbank angibt. Bei der Ausführung der Anweisung CREATE SERVER werden diese Daten der Katalogsicht SYSCAT.SERVEROPTIONS hinzugefügt, und das Optimierungsprogramm bezieht sie in die Entwicklung eines Zugriffsplans für die Datenquelle mit ein. Falls sich ein Statistikwert ändert (wie es beispielsweise bei einer Aufrüstung der CPU der Datenquelle möglich ist), können Sie die Katalogsicht SYSCAT.SERVEROPTIONS mithilfe der Anweisung ALTER SERVER mit dieser Änderung aktualisieren. Das Optimierungsprogramm verwendet dann die neuen Informationen beim nächsten Auswählen eines Zugriffsplans für die Datenquelle.

Kapitel 20. Katalogstatistiken

Wenn der Abfragecompiler die Abfragepläne optimiert, werden die Entscheidungen in hohem Maße von statistischen Informationen über die Größe der Tabellen, Indizes und statistischen Sichten der Datenbank beeinflusst. Das Optimierungsprogramm verwendet außerdem Informationen über die Verteilung von Daten in bestimmten Spalten von Tabellen, Indizes und statistischen Sichten, sofern diese Spalten zur Auswahl von Zeilen oder für den Join von Tabellen herangezogen werden. Das Optimierungsprogramm schätzt anhand dieser Informationen den Aufwand für alternative Zugriffspläne für jede Abfrage ab.

Über die Informationen zu Tabellengrößen und Datenverteilungen hinaus können Sie auch statistische Informationen über das Clusterverhältnis (Cluster ratio) von Indizes, die Anzahl von Blattseiten (Leaf pages) in Indizes, die Anzahl von Tabellenzeilen, die aus ihren ursprünglichen Seiten überlaufen, sowie die Anzahl gefüllter und leerer Seiten in einer Tabelle erfassen. Diese Informationen helfen Ihnen bei der Entscheidung, wann eine Reorganisation von Tabellen und Indizes durchzuführen ist.

Wenn Sie Statistikdaten für eine Tabelle in einer Umgebung mit partitionierten Datenbanken sammeln, werden nur Statistikdaten für den Teil der Tabelle gesammelt, der sich in der Datenbankpartition befindet, in der das Dienstprogramm ausgeführt wird; oder es werden nur Statistikdaten für die erste Datenbankpartition in der Datenbankpartitionsgruppe gesammelt, die die Tabelle enthält. Wenn Sie Statistikdaten für eine statistische Sicht erfassen, wenden für alle Datenbankpartitionen Statistikdaten erfasst.

Wenn Sie das Dienstprogramm RUNSTATS für eine Tabelle, eine statistische Sicht oder für eine Tabelle mit zugeordneten Indizes ausführen, werden die folgenden Arten von Statistikinformationen in den Systemkatalogtabellen gespeichert:

Für eine Tabelle und einen Index:

- Die Anzahl der verwendeten Seiten
- Die Anzahl der Seiten, die Zeilen enthalten
- Die Anzahl der Zeilen, die überlaufen
- Die Anzahl von Zeilen in der Tabelle (Kardinalität)
- Für MDC-Tabellen: die Anzahl von Blöcken, die Daten enthalten
- Für partitionierte Tabellen: der Grad der Datenclusterbildung in einer einzigen Datenpartition

Für jede Spalte in der Tabelle oder der statistischen Sicht und für die erste Spalte im Indexschlüssel:

- Die Kardinalität der Spalte
- Die Durchschnittslänge der Spalte
- Der zweithöchste Wert in der Spalte
- Der zweitniedrigste Wert in der Spalte
- Die Anzahl von Nullwerten (NULL) in der Spalte

Für jede XML-Spalte werden die folgenden Statistikdaten erfasst. In jeder Zeile einer XML-Spalte wird ein XML-Dokument gespeichert. Der Knotenzähler eines

angegebenen Pfads oder eines Pfad-Wert-Paars bezieht sich auf die Anzahl an Knoten, die durch den Pfad oder das Pfad-Wert-Paar erreichbar sind. Der Dokumentzähler eines angegebenen Pfads oder eines Pfad-Wert-Paars bezieht sich auf die Anzahl an Dokumenten, die der angegebene Pfad oder das angegebene Pfad-Wert-Paar enthält.

- Die Anzahl von XML-Dokumenten mit dem Wert NULL
- Die Anzahl von XML-Dokumenten mit dem Wert ungleich NULL
- Die Anzahl einzigartiger Pfade
- Die Summe des Knotenzählers für die einzelnen einzigartigen Pfade
- Die Summe des Dokumentzählers für die einzelnen einzigartigen Pfade
- Die *k* Paare aus Pfad und Knotenzähler für den größten Knotenzähler
- Die *k* Paare aus Pfad und Dokumentzähler für den größten Dokumentzähler
- Die *k* Triplets aus Pfad, Wert und Knotenzähler für den größten Knotenzähler
- Die *k* Triplets aus Pfad, Wert und Dokumentzähler für den größten Dokumentzähler
- Für jeden einzigartigen Pfad, der zu einem Text- oder Attributwert führt, gilt Folgendes:
  - Die Anzahl der unterschiedliche Werte dieses Pfads kann folgende Werte annehmen:
  - Den größten Wert
  - Den niedrigsten Wert
  - Die Anzahl an Text- oder Attributknoten
  - Die Anzahl an Dokumenten mit den Text- oder Attributknoten

Für Gruppen von Spalten, die Sie angeben:

- Ein auf einer Zeitmarke basierender Name für die Spaltengruppe
- Die Kardinalität der Spaltengruppe

Nur für Indizes:

- Die Anzahl von Indexeinträgen (Indexkardinalität)
- Die Anzahl von Blattseiten (leaf pages)
- Die Anzahl von Indexstufen
- Der Grad der Clusterbildung der Tabellendaten in Bezug auf den Index
- Der Grad der Clusterbildung der Indexschlüssel in Bezug auf die Datenpartitionen
- Das Verhältnis von Blattseiten auf dem Datenträger in der Reihenfolge des Indexschlüssels zur Anzahl von Seiten in dem Bereich von Seiten, die durch den Index belegt werden
- Die Anzahl unterschiedlicher Werte in der Spalte des Index
- Die Anzahl unterschiedlicher Werte in den ersten zwei, drei und vier Spalten des Index
- Die Anzahl unterschiedlicher Werte in allen Spalten des Index
- Die Anzahl von Blattseiten, die sich auf dem Datenträger in der Reihenfolge des Indexschlüssels mit wenigen oder keinen dazwischen liegenden Lücken befinden
- Die Anzahl von Seiten, in denen alle Satz-IDs (RIDs) als gelöscht markiert sind
- Die Anzahl als gelöscht markierter Satz-IDs in Seiten, in denen nicht alle Satz-IDs als gelöscht markiert sind

Wenn Sie detaillierte Statistiken für einen Index anfordern, speichern Sie auch feinere Informationen über den Grad der Clusterbildung der Tabelle in Bezug auf den Index sowie die Seitenabrufschätzwerte für verschiedene Puffergrößen.

Darüber hinaus können Sie die folgenden Arten von Statistiken über Tabellen und Indizes erfassen:

- Statistiken zur Datenverteilung

  Das Optimierungsprogramm nutzt die Statistiken über die Datenverteilung zur Abschätzung effizienter Zugriffspläne für Tabellen und statistische Sichten, in denen Daten nicht gleichmäßig verteilt sind und Spalten eine beträchtliche Anzahl mehrfach auftretender Werte aufweisen.
- Detaillierte Statistiken zu Indizes

  Das Optimierungsprogramm nutzt detaillierte Indexstatistiken zur Ermittlung, wie effizient der Zugriff auf eine Tabelle über einen Index realisiert werden kann.
- Statistiken zu Unterelementen

  Das Optimierungsprogramm nutzt Unterelementstatistiken für LIKE-Vergleichselemente, insbesondere für solche, die nach einem in eine Zeichenfolge eingebetteten Muster suchen, wie zum Beispiel `LIKE %disk%`.

Verteilungsstatistiken werden in folgenden Fällen nicht erfasst:

- Wenn die Konfigurationsparameter *num_freqvalues* und *num_quantiles* auf den Wert null (0) gesetzt sind.
- Wenn die Verteilung von Daten bekannt ist, zum Beispiel wenn jeder Datenwert nur einmal vorkommt.
- Wenn die Spalte einen Datentyp besitzt, für den keine Statistikdaten erfasst werden. Solche Datentypen sind LONG, LOB oder strukturierte Spalten.
- Wenn es sich um Zeilentypen in untergeordneten Tabellen handelt, werden die Statistiken für NPAGES, FPAGES und OVERFLOW der Tabellenebene nicht erfasst.
- Wenn Quantilverteilungsdaten angefordert werden, jedoch nur ein Nichtnullwert in der Spalte vorhanden ist.
- Wenn es sich um erweiterte Indizes oder deklarierte temporäre Tabellen handelt.

Anmerkung: Sie können das Dienstprogramm RUNSTATS für eine deklarierte temporäre Tabelle ausführen. Die Ergebnisstatistikdaten werden jedoch nicht in den Systemkatalogen gespeichert, weil deklarierte Tabellen keine Katalogeinträge besitzen. Allerdings werden die Statistikdaten in Speicherstrukturen abgelegt, welche die Kataloginformationen für deklarierte temporäre Tabellen darstellen. In bestimmten Fällen kann eine Ausführung von RUNSTATS für diese Tabellen daher sinnvoll sein.

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 bestimmen 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 Statistiken für eine Tabelle manuell geändert wurden, wird angenommen, dass der Benutzer die Statistiken für die Tabelle pflegt, sodass der Datenbankmanager keine Statistiken für diese Tabelle pflegt. Wenn der Datenbankmanager die Statistikdaten für eine Tabelle pflegen soll, deren Statistikdaten manuell aktualisiert wurden, führen Sie den Befehl RUNSTATS für die Tabelle aus. Für migrierte Tabellen, deren Statistikdaten zuvor manuell aktualisiert wurden, werden die Statistikdaten automatisch durch den Datenbankmanager gepflegt.

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. Wenn eine Tabelle in mehreren Datenbankpartitionen enthalten ist und die Tabelle keine Statistiken hat, erfasst der Datenbankmanager Statistiken (synchrone und asynchrone) immer in der ersten Datenbankpartition der Datenbankpartitionsgruppe. Wenn für die Tabelle bereits Statistiken vorhanden sind, werden die Statistiken in der Datenbank-

partition erfasst, in der die Statistiken zuvor zuletzt erfasst wurden. Dies stellt die Konsistenz der Statistikdaten sicher, da sie immer in derselben Datenbankpartition erfasst werden.

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 52. 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.

Bei der automatischen Erstellung von Statistikdaten und der Profilermittlung verwendeter Speicher

Die automatischen Funktionen zur Statistikerfassung und Reorganisation speichern Arbeitsdaten in Tabellen Ihrer Datenbank. Diese Tabellen werden im Tabellenbereich SYSTOOLSPACE erstellt. Der Tabellenbereich SYSTOOLSPACE wird automatisch mit Standardoptionen erstellt, wenn die Datenbank aktiviert wird. Der Speicherbedarf für diese Tabellen ist proportional zur Anzahl der Tabellen in der Datenbank und sollte mit ca. 1 KB pro Tabelle veranschlagt werden. Wenn dies für Ihre Datenbank eine beträchtliche Größe ist, kann es sinnvoll sein, den Tabellenbereich selbst zu löschen und erneut zu erstellen, um eine angemessene Speicherkapazität zuzuordnen. Die Tabellen der automatischen Verwaltung und des Diagnosemonitors in diesem Tabellenbereich werden automatisch erneut erstellt. Alle erfassten älteren Daten in diesen Tabellen gehen verloren, wenn der Tabellenbereich gelöscht wird.

Aktivitätsprotokollierung für die automatische Statistikerfassung

Zur Ermittlung der Aktivitäten der automatischen Statistikerfassung, die für eine Datenbank ausgeführt wurden, wird mit DB2 Version 9.5 ein Statistikprotokoll eingeführt. Das Statistikprotokoll zeichnet alle Aktivitäten zur Statistikerfassung für eine Datenbank auf. Dies gilt sowohl für automatische als auch manuelle Aktivitäten.

Der Standardname des Statistikprotokolls lautet db2optstats.*nummer*.log. Es befindet sich im Verzeichnis $DIAGPATH/events. Das Statistikprotokoll ist ein Umlaufprotokoll. Das Verhalten des Statistikprotokolls wird durch die Registrierdatenbankvariable **DB2_OPTSTATS_LOG** gesteuert.

Der Inhalt des Statistikprotokolls kann direkt angezeigt oder mithilfe der Tabellenfunktion SYSPROC.PD_GET_DIAG_HIST abgefragt werden.

Die Tabellenfunktion SYSPROC.PD_GET_DIAG_HIST gibt eine Reihe von Spalten zurück, die Standardinformationen zu jedem protokollierten Ereignis enthalten. Dies sind zum Beispiel die Zeitmarke, der DB2-Instanzname, der Datenbankname, die Prozess-ID, der Prozessname und die Thread-ID. Das Protokoll enthält außerdem generische Spalten zur Verwendung durch verschiedene Protokolleinrichtungen. In der folgenden Tabelle werden die generischen Spalten und ihre Verwendung im Statistikprotokoll beschrieben.

Tabelle 53. Generische Spalten in der Statistikprotokolldatei

Spaltenname	Datentyp	Beschreibung
OBJTYPE	VARCHAR(64)	Der Typ von Objekt, auf den sich das Ereignis bezieht. Bei der Statistikprotokollierung ist dies der Typ von Statistik, der zu erfassen ist. Der Typ kann sich auch auf einen Hintergrundprozess zur Statistikerfassung beziehen, wenn er gestartet oder gestoppt wird. Ferner kann er sich auf Aktivitäten beziehen, die von automatischen Statistikerfassungen ausgeführt werden, zum Beispiel Aktivitäten zu Stichprobentests, Anfangsstichproben und Tabellenbewertung. Die folgenden Werte sind für Aktionen der Statistikerfassung möglich: **TABLE STATS** Tabellenstatistiken werden erfasst. **INDEX STATS** Indexstatistiken werden erfasst. **TABLE AND INDEX STATS** Sowohl Tabellen- als auch Indexstatistiken werden erfasst.

Tabelle 53. Generische Spalten in der Statistikprotokolldatei (Forts.)

Spaltenname	Datentyp	Beschreibung
OBJTYPE (Fortsetzung)	VARCHAR(64)	Die folgenden Werte sind für automatische Statistikoperationen verfügbar: **EVALUATION** Der automatische, im Hintergrund ausgeführte Statistikerfassungsprozess hat mit der Bewertungsphase begonnen. In der Bewertungsphase werden Tabellen überprüft, um festzustellen, ob für sie Statistikdaten erforderlich sind. Ist dies der Fall, werden anschließend Statistikdaten erfasst. **INITIAL SAMPLING** Statistikdaten werden für eine Tabelle durch Stichprobenentnahme erfasst. Die Stichprobenstatistiken werden in den Systemkatalogen gespeichert. Dies ermöglicht der automatischen Statistikerfassung eine rasche Erfassung von Statistikdaten für eine Tabelle, die keine Statistiken hat. Nachfolgende Statistikerfassungen erfassen Statistikdaten ohne Stichprobenentnahme. Die anfängliche Stichprobenentnahme wird während der Bewertungsphase der automatischen Statistikerfassung ausgeführt. **SAMPLING TEST** Statistikdaten werden für eine Tabelle durch Stichprobenentnahme erfasst. Die Stichprobenstatistiken werden nicht in den Systemkatalogen gespeichert. Die Stichprobenstatistiken werden mit den aktuellen Katalogstatistiken verglichen, um festzustellen, ob vollständige Statistikdaten für die betroffene Tabelle gesammelt werden sollen und wann solche Statistiken gesammelt werden sollen. Der Stichprobentest wird während der Bewertungsphase der automatischen Statistikerfassung durchgeführt. **STATS DAEMON** Der Statistikdämon ist ein Hintergrundprozess zur Verarbeitung von Anforderungen, die durch die Echtzeitstatistikerfassung übergeben werden. Dieser Objekttyp wird protokolliert, wenn der Hintergrundprozess gestartet und gestoppt wird.
OBJNAME	VARCHAR(255)	Der Name des Objekts, für das das Ereignis gilt, falls verfügbar. Für das Statistikprotokoll enthält diese Spalte den Tabellen- oder Indexnamen. Wenn OBJTYPE den Wert STATS DAEMON oder EVALUATION enthält, ist OBJNAME der Datenbankname und OBJNAME_QUALIFIER hat den Wert NULL.

Tabelle 53. Generische Spalten in der Statistikprotokolldatei (Forts.)

Spaltenname	Datentyp	Beschreibung
OBJNAME_QUALIFIER	VARCHAR(255)	Für das Statistikprotokoll enthält diese Spalte das Tabellen- oder Indexschema.
EVENTTYPE	VARCHAR(24)	Der Ereignistyp ist die Aktion bzw. das Verb, das diesem Ereignis zugeordnet ist. Die folgenden Werte für die Statistikprotokollierung sind möglich: **COLLECT** Diese Aktion wird für eine Operation zur Statistikerfassung protokolliert. **START** Diese Aktion wird protokolliert, wenn der Hintergrundprozess der Echtzeitstatistikerfassung (OBJTYPE = STATS DAEMON) oder eine Bewertungsphase der automatischen Statistikerfassung (OBJTYPE = EVALUATION) gestartet wird. **STOP** Diese Aktion wird protokolliert, wenn der Hintergrundprozess der Echtzeitstatistikerfassung (OBJTYPE = STATS DAEMON) oder eine Bewertungsphase der automatischen Statistikerfassung (OBJTYPE = EVALUATION) gestoppt wird. **ACCESS** Es wurde versucht, auf eine Tabelle zu Zwecken der Statistikerfassung zuzugreifen. Dieser Ereignistyp dient zur Protokollierung eines fehlgeschlagenen Zugriffs, wenn das Objekt nicht verfügbar ist.
FIRST_EVENTQUALIFIERTYPE	VARCHAR(64)	Der Typ des ersten Ereignisqualifikationsmerkmals. Ereignisqualifikationsmerkmale dienen zur Beschreibung der von dem Ereignis betroffenen Bereiche. Für das Statistikprotokoll ist das erste Ereignisqualifikationsmerkmal die Zeitmarke für den Zeitpunkt, zu dem das Ereignis aufgetreten ist. Der Wert für den Typ des ersten Ereignisqualifikationsmerkmals ist AT.
FIRST_EVENTQUALIFIER	CLOB(16K)	Das erste Qualifikationsmerkmal für das Ereignis. Für die Statistikprotokollierung ist das erste Ereignisqualifikationsmerkmal die Zeitmarke für den Zeitpunkt, zu dem das Statistikereignis aufgetreten ist. Die Zeitmarke des Statistikereignisses kann sich von der Zeitmarke des Protokollsatzes, wie er in der Spalte TIMESTAMP dargestellt wird, unterscheiden.
SECOND_EVENTQUALIFIERTYPE	VARCHAR(64)	Der Typ des zweiten Ereignisqualifikationsmerkmals. Für die Statistikprotokollierung kann dieser Wert BY oder NULL sein. Dieses Feld wird für andere Ereignistypen nicht verwendet.

Tabelle 53. Generische Spalten in der Statistikprotokolldatei (Forts.)

Spaltenname	Datentyp	Beschreibung
SECOND_EVENTQUALIFIER	CLOB(16K)	Das zweite Qualifikationsmerkmal für das Ereignis. Bei der Statistikprotokollierung stellt diese Spalte für COLLECT-Ereignistypen dar, wie die Statistiken erfasst wurden. Die folgenden Werte sind möglich: **User** Die Statistikerfassung wurde durch einen DB2-Benutzer mithilfe des Befehls RUNSTATS, LOAD, CREATE INDEX oder REDISTRIBUTE ausgeführt. **Synchronous** Die Statistikerfassung wurde bei der SQL-Anweisungskompilierung durch DB2 ausgeführt. Die Statistiken sind im Statistikcache, jedoch nicht in den Systemkatalogen gespeichert. **Synchronous sampled** Die Statistikerfassung wurde durch Stichprobenentnahme bei der SQL-Anweisungskompilierung durch DB2 ausgeführt. Die Statistiken sind im Statistikcache, jedoch nicht in den Systemkatalogen gespeichert. **Fabricate** Die Statistikdaten wurden bei der SQL-Anweisungskompilierung mithilfe von Informationen konstruiert, die vom Daten- und Indexmanager gepflegt werden. Die Statistiken sind im Statistikcache, jedoch nicht in den Systemkatalogen gespeichert. **Fabricate partial** Nur einige Statistikdaten wurden bei der SQL-Anweisungskompilierung mithilfe von Informationen konstruiert, die vom Daten- und Indexmanager gepflegt werden. Insbesondere wurden nur die HIGH2KEY- und LOW2KEY-Statistiken für bestimmte Spalten konstruiert. Die Statistiken sind im Statistikcache, jedoch nicht in den Systemkatalogen gespeichert. **Asynchronous** Die Statistiken wurden durch einen Hintergrundprozess von DB2 erfasst und in den Systemkatalogen gespeichert. Dieses Feld wird für andere Ereignistypen nicht verwendet.
THIRD_EVENTQUALIFIERTYPE	VARCHAR(64)	Der Typ des dritten Ereignisqualifikationsmerkmals. Für die Statistikprotokollierung kann dieser Wert DUE TO oder NULL sein.

Tabelle 53. Generische Spalten in der Statistikprotokolldatei (Forts.)

Spaltenname	Datentyp	Beschreibung
THIRD_EVENTQUALIFIER	CLOB(16K)	Das dritte Qualifikationsmerkmal für das Ereignis. Für die Statistikprotokollierung stellt diese Spalte die Ursache dar, aus der die Statistikaktivität nicht abgeschlossen werden konnte. Die folgenden Werte sind möglich: **Timeout** Die synchrone Statistikerfassung hat das Zeitbudget überschritten. **Error** Die Statistikaktivität ist aufgrund eines Fehlers fehlgeschlagen. **RUNSTATS error** Die synchrone Statistikerfassung ist aufgrund eines RUNSTATS-Fehlers fehlgeschlagen. Bei einigen Fehlern wurde die SQL-Anweisungskompilierung vielleicht erfolgreich abgeschlossen, auch wenn die Statistikdaten nicht erfasst werden konnten. Wenn zum Beispiel nicht genügend Speicherplatz für die Statistikerfassung verfügbar ist, wird die SQL-Anweisungskompilierung fortgesetzt. **Object unavailable** Die Statistiken konnten für das Datenbankobjekt nicht erfasst werden, weil kein Zugriff auf das Objekt möglich war. Einige mögliche Ursachen: • Das Objekt ist im Z-Modus (Superexclusive) gesperrt. • Der Tabellenbereich, in dem sich das Objekt befindet, ist nicht verfügbar. • Die Tabellenindizes müssen erneut erstellt werden. **Conflict** Die synchrone Statistikerfassung wurde nicht ausgeführt, weil eine andere Anwendung bereits synchrone Statistiken erfasste. Überprüfen Sie die Spalte FULLREC oder die Datei db2diag.log auf Details zu diesem Fehler.
EVENTSTATE	VARCHAR(255)	Der Status des Objekts oder der Aktion infolge des Ereignisses. Für die Statistikprotokollierung gibt diese Spalte den Status der Statistikoperation an. Mögliche Werte: • start • success • failure

Beispiel

In diesem Beispiel gibt die Abfrage Protokollsätze für Ereignisse bis zu einem Jahr vor der aktuellen Zeitmarke zurück, indem sie die Tabellenfunktion PD_GET_DIAG_HIST aufruft.

```
SELECT
pid,
tid,
substr(eventtype, 1,10),
substr(objtype,1,30) as objtype,
substr(objname_qualifier,1,20) as objschema,
substr(objname,1,10) as objname,
substr(first_eventqualifier,1,26) as event1,
substr(second_eventqualifiertype,1,2) as event2_type,
substr(second_eventqualifier,1,20) event2,
substr(third_eventqualifiertype,1,6) event3_type,
substr(third_eventqualifier,1,15) event3,
substr(eventstate,1,20) as eventstate
FROM
TABLE( SYSPROC.PD_GET_DIAG_HIST
       ( 'optstats', 'EX', 'NONE',
          CURRENT_TIMESTAMP - 1 year, CAST( NULL AS TIMESTAMP ))) as sl
order by timestamp(varchar(substr(first_eventqualifier,1,26),26))
;
```

Die Ergebnisse werden nach der Zeitmarke in der Spalte FIRST_EVENTQUALIFIER geordnet, die den Zeitpunkt des Statistikereignisses angibt.

```
PID   TID                 EVENTTYPE  OBJTYPE                        OBJSCHEMA           OBJNAME    EVENT1                     EVENT2_TYPE EVENT2              EVENT3_TYPE EVENT3         EVENTSTATE
----- ------------------- ---------- ------------------------------ ------------------- ---------- -------------------------- ----------- ------------------- ----------- -------------- --------------------
28399          1082145120 START      STATS DAEMON                   -                   PROD_DB    2007-07-09-18.37.40.398905 -           -                   -           -              success
28389        183182027104 COLLECT    TABLE AND INDEX STATS          DB2USER             DISTRICT   2007-07-09-18.37.43.261222 BY          Synchronous         -           -              start
28389        183182027104 COLLECT    TABLE AND INDEX STATS          DB2USER             DISTRICT   2007-07-09-18.37.43.407447 BY          Synchronous         -           -              success
28399          1082145120 COLLECT    TABLE AND INDEX STATS          DB2USER             CUSTOMER   2007-07-09-18.37.43.471614 BY          Asynchronous        -           -              start
28399          1082145120 COLLECT    TABLE AND INDEX STATS          DB2USER             CUSTOMER   2007-07-09-18.37.43.524496 BY          Asynchronous        -           -              success
28399          1082145120 STOP       STATS DAEMON                   -                   PROD_DB    2007-07-09-18.37.43.526212 -           -                   -           -              success
28389        183278496096 COLLECT    TABLE STATS                    DB2USER             ORDER_LINE 2007-07-09-18.37.48.676524 BY          Synchronous sampled -           -              start
28389        183278496096 COLLECT    TABLE STATS                    DB2USER             ORDER_LINE 2007-07-09-18.37.53.677546 BY          Synchronous sampled DUE TO      Timeout        failure
28389          1772561034 START      EVALUATION                     -                   PROD_DB    2007-07-10-12.36.11.092739 -           -                   -           -              success
28389          8231991291 COLLECT    TABLE AND INDEX STATS          DB2USER             DISTRICT   2007-07-10-12.36.30.737603 BY          Asynchronous        -           -              start
28389          8231991291 COLLECT    TABLE AND INDEX STATS          DB2USER             DISTRICT   2007-07-10-12.36.34.029756 BY          Asynchronous        -           -              success
28389          1772561034 STOP       EVALUATION                     -                   PROD_DB    2007-07-10-12.36.39.685188 -           -                   -           -              success
```

Verbessern der Abfrageleistung für große Statistikprotokolle

Wenn die Statistikprotokolldateien groß sind, können Sie die Abfrageleistung verbessern, indem Sie die Protokollsätze in eine Tabelle kopieren, Indizes erstellen und anschließend Statistiken erfassen.

Vorgehensweise

1. Erstellen Sie eine Tabelle mit den gewünschten Spalten zur Aufnahme der Protokollsätze.

```
create table db2user.stats_log
(pid       bigint,
 tid       bigint,
 timestamp timestamp,
 dbname    varchar(128),
 retcode   integer,
 eventtype varchar(24),
 objtype   varchar(30),
 objschema varchar(20),
 objname   varchar(30),
 event1_type varchar(20),
 event1      timestamp,
 event2_type varchar(20),
 event2      varchar(40),
 event3_type varchar(20),
 event3      varchar(40),
 eventstate  varchar(20)
);
```

2. Deklarieren Sie einen Cursor für eine Abfrage über SYSPROC.PD_GET_DIAG-_HIST.

```
declare c1 cursor for
SELECT
pid,
tid,
timestamp,
dbname,
retcode,
eventtype,
substr(objtype,1,30) as objtype,
substr(objname_qualifier,1,20) as objschema,
substr(objname,1,30) as objname,
substr(first_eventqualifiertype,1,20),
substr(first_eventqualifier,1,26),
substr(second_eventqualifiertype,1,20),
substr(second_eventqualifier,1,40),
substr(third_eventqualifiertype,1,20),
substr(third_eventqualifier,1,40),
substr(eventstate,1,20)
FROM
TABLE( SYSPROC.PD_GET_DIAG_HIST
       ( 'optstats', 'EX', 'NONE',
          CURRENT_TIMESTAMP - 1 year, CAST( NULL AS TIMESTAMP ))) as sl
;
```

3. Laden Sie mithilfe des Befehls LOAD die Statistikprotokollsätze aus der Cursorfunktion in die Tabelle.

```
load from c1 of cursor replace into db2user.stats_log;
```

4. Erstellen Sie Indizes, und erfassen Sie Statistiken zu der Tabelle.

```
create index sl_ix1 on db2user.stats_log(eventtype, event1);
create index sl_ix2 on db2user.stats_log(objtype, event1);
create index sl_ix3 on db2user.stats_log(objname);

runstats on table db2user.stats_log with distribution and sampled detailed indexes all;
```

Richtlinien für die Erfassung und Aktualisierung von Statistiken

Der Befehl RUNSTATS erfasst Statistikdaten für Tabellen, Indizes und statistische Sichten, um genaue Informationen bereitzustellen, die das Optimierungsprogramm zur Auswahl von Zugriffsplänen auswerten kann.

Verwenden Sie das Dienstprogramm RUNSTATS zur Erfassung von Statistiken in folgenden Situationen:

- Wenn Daten in eine Tabelle geladen und die geeigneten Indizes erstellt wurden.
- Wenn Sie einen neuen Index für eine Tabelle erstellen. Sie müssen das Dienstprogramm RUNSTATS nur für den neuen Index ausführen, wenn die Tabelle seit der letzten Ausführung von RUNSTATS für sie nicht modifiziert wurde.
- Wenn eine Tabelle mit dem Dienstprogramm REORG reorganisiert wurde.
- Wenn die Tabelle und ihre Indizes durch Änderungen, Löschungen und Einfügungen von Daten umfangreich aktualisiert wurden. („Umfangreich" heißt in diesem Fall, dass 10 bis 20 Prozent der Tabellen- und Indexdaten von den Änderungen betroffen sind.)
- Vor dem Binden von Anwendungsprogrammen, deren Leistung von kritischer Bedeutung ist.
- Wenn Sie aktuelle und frühere Statistiken vergleichen wollen. Wenn Sie Statistiken in regelmäßigen Abständen aktualisieren, können Sie Leistungsprobleme frühzeitig erkennen.

- Wenn die Menge der vorab gelesenen Daten (PREFETCHSIZE) geändert wird.
- Wenn das Dienstprogramm REDISTRIBUTE DATABASE PARTITION GROUP zur Umverteilung der Daten verwendet wurde.

  Anmerkung: In früheren Versionen von DB2 arbeitete dieser Befehl mit dem Schlüsselwort NODEGROUP anstelle der Schlüsselwörter DATABASE PARTITION GROUP.
- Verwenden Sie das Dienstprogramm RUNSTATS zur Erfassung von Statistiken zu XML-Spalten: Wenn RUNSTATS nur zum Sammeln von Statistikdaten für XML-Spalten verwendet wird, werden Statistikdaten für nicht-XML-Spalten, die mit LOAD oder durch eine frühere Ausführung des Dienstprogramms RUNSTATS gesammelt wurden, behalten. In dem Fall, in dem Statistikdaten für einige XML-Spalten zuvor gesammelt wurden, werden diese Statistikdaten entweder gelöscht, falls keine Statistikdaten für diese XML-Spalte über den aktuellen Befehl gesammelt werden, oder sie werden ersetzt, falls Statistikdaten für diese XML-Spalte mithilfe des aktuellen Befehls gesammelt werden.

Zur Verbesserung der Leistung von RUNSTATS und zur Einsparung von Plattenspeicherplatz für Statistiken sollten Sie in Betracht ziehen, nur die Spalten anzugeben, für die Datenverteilungsstatistiken erfasst werden sollten.

Im Idealfall sollte für Anwendungsprogramme nach der Ausführung von RUNSTATS ein Rebind durchgeführt werden. Das Abfrageoptimierungsprogramm könnte aufgrund der neuen Statistiken einen anderen Zugriffsplan auszuwählen.

Wenn Sie nicht über genügend Zeit verfügen, alle gewünschten Statistikdaten auf einmal zu erfassen, bietet es sich an, RUNSTATS jeweils nur zur Aktualisierung einiger weniger Tabellen, Indizes oder statistischen Sichten nach dem Rotationsprinzip für eine Gruppe von Objekten auszuführen. Wenn aufgrund der Aktivitäten an den Tabellen zwischen den Zeitpunkten, zu denen Sie RUNSTATS jeweils zu einer Teilaktualisierung ausführen, Inkonsistenzen auftreten, wird während der Abfrageoptimierung eine Warnung (SQL0437W, Ursachencode 6) ausgegeben. Beispiel: Zu dieser Situation kann es kommen, wenn Sie RUNSTATS ausführen, um Statistikdaten zur Tabellenverteilung zu erfassen und RUNSTATS, nach einer Tabellenaktivität, erneut ausführen, um Indexstatistikdaten zu dieser Tabelle zu sammeln. Wenn Inkonsistenzen nach der Aktivität an der Tabelle auftreten und während der Abfrageoptimierung erkannt werden, wird die Warnung ausgegeben. Wenn dieser Fall eintritt, sollten Sie RUNSTATS erneut ausführen, um die Verteilungsstatistikdaten zu aktualisieren.

Um sicherzustellen, dass die Indexstatistikdaten mit den Tabellenstatistikdaten synchron sind, sollten Sie RUNSTATS so ausführen, dass Tabellen- und Indexstatistikdaten zu gleicher Zeit erfasst werden. Indexstatistikdaten behalten die Mehrheit der bei der letzten Ausführung von RUNSTATS erfassten Tabellen- und Spaltenstatistikdaten bei. Wenn seit der letzten Erfassung von Tabellenstatistikdaten umfangreiche Änderungen an der Tabelle vorgenommen wurden, geht durch die Erfassung nur der Indexstatistikdaten für diese Tabelle die Synchronisierung der beiden Arten von Statistikdaten auf allen Knoten verloren.

Das Aufrufen des Dienstprogramms RUNSTATS auf einem Produktionssystem kann sich negativ auf die Leistung der Auslastung im Produktionsbetrieb auswirken. Das Dienstprogramm RUNSTATS unterstützt jetzt eine Drosselungsoption, die zur Begrenzung der Leistungsbeeinträchtigung durch die Ausführung von RUNSTATS bei hohem Aufkommen an Datenbankaktivitäten eingesetzt werden kann.

Wenn Sie Statistikdaten für eine Tabelle in einer Umgebung mit partitionierten Datenbanken sammeln möchten, erfasst das Dienstprogramm RUNSTATS nur Statistikdaten für Tabellen in der Datenbankpartition, in der Sie es ausführen. Die RUNSTATS-Ergebnisse aus dieser Datenbankpartition werden für die anderen Datenbankpartitionen extrapoliert. Wenn die Datenbankpartition, in der Sie das Dienstprogramm RUNSTATS ausführen, keinen Teil einer bestimmten Tabelle enthält, wird die Anforderung an die erste Datenbankpartition in der Datenbankpartitionsgruppe gesendet, die einen Teil der Tabelle enthält.

Wenn Sie Statistikdaten für eine statistische Sicht erfassen, werden für alle Datenbankpartitionen Statistikdaten erfasst, die Basistabellen enthalten, auf die die Sicht verweist.

Beachten Sie die folgenden Hinweise zur Verbesserung der Effizienz von RUNSTATS und der Nützlichkeit der erfassten Statistikdaten:

- Erfassen Sie Statistikdaten nur für die Spalten, die für den Join von Tabellen bzw. in Klauseln WHERE, GROUP BY und ähnlichen von Abfragen verwendet werden. Wenn diese Spalten indexiert sind, können Sie die Spalten mit der Klausel ONLY ON KEY COLUMNS des Befehls RUNSTATS angeben.
- Passen Sie die Werte für die Parameter *num_freqvalues* und *num_quantiles* für bestimmte Tabellen und bestimmte Spalten in Tabellen an.
- Erfassen Sie detaillierte Indexstatistikdaten mit der Klausel SAMPLE DETAILED, um den Umfang der im Hintergrund für detaillierte Indexstatistiken durchgeführten Berechnungen zu verringern. Die Klausel SAMPLE DETAILED verkürzt die Zeit, die zur Erfassung von Statistikdaten benötigt wird, und liefert in den meisten Fällen eine angemessene Genauigkeit.
- Wenn Sie einen Index für eine mit Daten gefüllte Tabelle erstellen, fügen Sie die Klausel COLLECT STATISTICS hinzu, um die Statistikdaten bei der Erstellung des Index zu erfassen.
- Wenn Tabellenzeilen in größerem Umfang hinzugefügt oder gelöscht werden oder wenn Daten in Spalten, für die Statistiken erfasst werden, aktualisiert werden, sollten Sie RUNSTATS erneut ausführen, um die Statistiken zu aktualisieren.
- Da RUNSTATS nur Statistikdaten in einer einzigen Datenbankpartition sammelt, sind die Statistikdaten weniger genau, wenn die Daten nicht gleichmäßig über alle Datenbankpartitionen verteilt sind. Wenn Sie eine ungleichmäßige Datenverteilung vermuten, kann es sinnvoll sein, die Daten vor der Ausführung von RUNSTATS mithilfe des Befehls REDISTRIBUTE DATABASE PARTITION GROUP erneut auf Datenbankpartitionen zu verteilen.

Erfassen von Katalogstatistiken

Die Erfassung von Katalogstatistiken über Tabellen, Indizes und statistische Sichten dient der Bereitstellung von Informationen, die das Optimierungsprogramm zur Auswahl des besten Zugriffsplans für Abfragen verwendet.

Für Tabellen und Indizes müssen Sie über eine der folgenden Berechtigungen verfügen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL
- Berechtigung SYSMAINT
- Berechtigung DBADM
- Zugriffsrecht CONTROL für die Tabelle

- Berechtigung LOAD

Für statistische Sichten müssen Sie über eine der folgenden Berechtigungen verfügen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL
- Berechtigung SYSMAINT
- Berechtigung DBADM
- Zugriffsrecht CONTROL für die Tabelle

Außerdem müssen Sie über Zugriffsrechte zum Zugreifen auf Zeilen der statistischen Sicht verfügen. Sie müssen insbesondere für die einzelnen Tabellen, Sichten oder Kurznamen, auf die in der Definition der statistischen Sicht verwiesen wird, über eines der folgenden Zugriffsrechte verfügen:

- Berechtigung SYSADM oder DBADM
- Zugriffsrecht CONTROL
- Zugriffsrecht SELECT

Gehen Sie wie folgt vor, um Katalogstatistiken zu erfassen:

1. Stellen Sie eine Verbindung zu der Datenbank her, in der die Tabellen, Indizes und statistischen Sichten enthalten sind, für die Sie statistischen Informationen erfassen möchten.
2. Führen Sie über die DB2-Befehlszeile den Befehl RUNSTATS mit den gewünschten Optionen aus. Über diese Optionen können Sie die Statistikdaten anpassen, die für Abfragen erfasst werden, die für Tabellen, Indizes und statistische Sichten ausgeführt werden.
3. Setzen Sie im Anschluss an die Ausführung von RUNSTATS eine Anweisung COMMIT ab, um die Sperren freizugeben.
4. Führen Sie für die Pakete einen Rebind durch, die auf Tabellen, Indizes und statistische Sichten zugreifen, für die Sie statistische Informationen neu generiert haben.

Verwenden Sie die Steuerzentrale, wenn Sie eine grafische Benutzerschnittstelle zur Angabe von Optionen und zur Erfassung von Statistiken nutzen wollen.

Anmerkung:

1. Da das Dienstprogramm RUNSTATS keine Verwendung von Kurznamen unterstützt, gehen Sie bei der Aktualisierung von Statistiken für Abfragen in föderierten Datenbanken anders vor. Wenn Abfragen auf eine föderierte Datenbank zugreifen, führen Sie RUNSTATS für die Tabellen in allen Datenbanken aus, löschen die Kurznamen, über die auf ferne Tabellen zugegriffen wird, und erstellen diese Kurznamen erneut, um die neuen Statistiken für das Optimierungsprogramm verfügbar zu machen.
2. Wenn Sie Statistikdaten für eine Tabelle in einer Umgebung mit partitionierten Datenbanken sammeln möchten, erfasst das Dienstprogramm RUNSTATS nur Statistikdaten für Tabellen in der Datenbankpartition, in der Sie es ausführen. Die RUNSTATS-Ergebnisse aus dieser Datenbankpartition werden für die anderen Datenbankpartitionen extrapoliert. Wenn die Datenbankpartition, in der Sie das Dienstprogramm RUNSTATS ausführen, keinen Teil einer bestimmten Tabelle enthält, wird die Anforderung an die erste Datenbankpartition in der Datenbankpartitionsgruppe gesendet, die einen Teil der Tabelle enthält.

Wenn Sie Statistikdaten für eine statistische Sicht erfassen, werden für alle Datenbankpartitionen Statistikdaten erfasst, die Basistabellen enthalten, auf die die Sicht verweist.

Erfassen von Verteilungsstatistiken für bestimmte Spalten

Zur Gewährleistung der Effizienz sowohl des Dienstprogramms RUNSTATS als auch der nachfolgenden Abfrageplananalyse ist es sinnvoll, Verteilungsstatistikdaten nur für die Spalten zu erfassen, die von Abfragen in Klauseln wie WHERE, GROUP BY und ähnlichen verwendet werden. Darüber hinaus kann es auch nützlich sein, Statistikdaten zur Kardinalität für kombinierte Gruppen von Spalten zu erfassen. Das Optimierungsprogramm verwendet solche Informationen zur Erkennung von Spaltenkorrelationen bei der Abschätzung der Selektivität von Abfragen, die auf die Spalten in der Gruppe verweisen.

Die Beschreibung der folgenden Schritte geht von der Beispielannahme aus, dass die Datenbank **sales** die Tabelle **customers** mit den Indizes **custidx1** und **custidx2** enthält.

Sie müssen eine Verbindung zu der Datenbank mit den Tabellen und Indizes herstellen und über eine der folgenden Berechtigungsstufen verfügen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL
- Berechtigung SYSMAINT
- Berechtigung DBADM
- Zugriffsrecht CONTROL für die Tabelle

Anmerkung: Das Dienstprogramm RUNSTATS erfasst nur Statistikdaten für Tabellen in der Datenbankpartition, in der Sie es ausführen. Die RUNSTATS-Ergebnisse aus dieser Datenbankpartition werden für die anderen Datenbankpartitionen extrapoliert. Wenn die Datenbankpartition, in der Sie das Dienstprogramm RUNSTATS ausführen, keinen Teil einer Tabelle enthält, wird die Anforderung an die erste Datenbankpartition in der Datenbankpartitionsgruppe gesendet, die diesen Teil der Tabelle enthält.

Gehen Sie wie folgt vor, um Statistiken für bestimmte Spalten zu erfassen:

1. Stellen Sie eine Verbindung zur Datenbank **sales** her.
2. Führen Sie in Abhängigkeit von Ihren Anforderungen einen der folgenden Befehle über die DB2-Befehlszeile aus:
   - Zur Erfassung der Verteilungsstatistik für die Spalten **zip** und **ytdtotal**:
     ```
     RUNSTATS ON TABLE sales.customers
              WITH DISTRIBUTION ON COLUMNS (zip, ytdtotal)
     ```
   - Zur Erfassung der Verteilungsstatistik für die gleichen Spalten, jedoch mit Anpassung der Verteilungsstandardwerte:
     ```
     RUNSTATS ON TABLE sales.customers
              WITH DISTRIBUTION ON
              COLUMNS (zip, ytdtotal NUM_FREQVALUES 50 NUM_QUANTILES 75)
     ```
   - Zur Erfassung der Verteilungsstatistik für die Spalten, die in **custidx1** und **custidx2** indexiert sind:
     ```
     RUNSTATS ON TABLE sales.customer
                      ON KEY COLUMNS
     ```

- Zur Erfassung von Spaltenstatistiken nur für die Spalten **zip** und **ytdtotal** in der Tabelle sowie für eine Spaltengruppe, die die Spalten **region** und **territory** umfasst:

```
RUNSTATS ON TABLE sales.customers
                    ON COLUMNS (zip, (region, territory), ytdtotal)
```

- Angenommen, die Statistikdaten für andere Spalten (nicht XML) wurde mit dem Befehl LOAD zusammen mit der Option STATISTICS gesammelt. Zum Ergänzen der anderen Statistikdaten (nicht XML) durch die Statistikdaten der XML-Spalte **miscinfo**:

```
RUNSTATS ON TABLE sales.customers
      ON COLUMNS (miscinfo)
```

- Zur Erfassung der Spaltenstatistikdaten in der Tabelle nur für andere Spalten (nicht XML) (die Option EXCLUDING XML COLUMNS hat eine Vorrangstellung vor allen anderen Klauseln, die XML-Spalten angeben können):

```
RUNSTATS ON TABLE sales.customers
      EXCLUDING XML COLUMNS
```

Sie können Verteilungsstatistiken auch über die Steuerzentrale erfassen.

Erfassen von Indexstatistiken

Durch die Erfassung von Indexstatistikdaten erhält das Optimierungsprogramm die Möglichkeit, zu beurteilen, ob ein Index zur Erfüllung einer Abfrage verwendet werden sollte.

Die Beschreibung der folgenden Schritte geht von der Beispielannahme aus, dass die Datenbank **sales** die Tabelle **customers** mit den Indizes **custidx1** und **custidx2** enthält.

Sie müssen eine Verbindung zu der Datenbank mit den Tabellen und Indizes herstellen und über eine der folgenden Berechtigungsstufen verfügen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL
- Berechtigung SYSMAINT
- Berechtigung DBADM
- Zugriffsrecht CONTROL für die Tabelle

Die Ausführung von RUNSTATS mit der Option SAMPLED DETAILED erfordert 2 MB Statistikzwischenspeicher. Ordnen Sie wegen dieses zusätzlichen Speicherbedarfs dem Datenbankkonfigurationsparameter *stat_heap_sz* weitere 488 4-KB-Seiten zu. Wenn der Zwischenspeicher zu klein erscheint, gibt RUNSTATS einen Fehler zurück, bevor das Dienstprogramm versucht, Statistikdaten zu erfassen.

Gehen Sie wie folgt vor, um detaillierte Statistikdaten für einen Index zu erfassen:

1. Stellen Sie eine Verbindung zur Datenbank **sales** her.
2. Führen Sie in Abhängigkeit von Ihren Anforderungen einen der folgenden Befehle über die DB2-Befehlszeile aus:
   - Zur Erstellung detaillierter Statistikdaten für die beiden Indizes **custidx1** und **custidx2**:

     ```
     RUNSTATS ON TABLE sales.customers AND DETAILED INDEXES ALL
     ```

   - Zur Erstellung detaillierter Statistikdaten für die beiden Indizes, jedoch mit Stichprobenwerten (Sampling) anstelle detaillierter Berechnungen für jeden einzelnen Indexeintrag:

```
RUNSTATS ON TABLE sales.customers AND SAMPLED DETAILED INDEXES ALL
```

- Zur Erstellung detaillierter Stichprobenstatistiken für Indizes sowie zur Erstellung von Verteilungsstatistiken für die Tabelle, sodass die Statistikdaten für Indizes und Tabelle konsistent sind:

```
RUNSTATS ON TABLE sales.customers
          WITH DISTRIBUTION ON KEY COLUMNS
          AND SAMPLED DETAILED INDEXES ALL
```

Sie können Index- und Tabellenstatistiken auch über die Steuerzentrale erfassen.

Erfassen von Statistiken an einer Stichprobe der Tabellendaten

Tabellenstatistiken werden vom Abfrageoptimierungsprogramm bei der Auswahl des besten Zugriffsplans für eine gegebene Abfrage verwendet. Daher spielt es eine wichtige Rolle, dass Statistikdaten aktuell bleiben, um den Status einer Tabelle zu einem beliebigen Zeitpunkt adäquat wiederzugeben. Mit steigendem Aktivitätsvolumen für eine Tabelle sollte auch die Häufigkeit der Statistikerfassung erhöht werden. Mit wachsender Größe von Datenbanken wird es zunehmend wichtiger, effiziente Methoden zur Erfassung von Statistiken zu entwickeln. Zufällige Stichproben, die aus den Tabellendaten erstellt werden und über die Statistiken ermittelt werden, können die Leistung von RUNSTATS verbessern. Für E/A- oder CPU-gebundene Systeme können die Leistungsvorteile enorm sein. Je kleiner die Stichprobe, desto schneller wird RUNSTATS fertig.

Mit Version 8.2 stellt der Befehl RUNSTATS die Option TABLESAMPLE zur Erfassung von Statistiken über eine Stichprobe der Daten in der Tabelle zur Verfügung. Diese Funktion kann die Effizienz der Statistikerfassung erhöhen, da Stichproben nur eine Teilmenge der Daten verwenden. Gleichzeitig stellen Stichprobenmethoden einen hohen Grad an Genauigkeit sicher.

Es gibt zwei Möglichkeiten, die Art und Weise der Stichprobenerstellung anzugeben. Die BERNOULLI-Methode wählt die Stichprobendaten auf der Zeilenebene aus. Bei einer vollständigen Tabellensuche der Datenseite wird jede Zeile für sich geprüft und nach der Wahrscheinlichkeit P, ausgewählt, die durch den numerischen Parameter angegeben ist. Die Statistiken werden nur über diese ausgewählten Zeilen erfasst. In ähnlicher Weise erstellt die Methode SYSTEM eine Stichprobe der Daten auf Seitenebene. Das heißt, jede Seite wird mit einer Wahrscheinlichkeit 1-P/100 ausgewählt oder zurückgewiesen.

Die Leistung für Stichproben auf Seitenebene ist hervorragend, da nur eine E/A-Operation für jede ausgewählte Seite erforderlich ist. Bei Stichproben auf Zeilenebene ist der Ein-/Ausgabeaufwand nicht geringer, da jede Tabellenseite in einer vollständigen Tabellensuche abgerufen wird. Jedoch bieten Stichproben auf Zeilenebene erhebliche Verbesserungen, selbst wenn der Umfang der E/A-Operationen nicht kleiner wird, weil die Erfassung von Statistiken sehr rechenintensiv ist.

Stichproben auf Zeilenebene bieten Stichproben auf Seitenebene gegenüber Vorteile, wenn die Datenwerte eine hohe Clusterbildung aufweisen. Im Vergleich zur Seitenstichprobe vermittelt die Zeilenstichprobe einen genaueren Eindruck von den Daten, da sie P Prozent Zeilen von jeder Datenseite enthält. Bei der Seitenstichprobe befinden jeweils alle Zeilen der P Prozent Seiten in der Stichprobenmenge. Wenn die Zeilen nach dem Zufallsprinzip über die Tabelle verteilt sind, sind die Genauigkeiten von Statistiken über Zeilenstichproben und Seitenstichproben ähnlich.

Jede Stichprobe wird bei jeder Ausführung des Befehls RUNSTATS nach dem Zufallsprinzip neu generiert, sofern nicht die Option REPEATABLE verwendet wird. Mit der Klausel REPEATABLE wird die gleiche Stichprobe generiert wie bei der letzten Ausführung des Befehls RUNSTATS mit der Option TABLESAMPLE. Für Benutzer kann dies in solchen Fällen nützlich sein, in denen eine Generierung konsistenter Statistiken für Tabellen mit konstanten Daten wünschenswert ist.

Erfassen von Statistiken mit einem Statistikprofil

Das Dienstprogramm RUNSTATS bietet eine Option zur Registrierung und Verwendung eines Statistikprofils, bei dem es sich um eine Gruppe von Optionen handelt, die angeben, welche Statistiken für eine bestimmte Tabelle zu erfassen sind, wie zum Beispiel Tabellenstatistiken, Indexstatistiken oder Verteilungsstatistiken.

Diese Einrichtung vereinfacht die Erfassung von Statistiken, indem es Ihnen die Möglichkeit gibt, die Optionen, die Sie bei der Ausführung des Befehls RUNSTATS angeben, zu speichern, sodass Sie die gleichen Statistiken wiederholt für eine Tabelle erfassen können, ohne die Befehlsoptionen erneut eingeben zu müssen.

Sie können ein Statistikprofil mit und ohne gleichzeitige Erfassung von Statistiken registrieren oder aktualisieren. Um zum Beispiel ein Profil zu registrieren und gleichzeitig Statistiken zu erfassen, führen Sie den Befehl RUNSTATS mit der Option SET PROFILE aus. Soll nur ein Profil registriert werden, ohne tatsächlich Statistiken zu erfassen, führen Sie den Befehl RUNSTATS mit der Option SET PROFILE ONLY aus.

Zur Erfassung von Statistiken unter Verwendung eines bereits registrierten Statistikprofils führen Sie den Befehl RUNSTATS aus, indem Sie nur den Namen der Tabelle und die Option USE PROFILE angeben.

Wenn Sie prüfen wollen, welche Optionen zurzeit in dem Statistikprofil für eine bestimmte Tabelle angegeben sind, können Sie die Katalogtabellen mit der folgenden SELECT-Anweisung abfragen, wobei `tabellenname` der Name der Tabelle ist, für die Sie das Profil abrufen wollen:

```
SELECT STATISTICS_PROFILE FROM SYSIBM.SYSTABLES WHERE NAME = tabellenname
```

Automatische Statistikprofilerstellung

Statistikprofile können außerdem automatisch durch die DB2-Funktion zur automatischen Statistikprofilerstellung generiert werden. Wenn diese Funktion aktiviert ist, werden Informationen zur Datenbankaktivität gesammelt und in einem Abfrage-Feedback-Warehouse gespeichert. Auf der Grundlage dieser Daten wird ein Statistikprofil generiert. Die Aktivierung dieser Funktion kann das Problem der Unsicherheit, welche Statistiken für eine bestimmte Auslastung relevant sind, etwas mildern und die Erfassung einer minimalen Gruppe von Statistiken ermöglichen, um eine optimale Leistung für die Auslastung der Datenbank zu erzielen.

Diese Funktion kann zusammen mit der Funktion zur automatischen Statistikerfassung verwendet werden, die wiederum automatisch Statistikpflegeoperationen auf der Basis der Informationen terminiert, die in dem automatisch generierten Statistikprofil enthalten sind.

Zur Aktivierung dieser Funktion müssen Sie die automatische Tabellenpflege bereits mithilfe der entsprechenden Konfigurationsparameter aktiviert haben. Der Konfigurationsparameter AUTO_STATS_PROF aktiviert die Erfassung von Abfrage-

feedbackdaten, während der Konfigurationsparameter AUTO_PROF_UPD die Generierung eines Statistikprofils zur Verwendung durch die automatische Statistikerfassung aktiviert.

Anmerkung: Die automatische Statistikprofilerstellung kann nur im seriellen DB2-Modus aktiviert werden und wird für Abfragen in gewissen föderierten Umgebungen, in MPP-Umgebungen mit mehreren Partitionen sowie bei aktivierter partitionsinterner Parallelität blockiert.

Die Statistikprofilgenerierung eignet sich am besten für Umgebungen, in denen große, komplexe Abfragen ausgeführt werden, die viele Vergleichselemente anwenden und häufig Korrelationen in den Daten der Vergleichselementspalten haben und Joins und Gruppierungen mehrerer Tabellen erfordern. Sie eignet sich weniger für Umgebungen mit einer in erster Linie transaktionsorientierten Auslastung.

Es gibt zwei unterschiedliche Möglichkeiten, diese Funktion zu verwenden:

- In einer Testumgebung. Setzen Sie AUTO_STATS_PROF und AUTO_PROF_UPD auf ON in Testsystemen, in denen der Leistungsaufwand der Laufzeitüberwachung leicht toleriert werden kann. Wenn das Testsystem mit realistischen Daten und Abfragen arbeitet, ermöglicht dies eine Ermittlung der tatsächlichen Korrelationen und Einstellungen der Statistikparameter für RUNSTATS, die wiederum im Statistikprofil gespeichert werden. Diese Profile können anschließend auf das Produktionssystem übertragen werden, auf dem die Abfragen von ihnen profitieren können, ohne den Überwachungsaufwand zu verursachen.
- Zur Untersuchung von Leistungsproblemen für bestimmte Abfragen in einer Produktionsumgebung. Wenn Leistungsprobleme für eine bestimmte Gruppe von Abfragen festgestellt werden, die fehlerhaften Statistiken oder Korrelationen zugeschrieben werden können, können Sie AUTO_STATS_PROF aktivieren und die Zielauslastung über einen Zeitraum ausführen. Die automatische Statistikprofilgenerierung analysiert das Abfragefeedback und erstellt Empfehlungen in den Tabellen SYSTOOLS.OPT_FEEDBACK_RANKING*. Sie können diese Empfehlungen einsehen und die Statistikprofile manuell auf der Grundlage der Empfehlungen verfeinern. Wenn DB2 die Statistikprofile auf der Grundlage dieser Empfehlungen automatisch aktualisieren soll, aktivieren Sie AUTO_PROF_UPD, wenn Sie AUTO_STATS_PROF aktivieren.

Anmerkung: Die Überwachung der Abfragen und das Speichern von Abfragefeedbackdaten im Feedback-Warehouse ist mit einigem Leistungsaufwand verbunden.

Erstellen des Warehouse für Abfragefeedback

Das Feedback-Warehouse besteht aus fünf Tabellen im Schema SYSTOOLS, in denen Informationen zu den Vergleichselementen, die bei der Ausführung von Abfragen angetroffen werden, sowie Empfehlungen für die Statistikerfassung gespeichert werden. Diese fünf Tabellen heißen OPT_FEEDBACK_QUERY, OPT-_FEEDBACK_PREDICATE, OPT_FEEDBACK_PREDICATE_COLUMN, OPT_FEEDBACK_RANKING und OPT_FEEDBACK_RANKING_COLUMN.

Zur Verwendung der automatischen Statistikprofilgenerierung müssen Sie zunächst das Abfrage-Feedback-Warehouse mithilfe der gespeicherten Prozedur SYSINSTALLOBJECTS erstellen. Diese gespeicherte Prozedur ist die allgemeine gespeicherte Prozedur zum Erstellen und Löschen von Objekten im Schema SYSTOOLS.

Rufen Sie die gespeicherte Prozedur SYSINSTALLOBJECTS wie folgt auf:

```
call SYSINSTALLOBJECTS ( toolname, aktion, tabellenbereichsname, schemaname)
```

Dabei gilt Folgendes:

toolname
Gibt den Namen des Tools an, dessen Objekte zu erstellen oder löschen sind. In in diesem Fall 'ASP' bzw. 'AUTO STATS PROFILING'.

action Gibt die auszuführende Aktion an: 'C' für Erstellen (Create) und 'D' für Löschen (Drop).

tabellenbereichsname
Der Name des Tabellenbereichs, in dem die Tabellen für das Feedback-Warehouse erstellt werden sollen. Dieser Eingabeparameter ist optional. Wenn er nicht angegeben wird, wird der Standardbenutzertabellenbereich verwendet.

schemaname
Der Name des Schemas, in dem die Objekte erstellt oder gelöscht werden. Dieser Parameter wird momentan nicht verwendet.

Wenn Sie zum Beispiel das Feedback-Warehouse in Tabellenbereich ″A″ erstellen wollen, geben Sie Folgendes ein: `call SYSINSTALLOBJECTS ('ASP', 'C', 'A', '')`

Katalogstatistiktabellen

Die folgenden Tabellen bieten Informationen zu den Systemkatalogtabellen, die Katalogstatistiken enthalten, und zeigen die RUNSTATS-Optionen, die zur Erfassung bestimmter Statistiken verwendet werden.

Tabelle 54. Tabellenstatistiken (SYSCAT.TABLES und SYSSTAT.TABLES)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	Indizes
FPAGES	Anzahl der von einer Tabelle verwendeten Seiten	Ja	Ja
NPAGES	Anzahl der Zeilen enthaltenden Seiten	Ja	Ja
OVERFLOW	Anzahl der Überlaufzeilen	Ja	Nein
CARD	Anzahl der Zeilen in der Tabelle (Kardinalität)	Ja	Ja (Anm. 1)
ACTIVE_BLOCKS	Für MDC-Tabellen: die Gesamtzahl belegter Blöcke	Ja	Nein

Anmerkung:

1. Wenn für die Tabelle keine Indizes erstellt wurden und Sie die Indexstatistik anfordern, wird die CARD-Statistik mit keinem neuen Wert aktualisiert. Die vorige CARD-Statistik wird beibehalten.

Tabelle 55. Spaltenstatistiken (SYSCAT.COLUMNS und SYSSTAT.COLUMNS)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	Indizes
COLCARD	Anzahl der unterschiedlichen Werte der Spalte (Spaltenkardinalität)	Ja	Ja (Anm. 1)

Tabelle 55. Spaltenstatistiken (SYSCAT.COLUMNS und SYSSTAT.COLUMNS) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
AVGCOLLEN	Durchschnittslänge der Spalte	Ja	Ja (Anm. 1)
HIGH2KEY	Zweithöchster Wert der Spalte	Ja	Ja (Anm. 1)
LOW2KEY	Zweitniedrigster Wert der Spalte	Ja	Ja (Anm. 1)
NUMNULLS	Die Anzahl Nullwerte (NULLs) in einer Spalte	Ja	Ja (Anm. 1)
SUB_COUNT	Die durchschnittliche Anzahl von Unterelementen	Ja	Nein (Anm. 2)
SUB_DELIM_LENGTH	Durchschnittslänge jedes Begrenzers, der ein Unterelement trennt	Ja	Nein (Anm. 2)

Anmerkung:

1. Spaltenstatistiken werden für die erste Spalte im Indexschlüssel erfasst.
2. Diese Statistiken stellen Informationen zu Daten in Spalten bereit, die eine Reihe von Unterfeldern bzw. Unterelementen enthalten, die durch Leerzeichen begrenzt werden. Die Statistikwerte SUB_COUNT und SUB_DELIM_LENGTH werden nur für Spalten der Typen CHAR und VARCHAR mit einem Codepageattribut eines Einzelbytezeichensatzes (SBCS), mit FOR BIT DATA oder mit UTF-8 erfasst.

Tabelle 56. Spaltengruppenstatistiken (SYSCAT.COLGROUPS und SYSSTAT.COLGROUPS)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
COLGROUPCARD	Kardinalität der Spaltengruppe	Ja	Nein

Anmerkung: Die Verteilungsstatistiken für Spaltengruppen in den folgenden beiden Tabellen werden nicht von RUNSTATS erfasst. Sie können nicht manuell aktualisiert werden.

Tabelle 57. Verteilungsstatistiken für Spaltengruppen (SYSCAT.COLGROUPDIST und SYSSTAT.COLGROUPDIST)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
TYPE	F = Häufigkeitswert Q = Quantilwert	Ja	Nein
ORDINAL	Ordinalzahl der Spalte in der Gruppe	Ja	Nein
SEQNO	Folgenummer *n*, die den *n*-ten TYPE-Wert darstellt	Ja	Nein
COLVALUE	Der Datenwert als Zeichenliteral oder Nullwert	Ja	Nein

Tabelle 58. Verteilungsstatistiken für Spaltengruppen 2 (SYSCAT.COLGROUPDISTCOUNTS und SYSSTAT.COLGROUPDISTCOUNTS)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
TYPE	F = Häufigkeitswert Q = Quantilwert	Ja	Nein
SEQNO	Folgenummer *n*, die den *n*-ten TYPE-Wert darstellt	Ja	Nein
VALCOUNT	Bei TYPE = F ist VALCOUNT die Anzahl Vorkommen von COLVALUE-Werten für die Spaltengruppe, die durch diese Folgenummer (SEQNO) angegeben ist. Bei TYPE = Q ist VALCOUNT die Anzahl Zeilen, deren Wert kleiner oder gleich COLVALUE-Werten für die Spaltengruppe mit dieser Folgenummer ist.	Ja	Nein
DISTCOUNT	Bei TYPE = Q enthält diese Spalte die Anzahl unterschiedlicher Werte, die kleiner oder gleich COLVALUE-Werten für die Spaltengruppe mit dieser Folgenummer (SEQNO) sind. Null, falls nicht verfügbar.	Ja	Nein

Tabelle 59. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
NLEAF	Anzahl der Indexblattseiten (leaf pages)	Nein	Ja
NLEVELS	Anzahl der Indexstufen	Nein	Ja
CLUSTERRATIO	Grad der Clusterbildung der Tabellendaten	Nein	Ja (Anm. 2)
CLUSTERFACTOR	Feinerer Grad der Clusterbildung	Nein	Detailliert (Anm. 1,2)
DENSITY	Verhältnis (Prozentsatz) von SEQUENTIAL_ PAGES zur Anzahl der Seiten im Bereich der vom Index belegten Seiten (Anm. 3)	Nein	Ja
FIRSTKEYCARD	Anzahl der unterschiedlichen Werte in der ersten Spalte des Index	Nein	Ja

Tabelle 59. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
FIRST2KEYCARD	Anzahl der unterschiedlichen Werte in den ersten beiden Spalten des Index	Nein	Ja
FIRST3KEYCARD	Anzahl der unterschiedlichen Werte in den ersten drei Spalten des Index	Nein	Ja
FIRST4KEYCARD	Anzahl der unterschiedlichen Werte in den ersten vier Spalten des Index	Nein	Ja
FULLKEYCARD	Anzahl der unterschiedlichen Werte in allen Spalten des Index, mit Ausnahme der Schlüsselwerte in einem Index des Typs 2, für die alle Satz-IDs (RIDs) als gelöscht markiert sind	Nein	Ja
PAGE_FETCH_PAIRS	Geschätzte Anzahl der Seitenabrufe für verschiedene Puffergrößen	Nein	Detailliert (Anm. 1,2)
AVGPARTITION_CLUSTERRATIO	Der Grad der Datenclusterbildung in einer einzigen Datenpartition.	Nein	Ja (Anm. 2)
AVGPARTITION_CLUSTERFACTOR	Feinerer Messwert des Grades der Clusterbildung in einer einzigen Datenpartition.	Nein	Detailliert (Anm. 1,2)
AVGPARTITION_PAGE_FETCH_PAIRS	Geschätzte Anzahl der Seitenabrufe für verschiedene Puffergrößen, die auf der Basis einer einzigen Datenpartition generiert wurden.	Nein	Detailliert (Anm. 1,2)
DATAPARTITION_CLUSTERFACTOR	Die Anzahl an Datenpartitionsverweisen bei einer Indexsuche.	Nein (Anm. 6)	Ja (Anm. 6)
SEQUENTIAL_PAGES	Anzahl der Blattseiten (äußersten Seiten), die auf der Platte in der durch den Indexschlüssel definierten Reihenfolge mit wenigen oder keinen großen dazwischen liegenden Lücken gespeichert sind	Nein	Ja
AVERAGE_SEQUENCE_PAGES	Durchschnittliche Anzahl von Indexseiten, auf die in Reihenfolge zugegriffen werden kann. Dies ist die Anzahl von Indexseiten, die von Vorablesefunktionen als in der richtigen Reihenfolge vorliegend erkannt werden können.	Nein	Ja

Tabelle 59. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
AVERAGE_RANDOM_PAGES	Durchschnittliche Anzahl von Indexseiten, auf die zwischen den sequenziellen Seitenzugriffen wahlfrei zugegriffen werden muss	Nein	Ja
AVERAGE_SEQUENCE_GAP	Lücke zwischen den Seitenfolgen	Nein	Ja
AVERAGE_SEQUENCE_FETCH_PAGES	Durchschnittliche Anzahl von Tabellenseiten, auf die in Reihenfolge zugegriffen werden kann. Dies ist die Anzahl von Tabellenseiten, die von Vorablesefunktionen als in der richtigen Reihenfolge vorliegend erkannt werden können, wenn sie versuchen, Tabellenzeilen über den Index abzurufen.	Nein	Ja (Anm. 4)
AVERAGE_RANDOM_FETCH_PAGES	Durchschnittliche Anzahl von Tabellenseiten, auf die ein wahlfreier Zugriff erfolgt, zwischen sequenziellen Seitenzugriffen beim Abrufen von Tabellenzeilen über den Index	Nein	Ja (Anm. 4)
AVERAGE_SEQUENCE_FETCH_GAP	Lücke zwischen sequenziellen Zugriffen beim Abrufen von Tabellenzeilen über den Index	Nein	Ja (Anm. 4)
NUMRIDS	Anzahl von Satz-IDs (RIDs) im Index, einschließlich gelöschter Satz-IDs in Indizes des Typs 2	Nein	Ja
NUMRIDS_DELETED	Gesamtanzahl von Satz-IDs (RIDs), die im Index als gelöscht markiert sind, außer den Satz-IDs in Blattseiten, in denen alle Satz-IDs als gelöscht markiert sind	Nein	Ja
NUM_EMPTY_LEAFS	Gesamtanzahl von Blattseiten, in denen alle Satz-IDs als gelöscht markiert sind	Nein	Ja
INDCARD	Anzahl von Indexeinträgen (Indexkardinalität)	Nein	Ja

Tabelle 59. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	Indizes
Anmerkung: 1. Detaillierte Indexstatistikdaten werden erfasst, indem die Klausel DETAILED im Befehl RUNSTATS angegeben wird. 2. CLUSTERFACTOR und PAGE_FETCH_PAIRS werden mit der Klausel DETAILED nur dann erfasst, wenn die Tabelle eine beträchtliche Größe aufweist. Wenn die Tabelle größer als ca. 25 Seiten ist, werden die Statistikdaten für die Spalten CLUSTERFACTOR und PAGE_FETCH_PAIRS gesammelt. In diesem Fall ist CLUSTERRATIO (Clusterverhältnis) -1 (d. h. wird nicht gesammelt). Wenn die Tabelle relativ klein ist, wird nur die Spalte CLUSTERRATIO vom Dienstprogramm RUNSTATS ausgefüllt, während die Spalten CLUSTERFACTOR und PAGE_FETCH_PAIRS nicht berechnet werden. Wenn die Klausel DETAILED nicht angegeben wird, werden nur die Statistikdaten für CLUSTERRATIO erfasst. 3. Diese Statistik ermittelt den Prozentsatz von Seiten in der Datei, die den Index enthalten, der zu dieser Tabelle gehört. Für eine Tabelle, die nur einen definierten Index hat, sollte DENSITY normalerweise gleich 100 sein. DENSITY wird vom Optimierungsprogramm zur Abschätzung verwendet, wie viele irrelevante Seiten von anderen Indizes durchschnittlich vielleicht gelesen werden, wenn die Indexseiten vorabgelesen würden. 4. Diese Statistiken können nicht berechnet werden, wenn sich die Tabelle in einem DMS-Tabellenbereich befindet. 5. Statistiken über Vorablesezugriffe werden beim Laden von Indizes (LOAD INDEX) bzw. beim Erstellen von Indizes (CREATE INDEX) nicht erfasst, und zwar auch dann nicht, wenn die Funktion zum Erfassen von Statistiken beim Aufrufen des Befehls angegeben wird. Statistiken über Vorablesezugriffe werden ebenfalls nicht erfasst, wenn der Konfigurationsparameter für die Markierung der Sequenzerkennung (seqdetect) ausgeschaltet ist. 6. Wenn die RUNSTATS-Optionen für die Tabelle auf ″Nein″ gesetzt sind, bedeutet dies, dass bei der Erfassung der Tabellenstatistik keine Statistikdaten erfasst werden; ″Ja″ für Indizes bedeutet, dass Statistikdaten erfasst werden, wenn der Befehl RUNSTATS mit den INDEXES-Optionen verwendet wird.			

Tabelle 60. Spaltenverteilungsstatistiken (SYSCAT.COLDIST und SYSSTAT.COLDIST)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	Indizes
DISTCOUNT	Wenn TYPE den Wert Q hat, die Anzahl unterschiedlicher Werte, die kleiner oder gleich dem Statistikwert COLVALUE sind	Verteilung (Anm. 2)	Nein
TYPE	Gibt an, ob die Zeile statistische Daten über die Häufigkeit der Werte oder über Quantilwerte enthält	Verteilung	Nein
SEQNO	Die Stelle in der Häufigkeitsrangfolge einer Folgenummer, die als Hilfe zur eindeutigen Bestimmung der Zeile in der Tabelle verwendet werden kann	Verteilung	Nein
COLVALUE	Datenwert, für den Statistikdaten zur Häufigkeit oder zu Quantilwerten erfasst werden	Verteilung	Nein

Tabelle 60. Spaltenverteilungsstatistiken (SYSCAT.COLDIST und SYSSTAT.COLDIST) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
VALCOUNT	Häufigkeit, mit der der Datenwert in der Spalte auftritt, oder bei Quantilwerten die Anzahl der Werte, die kleiner oder gleich dem Datenwert (COLVALUE) sind	Verteilung	Nein

Anmerkung:

1. Verteilungsstatistikdaten über Spalten werden gesammelt, indem die Klausel WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird. Beachten Sie, dass Verteilungsstatistikdaten **nicht** gesammelt werden, wenn das Ausmaß der Ungleichmäßigkeit der Werteverteilung in den Spaltenwerten nicht hoch genug ist.
2. DISTCOUNT wird nur für Spalten erfasst, die die erste Schlüsselspalte in einem Index bilden.

Verteilungsstatistiken

Sie können zwei Arten von Statistiken über die Datenverteilung erfassen:

- Statistiken zur Häufigkeit von Datenwerten

  Diese Statistiken stellen Informationen über die Spalte und den Datenwert mit der höchsten Anzahl mehrfacher Vorkommen, die nächsthöchste Anzahl mehrfach vorkommender Werte usw. bis zu der Stufe, die durch den Wert des Datenbankkonfigurationsparameters *num_freqvalues* angegeben wird, bereit. Um die Erfassung der Statistik über die Häufigkeit von Werten zu inaktivieren, müssen Sie den Parameter *num_freqvalues* auf den Wert 0 setzen.

  Sie können den Wert für *num_freqvalues* auch als RUNSTATS-Optionen für jede Tabelle oder statistische Sicht und bestimmte Spalten festlegen.

- Quantilstatistiken

  Diese Statistiken liefern Informationen darüber, wie Datenwerte in Relation zu anderen Werten verteilt sind. Die Statistiken der so genannten K-Quantile stellen den Wert V dar, bei oder unter dem mindestens K Werte liegen. Ein K-Quantil lässt sich durch Sortieren der Werte in aufsteigender Reihenfolge ermitteln. Der K-Quantilwert ist der Wert an der K-ten Position vom unteren Ende des Wertebereichs betrachtet.

  Um die Anzahl der Abschnitte anzugeben, in die die Spaltendatenwerte gruppiert werden sollen, setzen Sie den Datenbankkonfigurationsparameter *num_quantiles* auf einen Wert zwischen 2 und 32.767. Der Standardwert ist 20. Dieser Wert stellt einen Schätzfehler durch das Optimierungsprogramm von maximal plus/minus 2,5% für ein beliebiges Vergleichselement mit einem der Operatoren „=", „>" oder „<" sowie einen maximalen Fehler von plus/minus 5% für ein beliebiges BETWEEN-Vergleichselement sicher. Zur Inaktivierung der Erfassung der Quantilstatistiken setzen Sie den Parameter *num_quantiles* auf den Wert 0 oder 1.

  Sie können den Parameter *num_quantiles* für jede Tabelle oder statistische Sicht und für bestimmte Spalten definieren.

Anmerkung: Wenn Sie höhere Werte für die Parameter *num_freqvalues* und *num_quantiles* angeben, werden bei der Ausführung von RUNSTATS mehr CPU-Kapazitäten und Speicher entsprechend dem Datenbankkonfigurationsparameter *stat_heap_sz* benötigt.

Wann sind Verteilungsstatistiken zu erfassen? Bei der Entscheidung, ob Verteilungsstatistikdaten für eine bestimmte Tabelle oder statistische Sicht erstellt und aktualisiert werden sollten, spielen zwei Faktoren eine ausschlaggebende Rolle:

- Ob Anwendungen mit statischen oder dynamischen SQL- oder XQuery-Anweisungen arbeiten.

  Verteilungsstatistiken sind am besten für dynamische und statische Abfragen geeignet, die keine Hostvariablen verwenden. Wenn Abfragen mit Hostvariablen verwendet werden, nutzt das Optimierungsprogramm die Verteilungsstatistikdaten nur im begrenzten Umfang.
- Ob die Datenwerte in Spalten gleichmäßig verteilt sind.

  Erstellen Verteilungsstatistiken, wenn mindestens in einer Spalte der Tabelle die Datenwerte höchst „ungleichmäßig" verteilt sind und diese Spalte häufig in Gleichheits- bzw. Bereichsvergleichselementen auftritt, wie zum Beispiel in folgenden Klauseln:

  ```
  WHERE C1 = KEY;
  WHERE C1 IN (KEY1, KEY2, KEY3);
  WHERE (C1 = KEY1) OR (C1 = KEY2) OR (C1 = KEY3);
  WHERE C1 <= KEY;
  WHERE C1 BETWEEN KEY1 AND KEY2;
  ```

Zwei Arten ungleichmäßiger Datenverteilungen treten möglicherweise zusammen auf:

- Datenwerte können in einem oder mehreren Unterintervallen gehäuft auftreten, anstatt gleichmäßig zwischen dem höchsten und niedrigsten Datenwert verteilt zu sein. Betrachten Sie die folgende Spalte, in der die Datenwerte im Bereich (5,10) eine Häufung aufweisen:

C1
0,0
5,1
6,3
7,1
8,2
8,4
8,5
9,1
93,6
100,0

  Quantilstatistiken helfen dem Optimierungsprogramms bei der Einschätzung dieser Art von Datenverteilung.

  Zur Ermittlung, ob Spaltendaten nicht gleichmäßig verteilt sind, können Sie eine Abfrage wie in folgendem Beispiel ausführen:

  ```
  SELECT C1, COUNT(*) AS OCCURRENCES
    FROM T1
  GROUP BY C1
  ORDER BY OCCURRENCES DESC;
  ```
- Mehrfach auftretende Datenwerte können häufig vorkommen. Betrachten Sie eine Spalte, in der die Daten mit den folgenden Häufigkeiten verteilt sind:

Datenwert	**Häufigkeit**
20	5
30	10
40	10

Datenwert	Häufigkeit
50	25
60	25
70	20
80	5

Zur Unterstützung des Optimierungsprogramms bei der Einschätzung mehrfach auftretender Werte sollten Sie sowohl Quantilstatistiken als auch Statistiken zur Wertehäufigkeit erstellen.

Wann sind nur Indexstatistiken zu erfassen

Sie können Statistiken nur für Indexdaten in folgenden Situationen erfassen:

- Seit der letzten Ausführung des Dienstprogramms RUNSTATS wurde ein neuer Index erstellt, und Sie möchten nicht erneut statistische Daten für die Tabellendaten sammeln.
- Es wurden viele Änderungen an den Daten vorgenommen, die die erste Spalte eines Index betreffen.

Welche Stufe an statistischer Genauigkeit ist anzugeben

Zur Festlegung der Genauigkeit, mit der Verteilungsstatistiken erfasst werden, definieren Sie die Datenbankkonfigurationsparameter *num_quantiles* und *num_freqvalues* mit den entsprechenden Werten. Sie können diese Parameter auch als RUNSTATS-Optionen angeben, wenn Sie Statistiken für eine Tabelle bzw. für Spalten erfassen. Je höher Sie diese Werte setzen, desto größer ist die Genauigkeit, mit der RUNSTATS Verteilungsstatistiken erstellt und aktualisiert. Allerdings erfordert eine größere Genauigkeit auch mehr Ressourcen, sowohl während der Ausführung von RUNSTATS als auch für die Speicherung in den Katalogtabellen.

Für die meisten Datenbanken empfiehlt sich ein Wert zwischen 10 und 100 für den Datenbankkonfigurationsparameter *num_freqvalues*. Im Idealfall sollten Häufigkeitsstatistikdaten so erstellt werden, dass die Häufigkeiten der verbleibenden Werte entweder einander annähernd gleich sind oder im Vergleich zu den Häufigkeiten der häufigsten Werte vernachlässigbar sind. Der Datenbankmanager erfasst unter Umständen weniger als diese Anzahl, da diese Statistikdaten nur für Datenwerte erhoben werden, die mehr als einmal auftreten. Wenn Sie nur Quantilstatistiken erfassen müssen, setzen Sie den Parameter *num_freqvalues* auf den Wert 0.

Zur Einstellung der Anzahl von Quantilen geben Sie einen Wert zwischen 20 und 50 für den Datenbankkonfigurationsparameter *num_quantiles* an. Als grobe Faustregel zur Bestimmung der Anzahl von Quantilen gilt:

- 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 es sich um ein BETWEEN-Vergleichselement handelt, und 50/P, wenn das Vergleichselement eine andere Art von Bereichsvergleichselement (<, <=, > oder >=) ist.

Zum Beispiel ergeben 25 Quantile einen maximalen Schätzfehler von 4 % bei BETWEEN-Vergleichselementen und 2 % bei Vergleichselementen mit ″>″. Allgemein sollten mindestens 10 Quantile angegeben werden. Mehr als 50 Quantile sind nur bei extrem ungleichmäßig verteilten Daten erforderlich. Wenn Sie nur Statistiken über die häufigsten Werte benötigen, setzen Sie den Parameter *num_quantiles* auf

den Wert 0. Wenn Sie diesen Parameter auf den Wert „1" setzen, werden keine Quantilstatistikdaten erfasst, da der gesamte Wertebereich in diesem Fall in nur einem Quantil liegt.

Verwendung der Verteilungsstatistiken durch das Optimierungsprogramm

Das Optimierungsprogramm nutzt Verteilungsstatistiken zu besseren Abschätzungen des Aufwands für verschiedene mögliche Zugriffspläne zur Erfüllung von Abfragen.

Wenn Sie das Dienstprogramm RUNSTATS nicht mit der Klausel WITH DISTRIBUTION ausführen, enthalten die Katalogstatistiktabellen nur Informationen über die Größe der Tabelle oder der statistischen Sicht, die höchsten und niedrigsten Werte in der Tabelle oder der statistischen Sicht, den Grad der Clusterbildung der Tabelle in Bezug auf ihre Indizes und die Anzahl der unterschiedlichen Werte in den indexierten Spalten.

Sofern keine zusätzlichen Informationen über die Verteilung von Werten zwischen dem niedrigsten und dem höchsten Wert vorliegen, geht das Optimierungsprogramm davon aus, dass die Datenwerte gleichmäßig verteilt sind. Wenn die Datenwerte erhebliche Abweichungen voneinander zeigen, in bestimmten Abschnitten des Wertebereichs Häufungen aufweisen oder einzelne Werte besonders zahlreich vorkommen, wählt das Optimierungsprogramm möglicherweise einen nicht optimalen Zugriffsplan aus.

Betrachten Sie das folgende Beispiel:

Das Optimierungsprogramm muss die Anzahl von Zeilen abschätzen, die einen Spaltenwert enthalten, der ein Gleichheits- oder Bereichsvergleichselement erfüllt, um den aufwandsgünstigsten Zugriffsplan auszuwählen. Je genauer die Schätzung ist, desto größer ist auch die Wahrscheinlichkeit, dass das Optimierungsprogramm den optimalen Zugriffsplan wählt. Betrachten Sie zum Beispiel die folgende Abfrage:

```
SELECT C1, C2
  FROM TABLE1
  WHERE C1 = 'NEW YORK'
  AND C2 <= 10
```

Nehmen Sie an, dass es einen Index sowohl für C1 als auch für C2 gibt. Ein möglicher Zugriffsplan besteht darin, über den Index für C1 alle Zeilen mit `C1 = 'NEW YORK'` abzurufen und anschließend jede abgerufene Zeile daraufhin zu überprüfen, ob `C2 <= 10` gilt. Ein alternativer Plan wäre, über den Index für C2 alle Zeilen mit `C2 <= 10` abzurufen und anschließend jede abgerufene Zeile daraufhin zu überprüfen, ob `C1 = 'NEW YORK'` gilt. Da der Hauptaufwand bei der Ausführung einer Abfrage in der Regel im Zusammenhang mit dem Abrufen der Zeilen anfällt, ist der Plan der beste, der die wenigsten Abrufe erfordert. Zur Auswahl dieses Plans ist eine Abschätzung der Anzahl von Zeilen erforderlich, die das jeweilige Vergleichselement erfüllen.

Wenn keine Verteilungsstatistiken verfügbar sind, jedoch RUNSTATS für eine Tabelle oder eine statistische Sicht ausgeführt wurde, sind die einzigen Informationen, die dem Optimierungsprogramm zur Verfügung stehen, der zweithöchste Datenwert (HIGH2KEY), der zweitniedrigste Datenwert (LOW2KEY), die Anzahl unterschiedlicher Werte (COLCARD) und die Anzahl von Zeilen für eine Spalte (CARD). Die Anzahl der Zeilen, die ein Gleichheitsvergleichselement oder ein

Bereichsvergleichselement erfüllen, wird dann unter der Annahme abgeschätzt, dass die Häufigkeiten der Datenwerte in einer Spalte alle gleich und die Datenwerte gleichmäßig über das Intervall (LOW2KEY, HIGH2KEY) verteilt sind. Im Einzelnen wird die Anzahl der Zeilen, die ein Gleichheitsvergleichselement `C1 = KEY` erfüllen, mit dem Wert CARD/COLCARD abgeschätzt. Die Anzahl der Zeilen, die ein Bereichsvergleichselement `C1 BETWEEN KEY1 AND KEY2` erfüllen, wird nach folgender Formel abgeschätzt:

```
      KEY2 - KEY1
------------------  x CARD       (1)
 HIGH2KEY - LOW2KEY
```

Diese Schätzwerte sind nur dann realistisch, wenn die tatsächliche Verteilung der Datenwerte weitgehend gleichmäßig ist. Wenn keine Verteilungsstatistikdaten verfügbar sind und entweder die Häufigkeiten der Datenwerte grob von einander abweichen oder die Datenwerte in einigen wenigen Unterbereichen des Intervalls (LOW_KEY, HIGH_KEY) Häufungen bilden, können die Schätzwerte um Größenordnungen von der Realität abweichen, sodass das Optimierungsprogramm möglicherweise einen nicht optimalen Zugriffsplan wählt.

Wenn Verteilungsstatistikdaten verfügbar sind, können die oben beschriebenen Schätzfehler wesentlich verringert werden, indem die statistischen Daten zur Häufigkeit der Werte zur Berechnung der Anzahl von Zeilen, die ein Gleichheitsvergleichselement erfüllen, und sowohl die statistischen Daten zur Häufigkeit der Werte als auch die Quantilstatistik zur Berechnung der Anzahl von Zeilen, die ein Bereichsvergleichselement erfüllen, herangezogen werden.

Erweiterte Beispiele zur Verwendung von Verteilungsstatistiken

Zur Erläuterung, wie das Optimierungsprogramm Verteilungsstatistiken nutzen kann, soll zunächst eine Abfrage betrachtet werden, die ein Gleichheitsvergleichselement der Form `C1 = KEY` enthält.

Beispiel für Statistiken über die Wertehäufigkeit

Wenn Statistiken zur Wertehäufigkeit verfügbar sind, kann das Optimierungsprogramm diese wie folgt zur Auswahl eines geeigneten Zugriffsplans verwenden:

- Wenn `KEY` einer der N häufigsten Werte ist, dann verwendet das Optimierungsprogramm die Häufigkeit von `KEY`, die im Katalog gespeichert ist.
- Wenn `KEY` nicht einer der N häufigsten Werte ist, schätzt das Optimierungsprogramm die Anzahl der Zeilen, die das Vergleichselement erfüllen, unter der Annahme ab, dass die nicht häufigen Werte (COLCARD - N) eine gleichmäßige Verteilung aufweisen. Das heißt, die Anzahl der Zeilen wird folgendermaßen abgeschätzt:

  ```
  CARD - NUM_FREQ_ROWS
  --------------------          (2)
     COLCARD - N
  ```

Dabei ist CARD die Anzahl von Zeilen in der Tabelle, COLCARD ist die Kardinalität der Spalte und COLCARD ist die Gesamtanzahl von Zeilen mit einem Wert, der gleich einem der N häufigsten Werte ist.

Betrachten Sie zum Beispiel eine Spalte (`C1`), für die sich die Häufigkeit der Datenwerte folgendermaßen darstellt:

Datenwert	Häufigkeit
1	2
2	3
3	40
4	4
5	1

Wenn Häufigkeitsstatistiken nur für die häufigsten Werte (d. h. N = 1) dieser Spalte verfügbar sind, ist die Anzahl der Zeilen in der Tabelle gleich 50 und die Spaltenkardinalität gleich 5. Das Vergleichselement `C1 = 3` wird exakt von 40 Zeilen erfüllt. Wenn das Optimierungsprogramm annimmt, dass die Datenwerte gleichmäßig verteilt sind, schätzt es die Anzahl der Zeilen, die das Vergleichselement erfüllen mit 50/5 = 10 ab, was einen Schätzfehler von -75% bedeutet. Wenn das Optimierungsprogramm auf Häufigkeitsstatistiken zurückgreifen kann, wird die Anzahl der Zeilen mit 40 abgeschätzt und es entsteht in diesem Fall kein Fehler.

Betrachten Sie ein anderes Beispiel, in dem zwei Zeilen das Vergleichselement `C1 = 1` erfüllen. Ohne Häufigkeitsstatistiken wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, auf 10 geschätzt. Mithin entsteht ein Fehler von 400 %. Die folgende Formel kann zur Berechnung des Schätzfehlers (in Prozent) herangezogen werden:

```
geschätzte Zeilen - tatsächliche Zeilen
---------------------------------------  X 100
          tatsächliche Zeilen
```

Bei Verwendung der Häufigkeitsstatistik (N = 1) schätzt das Optimierungsprogramm die Anzahl der Zeilen, die diesen Wert enthalten, anhand der oben gezeigten Formel (2). Zum Beispiel:

```
(50 - 40)
--------- = 3
 (5 - 1)
```

Der Fehler wird dabei um eine Größenordnung verringert, wie folgende Berechnung zeigt:

```
 3 - 2
------- = 50%
   2
```

Beispiel für Quantilstatistiken

In den folgenden Erläuterungen der Quantilstatistiken wird der Begriff „K-Quantile" verwendet. Das *K-Quantil* für eine Spalte ist der kleinste Datenwert `V`, sodass mindestens „K" Zeilen Datenwerte enthalten, die kleiner oder gleich `V` sind. Ein K-Quantil lässt sich berechnen, indem die Zeilen in der Spalte nach aufsteigenden Datenwerten sortiert werden. Das K-Quantil ist der Datenwert in der K-ten Zeile der sortierten Spalte.

Wenn Quantilstatistiken verfügbar sind, kann das Optimierungsprogramm die Anzahl von Zeilen besser abschätzen, die ein Bereichsvergleichselement erfüllen, wie in folgenden Beispielen veranschaulicht wird. Betrachten Sie eine Spalte (`C`), die folgende Werte enthält:

C
0,0
5,1
6,3
7,1
8,2
8,4
8,5
9,1
93,6
100,0

Nehmen Sie an, dass K-Quantile für K = 1, 4, 7 und 10 wie folgt zur Verfügung stehen:

K	K-Quantil
1	0,0
4	7,1
7	8,5
10	100,0

Betrachten Sie zunächst das Vergleichselement `C <= 8,5`. Für die oben angegebenen Daten erfüllen exakt sieben Zeilen dieses Vergleichselement. Unter der Annahme einer gleichmäßigen Datenverteilung und unter Verwendung der oben angegeben Formel (1), wobei KEY1 durch LOW2KEY ersetzt wird, wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, folgendermaßen abgeschätzt:

```
 8,5 - 5,1
---------- x 10 *= 0
93,6 - 5,1
```

Die Notation `*=` bedeutet „annähernd gleich". Der Fehler bei dieser Schätzung ist annähernd -100%.

Wenn Quantilstatistiken verfügbar sind, schätzt das Optimierungsprogramm die Anzahl der Zeilen, die das gleiche Vergleichselement (`C <= 8,5`) erfüllen ab, indem es 8,5 als höchsten Wert eines der K-Quantile feststellt und die Anzahl der Zeilen ermittelt, indem es den entsprechenden Wert von K, also 7, verwendet. In diesem Fall reduziert sich der Fehler auf 0.

Betrachten Sie nun das Vergleichselement `C <= 10`. Dieses Vergleichselement wird von exakt acht Zeilen erfüllt. Wenn das Optimierungsprogramm von einer gleichmäßigen Datenwertverteilung ausgehen und Formel (1) verwenden muss, schätzt es die Anzahl von Zeilen, die das Vergleichselement erfüllen, auf 1, was einen Fehler von -87,5% ergibt.

Im Unterschied zum vorigen Beispiel ist der Wert 10 keiner der gespeicherten K-Quantile. Allerdings kann das Optimierungsprogramm mithilfe von Quantilen die Anzahl von Zeilen, die das Vergleichselement erfüllen, als `r_1 + r_2` abschätzen, wobei `r_1` die Anzahl von Zeilen ist, die das Vergleichselement `C <= 8,5` erfüllen, und `r_2` die Anzahl von Zeilen ist, die das Vergleichselement `C > 8,5 AND C`

<= 10 erfüllen. Wie im vorigen Beispiel gilt r_1 = 7. Zur Abschätzung von r_2 verwendet das Optimierungsprogramm die lineare Interpolation:

```
        10 - 8,5
r_2 *= ---------- x (Anzahl Zeilen mit Wert > 8,5 und <= 100,0)
       100 - 8,5
       10 - 8,5
r_2 *= ---------- x (10 - 7)
       100 - 8,5
       1,5
r_2 *= ---- x (3)
       91,5
r_2 *= 0
```

Die abschließende Schätzung ist r_1 + r_2 *= 7, und der Fehler beträgt nur -12,5 %.

Quantile verbessern die Schätzgenauigkeit in den obigen Beispielen deswegen, weil die realen Datenwerte „Häufungen" im Bereich von 5 bis 10 bilden, aber die Standardformeln zur Schätzung von einer gleichmäßigen Verteilung der Werte zwischen 0 und 100 ausgehen.

Die Verwendung von Quantilen erhöht auch die Genauigkeit, wenn es wesentliche Unterschiede in den Häufigkeiten verschiedener Datenwerte gibt. Betrachten Sie eine Spalte, die Datenwerte mit den folgenden Häufigkeiten enthält:

Datenwert	**Häufigkeit**
20	5
30	5
40	15
50	50
60	15
70	5
80	5

Angenommen, es stehen K-Quantile zur Verfügung für K = 5, 25, 75, 95 und 100:

K	**K-Quantil**
5	20
25	40
75	50
95	70
100	80

Nehmen Sie außerdem an, dass statistische Häufigkeitsdaten für die 3 häufigsten Werte verfügbar sind.

Betrachten Sie das Vergleichselement C BETWEEN 20 AND 30. An der Verteilung der Datenwerte können Sie sehen, dass genau 10 Zeilen das Vergleichselement erfüllen. Unter der Annahme einer gleichmäßigen Datenverteilung und unter Verwendung der Formel (1) wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, wie folgt abgeschätzt:

```
30 - 20
-------  x 100 = 25
70 - 30
```

Diese Abschätzung enthält einen Fehler von 150%.

Unter Verwendung der Häufigkeitsstatistik und der Quantile wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, als r_1 + r_2 abgeschätzt, wobei r_1 die Anzahl der Zeilen ist, die das Vergleichselement C = 20 erfüllen, und r_2 die Anzahl der Zeilen ist, die das Vergleichselement C > 20 AND C <= 30 erfüllen. Bei Verwendung der Formel (2) wird r_1 folgendermaßen abgeschätzt:

```
100 - 80
-------- = 5
  7 - 3
```

r_2 wird mit linearer Interpolation folgendermaßen abgeschätzt:

```
    30 - 20
    ------- x (Anzahl Zeilen mit Wert > 20 und <= 40)
    40 - 20
    30 - 20
= ------- x (25 - 5)
    40 - 20
= 10,
```

Als endgültiger Schätzwert ergibt sich 15, wodurch der Schätzfehler um den Faktor 3 verringert wird.

Detaillierte Indexstatistiken

Wenn Sie RUNSTATS für Indizes mit der Klausel DETAILED ausführen, erfassen Sie statistische Informationen zu Indizes, mit deren Hilfe das Optimierungsprogramm abschätzen kann, wie viele Datenseitenabrufe für die unterschiedlichen Pufferpoolgrößen erforderlich würden. Diese Zusatzinformationen unterstützen das Optimierungsprogramm bei einer besseren Abschätzung des Aufwands für den Zugriff auf eine Tabelle über einen Index.

Anmerkung: Wenn Sie detaillierte Indexstatistiken erfassen, benötigt RUNSTATS mehr Zeit und erfordert mehr Speicher und CPU-Kapazitäten. Für die Option SAMPLED DETAILED, bei der eine Berechnung von Daten nur für eine statistisch relevante Anzahl von Einträgen erfolgt, sind 2 MB Statistikzwischenspeicher erforderlich. Ordnen Sie wegen dieses zusätzlichen Speicherbedarfs dem Datenbankkonfigurationsparameter *stat_heap_sz* weitere 488 4-KB-Seiten zu. Wenn der Zwischenspeicher zu klein erscheint, gibt RUNSTATS einen Fehler zurück, bevor das Dienstprogramm versucht, Statistikdaten zu erfassen.

Die DETAILED-Statistiken PAGE_FETCH_PAIRS und CLUSTERFACTOR werden nur erfasst, wenn die Tabelle eine ausreichende Größe hat (ca. 25 Seiten). In diesem Fall hat CLUSTERFACTOR einen Wert zwischen 0 und 1, während CLUSTERRATIO den Wert -1 (nicht erfasst) hat. Für Tabellen, die kleiner als 25 Seiten sind, hat CLUSTERFACTOR den Wert -1 (nicht erfasst) und CLUSTERRATIO einen Wert zwischen 0 und 100, auch wenn die Klausel DETAILED für einen Index für diese Tabelle angegeben wird.

Mithilfe der Klausel DETAILED werden komprimierte Informationen über die Anzahl der physischen E/A-Operationen bereitgestellt, die für den Zugriff auf die Datenseiten einer Tabelle erforderlich werden, wenn eine vollständige Indexsuche bei verschiedenen Puffergrößen ausgeführt wird. Das Dienstprogramm RUNSTATS sucht die Seiten des Index ab und modelliert dabei die verschiedenen Puffergrößen

und sammelt Schätzwerte darüber, wie häufig eine Fehlseitenbedingung auftritt. Wenn zum Beispiel nur eine Pufferseite verfügbar ist, führt jede neue Seite, auf die der Index verweist, zu einer Fehlseite. In einem ungünstigeren Fall könnte jede Zeile auf eine andere Seite verweisen, was zur gleichen Anzahl von E/A-Operationen wie Zeilen in der indexierten Tabelle führt. Im entgegengesetzten Extremfall, wenn der Puffer groß genug ist, um die gesamte Tabelle (abhängig von der maximalen Puffergröße) aufzunehmen, werden alle Tabellenseiten nur einmal gelesen. Die Anzahl der physischen E/A-Operationen ist infolgedessen eine monotone, nicht steigende Funktion der Puffergröße.

Die statistischen Informationen ermöglichen außerdem feinere Schätzwerte für den Grad der Clusterbildung der Tabellenzeilen in Bezug auf die Indexreihenfolge. Je geringer die Clusterbildung der Tabellenzeilen in Relation zum Index ist, desto mehr E/A-Operationen sind für den Zugriff auf Tabellenzeilen über den Index erforderlich. Das Optimierungsprogramm zieht sowohl die Puffergröße als auch den Grad der Clusterbildung bei der Abschätzung des Aufwands für den Zugriff auf eine Tabelle über einen Index in Betracht.

Die detaillierten Indexstatistiken sollten erfasst werden, wenn Abfragen auf Spalten zugreifen, die nicht im Index enthalten sind. Darüber hinaus sollten detaillierte Indexdaten in folgenden Fällen verwendet werden:

- Die Tabelle hat mehrere Indizes ohne Clustering mit verschiedenen Graden von Clusterbildung.
- Der Grad der Clusterbildung unter den Schlüsselwerten ist nicht gleichmäßig.
- Die Werte im Index werden ungleichmäßig aktualisiert.

Eine Beurteilung dieser Bedingungen ist ohne Vorwissen bzw. ohne eine zwangsweise durchgeführte Indexsuche unter verschiedenen Puffergrößen und eine Überwachung der sich ergebenden physischen E/A-Operationen schwierig. Die Methode des geringsten Aufwands zur Feststellung, ob eine dieser Situationen auftritt, besteht wahrscheinlich darin, die DETAILED-Statistikdaten für einen Index zu erfassen, zu untersuchen und zu behalten, wenn die resultierenden Wertepaare für PAGE_FETCH_PAIRS nicht linear sind.

Unterelementstatistiken

Wenn Tabellen Spalten enthalten, die durch Leerzeichen getrennte Unterfelder oder Unterelemente enthalten, und Abfragen auf diese Spalten in WHERE-Klauseln verweisen, sollten Sie Unterelementstatistiken erfassen, um die Auswahl der besten Zugriffspläne zu gewährleisten.

Nehmen Sie zum Beispiel an, eine Datenbank enthält eine Tabelle namens DOCUMENTS, deren Zeilen jeweils ein Dokument beschreiben, und nehmen Sie ferner an, DOCUMENTS enthält eine Spalte namens KEYWORDS, die eine Liste relevanter Schlüsselwörter für das Dokument enthält, die zum Abrufen von Texten verwendet werden. Die Spalte KEYWORDS könnte zum Beispiel folgende Werte enthalten:

```
'Datenbank Simulation Analytisch Business Intelligence'
'Simulation Modell Fruchtfliege Reproduktion Temperatur'
'Forstwirtschaft Fichte Boden Erosion Regen'
'Wald Temperatur Boden Niederschlag Brand'
```

In diesem Beispiel besteht jede Spalte aus 5 Unterelementen, von denen jedes ein Wort (das Schlüsselwort) ist und durch ein Leerzeichen von den anderen Wörtern getrennt ist.

Bei Abfragen, die für solche Spalten das Vergleichselement LIKE mit dem Universalplatzhalterzeichen % verwenden:

```
SELECT .... FROM DOCUMENTS WHERE KEYWORDS LIKE '%Simulation%'
```

ist es für das Optimierungsprogramm häufig von Nutzen, wenn es einige grundlegende Statistikdaten zur Unterelementstruktur der Spalte zur Verfügung hat.

Die folgenden Statistiken werden für Spalten der Typen CHAR und VARCHAR mit einem Codepageattribut eines Einzelbytezeichensatzes (SBCS), mit FOR BIT DATA oder mit UTF-8 erfasst, wenn Sie das Dienstprogramm RUNSTATS mit der Klausel LIKE STATISTICS ausführen:

SUB_COUNT
Die durchschnittliche Anzahl der Unterelemente.

SUB_DELIM_LENGTH
Die durchschnittliche Länge der einzelnen Begrenzer, die Unterelemente voneinander trennen. Unter einem Begrenzer ist in diesem Zusammenhang ein Leerzeichen oder mehrere aufeinander folgende Leerzeichen zu verstehen.

Im Beispiel der Spalte KEYWORDS hat SUB_COUNT den Wert 5, und SUB_DELIM_LENGTH den Wert 1, da jeder Begrenzer ein einzelnes Leerzeichen ist.

Die Registrierdatenbankvariable DB2_LIKE_VARCHAR beeinflusst die Art und Weise, wie das Optimierungsprogramm ein Vergleichselement des folgenden Formats behandelt:

```
COLUMN LIKE '%xxxxxx'
```

(Dabei steht *xxxxxx* für eine beliebige Zeichenfolge) Das heißt, ein beliebiges LIKE-Vergleichselement, dessen Suchwert mit einem Prozentzeichen (%) beginnt. (Der Suchwert kann, muss aber nicht mit % enden.) Solche Vergleichselemente werden in der Folge als LIKE-Vergleichselemente mit Platzhalter bezeichnet. Das Optimierungsprogramm muss für alle Vergleichselemente abschätzen, wie viele Zeilen dem Vergleichselement entsprechen. Bei LIKE-Vergleichselementen mit Platzhaltern nimmt das Optimierungsprogramm an, dass die verglichene Spalte (COLUMN) eine Reihe miteinander verketteter Elemente enthält, und schätzt die Länge jedes Elements ausgehend von der Länge der Zeichenfolge ohne führende und abschließende %-Zeichen ab.

Sie können die Werte der Unterelementstatistiken durch eine Abfrage der Tabelle SYSIBM.SYSCOLUMNS untersuchen. Beispiel:

```
select substr(NAME,1,16), SUB_COUNT, SUB_DELIM_LENGTH
   from sysibm.syscolumns where tbname = 'DOCUMENTS'
```

Anmerkung: Bei Verwendung der Klausel LIKE STATISTICS benötigt das Dienstprogramm eventuell mehr Zeit. Bei einer Tabelle mit fünf Zeichenspalten zum Beispiel kann die Ausführung von RUNSTATS zwischen 15 und 40 % länger dauern, wenn die Optionen DETAILED und DISTRIBUTION nicht verwendet werden. Wird die Option DETAILED oder DISTRIBUTION angegeben, ist der Mehraufwand prozentual geringer, auch wenn der Mehraufwand absolut betrachtet derselbe ist. Wenn Sie die Verwendung dieser Option beabsichtigen, sollten Sie zwischen diesem Mehraufwand und den Verbesserungen der Abfrageleistung abwägen.

Katalogstatistiken, die von Benutzern aktualisiert werden können

Statistiken für benutzerdefinierte Funktionen

Zur Erstellung statistischer Informationen für benutzerdefinierte Funktionen (UDFs) editieren Sie die Katalogsicht SYSSTAT.ROUTINES. Wenn UDF-Statistiken verfügbar sind, können sie vom Optimierungsprogramm zur Abschätzung des Aufwands für verschiedene Zugriffspläne verwendet werden. Das Dienstprogramm RUNSTATS erfasst keine Statistiken für benutzerdefinierte Funktionen. Sind keine Statistiken verfügbar, enthalten die Statistikspalten den Wert -1, und das Optimierungsprogramm verwendet Standardwerte, die von einer einfachen benutzerdefinierten Funktion ausgehen.

Die folgende Tabelle enthält Informationen zu den Statistikspalten, für die Sie Schätzwerte angeben können, um die Leistung zu verbessern:

Tabelle 61. Funktionsstatistiken (SYSCAT.ROUTINES und SYSSTAT.ROUTINES)

Statistik	Beschreibung
IOS_PER_INVOC	Geschätzte Anzahl der Schreib-/Leseanforderungen, die jedes Mal ausgeführt werden, wenn eine Funktion ausgeführt wird
INSTS_PER_INVOC	Geschätzte Anzahl der Maschinenanweisungen, die jedes Mal ausgeführt werden, wenn eine Funktion ausgeführt wird
IOS_PER_ARGBYTE	Geschätzte Anzahl der Schreib-/Leseanforderungen, die für jedes Eingabeargumentbyte ausgeführt werden
INSTS_PER_ARGBYTES	Geschätzte Anzahl der Maschinenanweisungen, die für jedes Eingabeargumentbyte ausgeführt werden
PERCENT_ARGBYTES	Geschätzter Durchschnittsprozentwert der Eingabeargumentbyte, die von der Funktion tatsächlich verarbeitet werden
INITIAL_IOS	Geschätzte Anzahl der Schreib-/Leseanforderungen, die nur beim ersten/letzten Aufruf der Funktion ausgeführt werden
INITIAL_INSTS	Geschätzte Anzahl der Maschinenanweisungen, die nur beim ersten/letzten Aufruf der Funktion ausgeführt werden
CARDINALITY	Geschätzte Anzahl von Zeilen, die von einer Tabellenfunktion generiert werden

Betrachten Sie zum Beispiel eine benutzerdefinierte Funktion (EU_SHOE), die eine amerikanische Schuhgröße in die entsprechende europäische Schuhgröße umwandelt. (Diese beiden Schuhgrößen könnten benutzerdefinierte Datentypen (UDTs) haben.) Für diese UDF könnten Sie die Statistikspalten mit folgenden Werten versehen:

- Für INSTS_PER_INVOC könnte als Wert die geschätzte Anzahl der Maschinenanweisungen festgelegt werden, die zu folgenden Operationen erforderlich sind:
  - Aufrufen der Funktion EU_SHOE
  - Initialisieren der Ausgabezeichenfolge

– Rückgabe des Ergebnisses
- Für INSTS_PER_ARGBYTE könnte als Wert die geschätzte Anzahl der Maschineninstruktionen festgelegt werden, die zur Umwandlung der Eingabezeichenfolge in die europäische Schuhgröße erforderlich ist.
- Der Wert für PERCENT_ARGBYTES könnte auf 100 gesetzt werden, was anzeigt, dass die gesamte Eingabezeichenfolge umzuwandeln ist.
- Die Werte für INITIAL_INSTS, IOS_PER_INVOC, IOS_PER_ARGBYTE und INITIAL_IOS könnten jeweils auf 0 gesetzt werden, da diese benutzerdefinierte Funktion lediglich Berechnungen ausführt.

PERCENT_ARGBYTES würde für eine Funktion verwendet, die nicht immer die gesamte Eingabezeichenfolge verarbeitet. Ein Beispiel wäre eine benutzerdefinierte Funktion (LOCATE), an die zwei Argumente als Eingabe übergeben werden und die die Anfangsposition des ersten Vorkommens des ersten Arguments innerhalb des zweiten Arguments als Ergebnis zurückliefert. Nehmen Sie an, dass die Länge des ersten Arguments ausreichend klein ist, um im Vergleich zum zweiten Argument kaum eine Rolle zu spielen, und im Durchschnitt 75% des zweiten Arguments zum Auffinden des ersten durchsucht werden. Aufgrund dieser Informationen sollte der Wert für PERCENT_ARGBYTES auf 75 gesetzt werden. Die obige Abschätzung eines Durchschnitts von 75% fußt dabei auf folgenden zusätzlichen Annahmen:

- In der Hälfte der Fälle wird das erste Argument gar nicht gefunden, d. h., das gesamte zweite Argument wird durchsucht.
- Die Wahrscheinlichkeit, dass das erste Argument innerhalb des zweiten Arguments auftritt, ist an allen Stellen gleich groß, d. h., in den Fällen, in denen das erste Argument überhaupt gefunden wird, muss im Durchschnitt die Hälfte des zweiten Arguments durchsucht werden.

Sie können die Spalten INITIAL_INSTS bzw. INITIAL_IOS zur Eintragung der geschätzten Anzahl von Maschineninstruktionen bzw. Schreib-/Leseanforderungen verwenden, die nur beim ersten bzw. letzten Aufruf der Funktion ausgeführt werden, um zum Beispiel den Aufwand zur Einrichtung eines Arbeitspufferbereichs zu vermerken.

Informationen über Ein-/Ausgaben und Instruktionen, die von einer benutzerdefinierten Funktion verursacht werden, erhalten Sie über die Ausgaben des Compilers für die Programmiersprache bzw. der vom Betriebssystem bereitgestellten Überwachungsprogramme.

Katalogstatistiken zu Modellierung und Fallstudien

Sie können die statistischen Informationen in den Systemkatalogen abweichend vom tatsächlichen Status von Tabellen und Indizes ändern, um verschiedene mögliche Änderungen an der Datenbank zu Planungszwecken zu untersuchen. Aufgrund dieser Aktualisierbarkeit ausgewählter Systemkatalogstatistiken haben Sie folgende Möglichkeiten:

- Sie können die Abfrageleistung auf einem Entwicklungssystem unter Verwendung wirklichkeitsnaher Systemstatistiken eines geschäftlich genutzten Systems nachmodellieren.
- Sie können Fallstudien („Was wäre, wenn?“) für die Abfrageleistung durchführen und die Ergebnisse analysieren.

Nehmen Sie keine manuellen Aktualisierungen an den Statistiken eines Produktionssystems vor. Anderenfalls wählt das Optimierungsprogramm

möglicherweise nicht den besten Zugriffsplan für Abfragen in der Produktionsumgebung aus, die dynamische SQL- oder XQuery-Anweisungen enthalten.

Voraussetzungen

Sie müssen über eine explizite DBADM-Berechtigung für die Datenbank verfügen, um Statistiken für Tabellen und Indizes und ihre Komponenten modifizieren zu können. Das heißt, Ihre Benutzer-ID muss in der Tabelle SYSCAT.DBAUTH mit der Berechtigung DBADM eingetragen sein. Durch die Zugehörigkeit zu einer DBADM-Gruppe wird diese Berechtigung nicht explizit erteilt. Ein DBADM-Benutzer kann die Statistikzeilen für alle Benutzer abrufen und kann SQL-Anweisungen UPDATE an den für das Schema SYSSTAT definierten Sichten ausführen, um die Werte dieser Statistikspalten zu aktualisieren.

Ein Benutzer ohne DBADM-Berechtigung kann nur die Zeilen sehen, die statistische Daten für Objekte enthalten, für die er das Zugriffsrecht CONTROL hat. Wenn Sie keine DBADM-Berechtigung besitzen, können Sie die Statistiken für einzelne Datenbankobjekte ändern, wenn Sie die folgenden Zugriffsrechte für jedes Objekt haben:

- Explizites Zugriffsrecht CONTROL für Tabellen. Sie können auch die Statistiken für Spalten und Indizes dieser Tabellen aktualisieren.
- Explizites Zugriffsrecht CONTROL für Kurznamen in einem System föderierter Datenbanken. Sie können auch Statistiken für Spalten und Indizes für diese Kurznamen aktualisieren. Beachten Sie, dass die Aktualisierung nur lokale Metadaten betrifft (Tabellenstatistiken der Datenquelle werden nicht geändert). Diese Aktualisierungen betreffen nur die globale Zugriffsstrategie, die vom DB2-Optimierungsprogramm generiert wird.
- Eigentumsrecht für benutzerdefinierte Funktionen (UDFs).

Das folgende Beispiel zeigt, wie die Tabellenstatistik für die Tabelle EMPLOYEE aktualisiert werden kann:

```
UPDATE SYSSTAT.TABLES
SET  CARD   = 10000,
     NPAGES = 1000,
     FPAGES = 1000,
     OVERFLOW = 2
WHERE TABSCHEMA = 'benutzer-id'
  AND TABNAME   = 'EMPLOYEE'
```

Bei der manuellen Aktualisierung von Katalogstatistiken ist Vorsicht geboten. Willkürliche Änderungen können die Leistung nachfolgender Abfragen ernsthaft beeinflussen. Auch in einer nicht in der Produktion eingesetzten Datenbank, die Sie zum Testen oder Modellieren verwenden, können Sie jede der folgenden Methoden anwenden, um Aktualisierungen, die Sie auf diese Tabellen angewendet haben, zu erneuern und die Statistiken in einen konsistenten Status zu bringen:

- Rollback der UOW (Unit of Work, Arbeitseinheit), in der Änderungen durchgeführt wurden (unter der Annahme, dass die UOW noch nicht mit COMMIT festgeschrieben wurde).
- Ausführen des Dienstprogramms RUNSTATS, um die Katalogstatistiken neu berechnen und aktualisieren zu lassen.
- Aktualisieren der Katalogstatistiken, um anzugeben, dass keine Statistikdaten gesammelt wurden. (Zum Beispiel wird durch den Wert -1 in der Spalte NPAGES angezeigt, dass keine Statistik zur Anzahl von Seiten erfasst wurde.)

- Ersetzen der Katalogstatistiken durch die Daten, die sie enthielten, bevor Änderungen durchgeführt wurden. Diese Methode ist nur möglich, wenn Sie mit dem Tool *db2look* die Statistiken erfasst haben, bevor Sie Änderungen vorgenommen haben.

In einigen Fällen kann es vorkommen, dass das Optimierungsprogramm einen bestimmten statistischen Wert oder eine Kombination von Werten als ungültig erkennt. In diesem Fall verwendet es die Standardwerte und gibt eine Warnung aus. Situationen dieser Art sind jedoch selten, da der Hauptteil der Gültigkeitsprüfungen bei der Aktualisierung der Statistiken durchgeführt wird.

Statistiken zur Modellierung von Produktionsdatenbanken

Manchmal ist es wünschenswert, auf einem Testsystem einen Teil der Daten eines tatsächlich geschäftlich genutzten Systems nachzubilden. Jedoch sind die Zugriffspläne, die auf einem solchen Testsystem ausgewählt werden, nicht unbedingt dieselben wie die, die auf dem tatsächlich genutzten System gewählt würden, sofern nicht die Katalogstatistiken und die Konfigurationsparameter auf dem Testsystem so aktualisiert werden, dass sie mit denen auf dem Produktionssystem übereinstimmen.

Das Tool *db2look* kann für die Produktionsdatenbank ausgeführt werden, um die Aktualisierungsanweisungen (UPDATE-Anweisungen) zu generieren, die erforderlich sind, um die Katalogstatistiken der Testdatenbank in Übereinstimmung mit denen der Produktionsdatenbank zu bringen. Diese Aktualisierungsanweisungen können mithilfe des Programms *db2look* mit der Option `-m` (mimic-Modus) generiert werden. In diesem Fall generiert das Tool *db2look* ein Befehlsprozessorscript, das alle Anweisungen enthält, die erforderlich sind, um die Katalogstatistiken der Produktionsdatenbank nachzubilden. Dies kann bei der Analyse von SQL- oder XQuery-Anweisungen mithilfe von Visual Explain in einer Testumgebung nützlich sein.

Sie können Datenbankdatenobjekte einschließlich Tabellen, Sichten, Indizes und andere Objekte in einer Datenbank wieder erstellen, indem Sie die DDL-Anweisungen mit dem Befehl *db2look -e* extrahieren. Sie können das Befehlsprozessorscript, das mit diesem Befehl erstellt wurde, in einer anderen Datenbank ausführen, um die Datenbank neu zu erstellen. Sie können die Option `-e` und die Option `-m` zusammen in einem Script verwenden, das die Datenbank neu erstellt und die Statistikdaten festlegt.

Nach der Ausführung der von `db2look` erstellten Aktualisierungsanweisungen im Testsystem kann das Testsystem zur Prüfung der Zugriffspläne verwendet werden, die in der Produktionsdatenbank generiert werden sollen. Da das Optimierungsprogramm die Art und Konfiguration der Tabellenbereiche zur Abschätzung der E/A-Aufwände verwendet, muss das Testsystem über dieselbe Tabellenbereichsanordnung bzw. -konfiguration verfügen. Das heißt, sie muss dieselbe Anzahl von Containern desselben Typs, SMS oder DMS, haben.

Das Tool *db2look* befindet sich im Unterverzeichnis *bin*.

Weitere Informationen zur Verwendung dieses Tools erhalten Sie, wenn Sie folgenden Befehl in eine Befehlszeile eingeben:

```
db2look -h
```

Darüber hinaus stellt die Steuerzentrale eine Schnittstelle zum Dienstprogramm *db2look* mit dem Namen „DDL generieren" bereit. Die Steuerzentrale ermöglicht

eine Integration der Ergebnisdatei aus dem Dienstprogramm in die Scriptzentrale. Sie können außerdem den Befehl *db2look* über die Steuerzentrale zeitlich terminieren. Ein Unterschied bei der Verwendung der Steuerzentrale besteht darin, dass eine Analyse nur einer Tabelle durchgeführt werden kann, während in einem Aufruf des Befehls *db2look* bis zu 30 Tabellen analysiert werden können. Beachten Sie darüber hinaus, dass LaTex- und Graphical-Ausgaben über die Steuerzentrale nicht unterstützt werden.

Sie können das Dienstprogramm `db2look` auch für eine OS/390- oder z/OS-Datenbank ausführen. Das Dienstprogramm `db2look` extrahiert die DDL- und UPDATE-Anweisungen für die Statistiken von OS/390-Objekten. Diese Funktion ist sehr nützlich, wenn Sie OS/390- oder z/OS-Objekte extrahieren und in DB2 Database für Linux, UNIX und Windows neu erstellen möchten.

Es gibt einige Unterschiede zwischen den Statistikdaten für DB2 Database für Linux, UNIX und Windows und den Statistikdaten für OS/390. Das Dienstprogramm `db2look` führt die entsprechenden Umsetzungen von DB2 für OS/390 oder z/OS in DB2 Database für Linux, UNIX und Windows aus, wenn dies zutreffend ist, und setzt die Statistikdaten von DB2 Database für Linux, UNIX und Windows, für die in DB2 für OS/390 keine Entsprechung vorhanden ist, auf den Standardwert (-1). Im Folgenden wird beschrieben, wie das Dienstprogramm `db2look` die Statistikdaten von DB2 für OS/390 oder z/OS den Statistikdaten von DB2 Database für Linux, UNIX und Windows zuordnet. In den nachfolgenden Erläuterungen steht jeweils „UDB_x" für eine Statistikdatenspalte von DB2 Database für Linux, UNIX und Windows; „S390_x" steht für eine Statistikdatenspalte von DB2 für OS/390 oder z/OS.

1. Statistikdaten auf Tabellenebene.

   UDB_CARD = S390_CARDF
   UDB_NPAGES = S390_NPAGES

   Es gibt keinen Parameter S390_FPAGES. DB2 für OS/390 oder z/OS verfügt jedoch über einen anderen Parameter für Statistikdaten mit dem Namen PCTPAGES, der den Prozentsatz von aktiven Tabellenbereichsseiten darstellt, die Zeilen der Tabelle enthalten. Daher kann der Wert des Parameters UDB_FPAGES auf der Basis von S390_NPAGES und S390_PCTPAGES wie folgt berechnet werden:

   ```
   UDB_FPAGES=(S390_NPAGES * 100)/S390_PCTPAGES
   ```

   Es gibt keinen Parameter S390_OVERFLOW, der UDB_OVERFLOW zugeordnet werden kann. Daher wird dieser Wert vom Dienstprogramm `db2look` einfach auf den Standardwert gesetzt:

   ```
   UDB_OVERFLOW=-1
   ```

2. Statistikdaten auf Spaltenebene.

   UDB_COLCARD = S390_COLCARDF
   UDB_HIGH2KEY = S390_HIGH2KEY
   UDB_LOW2KEY = S390_LOW2KEY

   Es gibt keinen Parameter S390_AVGCOLLEN, der UDB_AVGCOLLEN zugeordnet werden kann. Daher wird dieser Wert vom Dienstprogramm `db2look` einfach auf den Standardwert gesetzt:

   ```
   UDB_AVGCOLLEN=-1
   ```

3. Statistikdaten auf Indexebene.

   UDB_NLEAF = S390_NLEAF
   UDB_NLEVELS = S390_NLEVELS

UDB_FIRSTKEYCARD= S390_FIRSTKEYCARD
UDB_FULLKEYCARD = S390_FULLKEYCARD
UDB_CLUSTERRATIO= S390_CLUSTERRATIO

Die anderen Statistikdaten, für die es keine OS/390- oder z/OS-Entsprechungen gibt, werden einfach auf den Standardwert gesetzt. Das heißt:

```
UDB_FIRST2KEYCARD = -1
UDB_FIRST3KEYCARD = -1
UDB_FIRST4KEYCARD  = -1
UDB_CLUSTERFACTOR = -1
UDB_SEQUENTIAL_PAGES = -1
UDB_DENSITY  = -1
```

4. Spaltenverteilungsstatistiken.

   In der Tabelle SYSIBM.SYSCOLUMNS unter DB2 für OS/390 oder z/OS gibt es zwei Typen von Statistikdaten. Der Typ „F" für häufige (Frequent) Werte und der Typ „C" für Kardinalität (Cardinality). Nur Einträge des Typs „F" gelten für DB2 Database für Linux, UNIX und Windows. Dies sind auch die Einträge, die berücksichtigt werden.

   ```
   UDB_COLVALUE = S390_COLVALUE
   UDB_VALCOUNT = S390_FrequencyF * S390_CARD
   ```

   Außerdem gibt es in der Tabelle SYSIBM.SYSCOLUMNS unter DB2 für OS/390 keine Spalte SEQNO. fDa eine solche Spalte für DB2 erforderlich ist, generiert das Dienstprogramm `db2looek` eine solche Spalte automatisch.

Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken

Die wichtigste allgemeine Regel, die bei einer Aktualisierung der Katalogstatistik zu beachten ist, ist die, dass gültige Werte, Wertebereiche und Formate der verschiedenen Statistikdaten in den Sichten für die Statistikdaten gespeichert werden. Darüber hinaus muss die Konsistenz der Beziehungen zwischen verschiedenen Statistiken gewahrt bleiben.

Zum Beispiel muss der Wert für COLCARD in der Sicht SYSSTAT.COLUMNS kleiner sein als der für CARD in der Sicht SYSSTAT.TABLES (die Anzahl der unterschiedlichen Werte in einer Spalte darf die Anzahl der Zeilen nicht überschreiten). Nehmen Sie an, Sie wollen den Wert für COLCARD von 100 auf 25 und den Wert für CARD von 200 auf 50 verringern. Wenn Sie die Sicht SYSSTAT.TABLES zuerst aktualisieren, sollten Sie eine Fehlernachricht empfangen (da CARD kleiner als COLCARD würde). Die richtige Reihenfolge wäre, zuerst den Wert für COLCARD in der Sicht SYSSTAT.COLUMNS und anschließend den Wert für CARD in der Sicht SYSSTAT.TABLES zu aktualisieren. Der Fall stellt sich umgekehrt dar, wenn Sie den Wert von COLCARD von 100 auf 250 und den Wert für CARD von 200 auf 300 erhöhen wollen. In diesem Fall müssten Sie zuerst den Wert CARD und anschließend den Wert COLCARD aktualisieren.

Wenn ein Konflikt zwischen einer aktualisierten Statistik und einer anderen Statistik festgestellt wird, wird eine Fehlernachricht ausgegeben. Jedoch werden vielleicht nicht immer Fehler gemeldet, wenn Konflikte auftreten. In einigen Fällen können die Konflikte nur schwer festgestellt und als Fehler gemeldet werden, besonders wenn die beiden zusammengehörigen Statistiken in verschiedenen Katalogen gespeichert sind. Aus diesem Grund sollten Sie solche Konflikte umsichtig vermeiden.

Die folgenden allgemeinen Prüfungen sollten vor der Aktualisierung einer Katalogstatistik durchgeführt werden:

1. Numerische Statistikdaten müssen -1 bzw. größer oder gleich null (0) sein.

2. Numerische Statistikdaten, die Prozentwerte darstellen (z. B. CLUSTERRATIO in der Katalogsicht SYSSTAT.INDEXES), müssen zwischen 0 und 100 liegen.

Anmerkung: Bei Zeilentypen sind die statistischen Daten auf Tabellenebene für NPAGES, FPAGES und OVERFLOW für eine untergeordnete Tabelle nicht aktualisierbar.

Wenn eine Tabelle erstmals erstellt wird, werden die Systemkatalogstatistikdaten auf -1 gesetzt; dies bedeutet, dass die Tabelle über keine Statistikdaten verfügt. DB2 verwendet für die Kompilierung und Optimierung von SQL- oder XQuery-Anweisungen Standardwerte, bis die Statistikdaten erfasst worden sind. Die Aktualisierung der Tabelle oder der Indexstatistik schlägt möglicherweise fehl, wenn die neuen Werte mit den Standardwerten nicht übereinstimmen. Aus diesem Grund ist es empfehlenswert, RUNSTATS nach der Erstellung einer Tabelle auszuführen, bevor Sie die Tabelle oder Indexstatistik aktualisieren.

Regeln zur manuellen Aktualisierung von Spaltenstatistiken

Bei der Aktualisierung von Statistikdaten in der Katalogsicht SYSSTAT.COLUMNS sind folgende Regeln zu beachten.

- Folgen Sie bei der manuellen Aktualisierung der Werte für die Spalten HIGH2KEY und LOW2KEY in der Sicht SYSSTAT.COLUMNS der Funktionsweise der generierten Werte:
  - Die Werte für HIGH2KEY und LOW2KEY müssen gültige Werte für den Datentyp der entsprechenden Benutzerspalte sein.
  - Die Länge der Werte für HIGH2KEY, LOW2KEY muss entweder 33 oder die maximale Länge des Datentyps der Zielspalte betragen, je nachdem, welcher der beiden Werte kleiner ist. Dies schließt zusätzliche Anführungszeichen nicht mit ein, durch die die Länge der Zeichenfolge auf 68 anwachsen kann. Dies bedeutet, dass nur die ersten 33 Zeichen des Werts in der entsprechenden Benutzerspalte bei der Bestimmung der Werte für HIGH2KEY, LOW2KEY in Betracht gezogen werden.
  - Die HIGH2KEY/LOW2KEY-Werte werden so gespeichert, dass sie in der SET-Klausel einer UPDATE-Anweisung und ohne Manipulation an Aufwandsberechnungen verwendet werden können. Für Zeichenfolgen bedeutet dies, dass einfache Anführungszeichen am Anfang und am Ende der Zeichenfolge angefügt und ein zusätzliches Anführungszeichen für jedes bereits in der Zeichenfolge vorhandene Anführungszeichen hinzugefügt wird. Beispiele von Benutzerspaltenwerten und ihren entsprechenden Werten für HIGH2KEY,LOW2KEY finden Sie in der folgenden Tabelle.

Tabelle 62. HIGH2KEY- und LOW2KEY-Werte für Datentypen

Datentyp in Benutzerspalte	Benutzerdaten	Entsprechender HIGH2KEY-, LOW2KEY-Wert
INTEGER	-12	-12
CHAR	abc	'abc'
CHAR	ab'c	'ab''c'

  - Der Wert für HIGH2KEY sollte größer als der Wert für LOW2KEY sein, wenn es mehr als drei unterschiedliche Werte in der entsprechenden Spalte gibt.

- Die Kardinalität einer Spalte (Statistik COLCARD in der Katalogsicht SYSSTAT-.COLUMNS) darf nicht größer sein als die Kardinalität der zugehörigen Tabelle oder statistischen Sicht (Statistik CARD in der Katalogsicht SYSSTAT.TABLES).
- Die Anzahl der Nullwerte in einer Spalte (Statistik NUMNULLS in der Katalogsicht SYSSTAT.COLUMNS) darf nicht größer sein als die Kardinalität der zugehörigen Tabelle oder statistischen Sicht (Statistik CARD in der Katalogsicht SYSSTAT.TABLES).
- Für Spalten mit folgenden Datentypen werden keine Statistikdaten unterstützt: LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB.

Regeln zur manuellen Aktualisierung von Verteilungsstatistiken

Manuelle Aktualisierungen von Verteilungsstatistiken nehmen Sie nur zu dem Zweck vor, eine Produktionsdatenbank nachzumodellieren oder Fallstudien für eine künstlich konstruierte Datenbank durchzuführen. Nehmen Sie keine manuellen Aktualisierungen an den Verteilungsstatistiken einer Produktionsdatenbank vor.

Stellen Sie sicher, dass alle Statistiken im Katalog konsistent sind. Im Einzelnen müssen die Katalogeinträge für Häufigkeitsstatistiken und Quantildaten für jede Spalte die folgenden Bedingungen erfüllen:

- Häufigkeitsstatistiken (in der Katalogsicht SYSSTAT.COLDIST). Dazu gehören:
  - Die Werte in der Spalte VALCOUNT müssen bei steigenden Werten für SEQNO unverändert bleiben oder abnehmen.
  - Die Anzahl der Werte in der Spalte COLVALUE muss kleiner oder gleich der Anzahl der unterschiedlichen Werte in der Spalte sein. Diese Anzahl wird in der Spalte COLCARD der Katalogsicht SYSSTAT.COLUMNS gespeichert.
  - Die Summe der Werte in der Spalte VALCOUNT muss kleiner oder gleich der Anzahl der Zeilen in der Spalte sein. Diese Anzahl wird in der Spalte CARD der Katalogsicht SYSSTAT.TABLES gespeichert.
  - In der Regel sollten die Werte in der Spalte COLVALUE zwischen dem zweithöchsten und dem zweitniedrigsten Datenwert für die Spalte liegen. Diese beiden Werte werden in der Spalte HIGH2KEY bzw. LOW2KEY der Katalogsicht SYSSTAT.COLUMNS gespeichert. Es kann nur einen häufigen Wert geben, der größer als der Wert für HIGH2KEY ist, und einen, der kleiner als der Wert für LOW2KEY ist.
- Quantile (in der Katalogsicht SYSSTAT.COLDIST). Dazu gehören:
  - Die Werte in der Spalte COLVALUE müssen bei steigenden Werten für SEQNO unverändert bleiben oder abnehmen.
  - Die Werte in der Spalte VALCOUNT müssen bei steigenden Werten für SEQNO steigen.
  - Der größte Wert in der Spalte COLVALUE muss einen entsprechenden Eintrag in der Spalte VALCOUNT haben, der gleich der Anzahl von Zeilen in der Spalte ist.
  - In der Regel sollten die Werte in der Spalte COLVALUE zwischen dem zweithöchsten und dem zweitniedrigsten Datenwert für die Spalte liegen. Diese beiden Werte werden in der Spalte HIGH2KEY bzw. LOW2KEY der Katalogsicht SYSSTAT.COLUMNS gespeichert.

Nehmen Sie an, es stehen Verteilungsstatistikdaten für eine Spalte C1 mit „Z" Zeilen zur Verfügung, und Sie möchten die Statistiken so modifizieren, dass sie einer Spalte mit identischen relativen Proportionen der Datenwerte, aber mit „(F x Z)" Zeilen entsprechen. Um die Größenordnung der Häufigkeitsstatistiken um einen

Faktor F zu erhöhen, muss jeder Eintrag der Spalte VALCOUNT mit F multipliziert werden. Um die Quantilwerte ebenfalls um einen Faktor F zu vergrößern, muss jeder Eintrag der Spalte VALCOUNT mit F multipliziert werden. Wenn Sie diese Regeln nicht beachten, kann das Optimierungsprogramm den falschen Filterfaktor verwenden, was bei der Ausführung der Abfrage zu nicht vorhersehbaren Auswirkungen auf die Leistung führen kann.

Regeln zur manuellen Aktualisierung von Tabellen- und Kurznamenstatistiken

Die einzigen statistischen Werte, die Sie in der Katalogsicht SYSSTAT.TABLES aktualisieren können, sind CARD, FPAGES, NPAGES und OVERFLOW sowie für MDC-Tabellen ACTIVE_BLOCKS. Beachten Sie dabei Folgendes:

1. Der Wert für CARD der Tabelle muss größer als oder gleich allen auf diese Tabelle bezogenen Werten für COLCARD in der Katalogsicht SYSSTAT.COLUMNS sein.
2. Der Wert für CARD muss größer als der Wert für NPAGES sein.
3. Der Wert für FPAGES muss größer als der Wert für NPAGES sein.
4. Der Wert für NPAGES muss kleiner oder gleich jedem Wert für den Abrufteil der Wertepaare in der Spalte PAGE_FETCH_PAIRS für jeden Index sein (unter der Annahme, dass diese Statistik für den Index relevant ist).
5. Der Wert für CARD darf nicht kleiner oder gleich irgendeinem Wert für den Seitenabruf in den Wertepaaren der Spalte PAGE_FETCH_PAIRS für jeden Index sein (unter der Annahme, dass diese Statistik für den Index relevant ist).

Wenn Sie in einem System föderierter Datenbanken arbeiten, gehen Sie bei der Erstellung bzw. Aktualisierung von Statistiken für einen Kurznamen über eine ferne Sicht sehr vorsichtig vor. Die statistischen Informationen, wie zum Beispiel die Anzahl von Zeilen, die von diesem Kurznamen zurückgegeben werden, entsprechen möglicherweise nicht dem realen Aufwand zur Auswertung dieser Sicht und können das DB2-Optimierungsprogramm irreführen. Fälle, die von Aktualisierungen der Statistiken profitieren können, sind ferne Sichten, die für eine einzelne ferne Tabelle ohne Anwendung von Spaltenfunktionen in der SELECT-Liste definiert wurden. Komplexe Sichten erfordern möglicherweise einen komplexen Optimierungsprozess, der die Optimierung jeder einzelnen Abfrage erforderlich macht. Ziehen Sie stattdessen die Erstellung lokaler Sichten über Kurznamen in Betracht, damit das DB2-Optimierungsprogramm in der Lage ist, den Aufwand für die Sicht genauer abzuschätzen.

Regeln zur manuellen Aktualisierung von Indexstatistiken

Bei der Aktualisierung der Statistiken in der Katalogsicht SYSSTAT.INDEXES sind folgende Regeln zu beachten:

1. Die Werte für PAGE_FETCH_PAIRS (in der Katalogsicht SYSSTAT.INDEXES) müssen folgende Regeln erfüllen:
   - Einzelne Werte in der Statistik PAGE_FETCH_PAIRS müssen durch eine Folge von Leerzeichenbegrenzern getrennt werden.
   - Einzelne Werte in der Statistik PAGE_FETCH_PAIRS dürfen eine Länge von 10 Stellen nicht überschreiten und müssen kleiner als der größte ganzzahlige Wert (MAXINT = 2147483647) sein.
   - Es muss immer einen gültigen Wert für PAGE_FETCH_PAIRS geben, wenn für CLUSTERFACTOR ein Wert größer null (0) angegeben ist.
   - Es müssen sich genau 11 Wertepaare in einer einzelnen Statistik PAGE_FETCH_PAIR befinden.

- Die Einträge für die Puffergrößen in PAGE_FETCH_PAIRS müssen eine aufsteigende Wertefolge bilden.
- Kein Wert für die Puffergröße in einem Eintrag für PAGE_FETCH_PAIRS darf den Wert MIN(NPAGES, 524287) bei 32-Bit-Betriebssystemen bzw. MIN(NPAGES, 2147483647) bei 64-Bit-Betriebssystemen überschreiten, wobei NPAGES die Anzahl der Seiten in der entsprechenden Tabelle (in der Katalogsicht SYSSTAT.TABLES) ist.
- Einträge für „Seitenabrufe" in PAGE_FETCH_PAIRS müssen in absteigender Wertefolge angegeben werden, wobei kein einzelner Eintrag für „Seitenabrufe" kleiner als NPAGES sein darf. Einträge für „Seitenabrufe" in PAGE_FETCH_PAIRS können nicht größer als der Wert der Statistik CARD (Kardinalität) der zugehörigen Tabelle sein.
- Wenn der Wert für die Puffergröße in zwei aufeinander folgenden Paaren übereinstimmt, muss der Wert für den Seitenabruf (Page fetch) in beiden Paaren ebenfalls übereinstimmen (in der Katalogsicht SYSSTAT.TABLES).

Eine gültige Aktualisierung der Wertefolge für PAGE_FETCH_PAIRS ist zum Beispiel:

```
PAGE_FETCH_PAIRS =
 '100 380 120 360 140 340 160 330 180 320 200 310 220 305 240 300
  260 300 280 300 300 300'
```

Dabei gilt Folgendes:

```
NPAGES = 300
CARD   = 10000
CLUSTERRATIO  =  -1
CLUSTERFACTOR = 0.9
```

2. Die Werte für CLUSTERRATIO und CLUSTERFACTOR (in der Katalogsicht SYSSTAT.INDEXES) müssen folgende Regeln erfüllen:
   - Gültige Werte für CLUSTERRATIO (Clusterverhältnis) sind -1 bzw. Werte zwischen 0 und 100.
   - Gültige Werte für CLUSTERFACTOR (Clusterfaktor) sind -1 bzw. Werte zwischen 0 und 1.
   - Es muss immer mindestens einer der Werte für CLUSTERRATIO und CLUSTERFACTOR gleich -1 sein.
   - Wenn für CLUSTERFACTOR ein positiver Wert angegeben ist, müssen gültige Statistikdaten in der Spalte PAGE_FETCH_PAIR enthalten sein.
3. Bei relationalen Indizes gelten die folgenden Regeln für FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD, FULLKEYCARD und INDCARD:
   - Der Wert für FIRSTKEYCARD muss gleich dem Wert für FULLKEYCARD für einen einspaltigen Index sein.
   - Der Wert für FIRSTKEYCARD muss gleich dem Wert COLCARD (in SYSSTAT.COLUMNS) für die entsprechende Spalte sein.
   - Wenn einige dieser Indexstatistikwerte nicht relevant sind, sollten Sie sie auf den Wert -1 setzen. Wenn Sie beispielsweise einen Index mit nur drei Spalten haben, setzen Sie FIRST4KEYCARD auf den Wert -1.
   - Für mehrspaltige Indizes gilt die folgende Beziehung, wenn alle Statistikwerte relevant sind:

     ```
     FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
        <= FULLKEYCARD <= INDCARD == CARD
     ```

4. Bei Indizes für XML-Daten gelten die folgenden Regeln für FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD, FULLKEYCARD und INDCARD:
   - Die Beziehung muss wie folgt aussehen:

     ```
     FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
        <= FULLKEYCARD <= INDCARD
     ```
5. Die folgenden Regeln gelten für SEQUENTIAL_PAGES und DENSITY:
   - Gültige Werte für SEQUENTIAL_PAGES sind -1 bzw. Werte zwischen 0 und NLEAF.
   - Gültige Werte für DENSITY sind -1 bzw. Werte zwischen 0 und 100.

Kapitel 21. Routinen

Richtlinien für gespeicherte Prozeduren

Mithilfe gespeicherter Prozeduren kann durch einen Aufruf an eine ferne Datenbank eine vorprogrammierte Prozedur in einer Datenbankanwendungsumgebung ausgeführt werden, in der viele Situationen wiederholt auftreten. Zum Beispiel könnte beim Empfang einer festen Menge von Daten die Ausführung derselben Gruppe aus mehreren Anforderungen für eine Datenbank bzw. die Rückgabe einer festen Menge von Daten mehrere Zugriffe auf die Datenbank darstellen.

Die Verarbeitung einer einzelnen SQL-Anweisung für eine ferne Datenbank erfordert das Senden von zwei Übertragungen: eine Anforderungs- und eine Empfangsübertragung. Da eine Anwendung viele SQL-Anweisungen enthält, benötigt sie viele Übertragungen, um ihre Arbeit auszuführen.

Wenn ein IBM Data Server-Client jedoch eine gespeicherte Prozedur verwendet, die viele SQL-Anweisungen einbindet, benötigt er nur zwei Übertragungen für den gesamten Prozess.

Gespeicherte Prozeduren werden in der Regel in Prozessen ausgeführt, die von den Datenbankagenten getrennt sind. Diese Trennung erfordert, dass die Prozesse von gespeicherten Prozeduren und die Agentenprozesse über einen Router kommunizieren. Ein spezieller Typ von gespeicherter Prozedur, der im Agentenprozess ausgeführt wird, kann jedoch die Leistung verbessern, obwohl er ein erhebliches Risiko einer Daten- oder Datenbankbeschädigung birgt.

Diese riskanten gespeicherten Prozeduren sind diejenigen, die als *nicht abgeschirmt* (NOT FENCED) erstellt werden. Bei einer nicht abgeschirmten gespeicherten Prozedur gibt es keine Trennung zwischen der gespeicherten Prozedur und den Datenbanksteuerstrukturen, die vom Datenbankagenten verwendet werden. Wenn ein Datenbankadministrator (DBA) sicherstellen will, dass die Ausführung von gespeicherten Prozeduren die Datenbanksteuerstrukturen nicht versehentlich oder vorsätzlich beschädigt, darf er die Option *nicht abgeschirmt* nicht angeben.

Aufgrund des Risikos einer Datenbankbeschädigung sollten Sie *nicht abgeschirmte* gespeicherte Prozeduren **nur** verwenden, wenn es auf maximale Leistungsvorteile ankommt. Darüber hinaus müssen Sie absolut sicherstellen, dass die Prozedur korrekt codiert ist und gründlich getestet wurde, bevor Sie zulassen, dass sie als nicht abgeschirmte gespeicherte Prozedur ausgeführt wird. Wenn ein schwer wiegender Fehler auftritt, während eine nicht abgeschirmte gespeicherte Prozedur ausgeführt wird, stellt der Datenbankmanager fest, ob der Fehler im Anwendungscode oder im Datenbankmanagercode aufgetreten ist, und führt die entsprechende Recovery durch.

Eine nicht abgeschirmte gespeicherte Prozedur kann den Datenbankmanager so beschädigen, dass keine Recovery mehr möglich ist. Dies kann zu Datenverlusten und potenziell zu einer beschädigten Datenbank führen. Gehen Sie äußerst vorsichtig vor, wenn Sie nicht abgeschirmte, zuverlässige gespeicherte Prozeduren ausführen. In nahezu allen Fällen lässt sich durch eine geeignete Leistungsanalyse für eine Anwendung auch ohne Verwendung nicht abgeschirmter gespeicherter Prozeduren eine gute Leistung erzielen. Zum Beispiel könnte die Leistung mithilfe von Triggern verbessert werden.

Verbessern der Leistung von SQL-Prozeduren

Übersicht über die Kompilierung von SQL PL und Inline-SQL-PL durch DB2

Zur Erläuterung, wie sich die Leistung von SQL-Prozeduren verbessern lässt, soll zunächst in einer Übersicht dargestellt werden, wie DB2 solche Prozeduren nach der Ausführung der Anweisung CREATE PROCEDURE kompiliert.

Wenn eine SQL-Prozedur erstellt wird, trennt DB2 die SQL-Abfragen im Hauptteil der Prozedur von der Prozedurlogik ab. Zur Maximierung der Leistung werden die SQL-Abfragen statisch in Abschnitte in einem Paket kompiliert. Für eine statisch kompilierte Abfrage besteht ein Abschnitt im Wesentlichen aus dem Zugriffsplan, der vom DB2-Optimierungsprogramm für die betreffende Abfrage ausgewählt wurde. Ein Paket ist eine Sammlung von Abschnitten. Weitere Informationen zu Paketen und Abschnitten finden Sie im Handbuch DB2 SQL Reference. Die Prozedurlogik wird in eine dynamisch verlinkte Bibliothek (DLL) kompiliert.

Während der Ausführung einer Prozedur findet jedes Mal, wenn die Steuerung von der Prozedurlogik zu einer SQL-Anweisung übergeht, ein *Kontextwechsel* zwischen der DLL-Datei und der DB2-Steuerkomponente statt. Seit DB2 Version 8.1 werden SQL-Prozeduren im 'nicht abgeschirmten Modus' ausgeführt. Das bedeutet, dass sie im selben Adressraum wie die DB2-Steuerkomponente ausgeführt werden. Bei diesem Kontextwechsel handelt es sich jedoch nicht um einen vollständigen Kontextwechsel auf Betriebssystemebene, sondern vielmehr um einen Wechsel der Ebene innerhalb von DB2. Die Reduzierung von Kontextwechseln in Prozeduren, die sehr häufig aufgerufen werden, wie zum Beispiel Prozeduren in einer OLTP-Anwendung, oder die große Anzahlen von Zeilen verarbeiten, wie zum Beispiel Prozeduren, die Datenbereinigungsoperationen ausführen, kann eine spürbare Auswirkung auf die Prozedurleistung haben.

Während eine SQL-Prozedur, die SQL PL (SQL Procedural Language) enthält, implementiert wird, indem die einzelnen SQL-Abfragen der Prozedur statisch in Abschnitte in einem Paket kompiliert werden, wird eine Inline-SQL-PL-Funktion implementiert, wie der Name andeutet, indem der Hauptteil der Funktion 'inline' in die Abfrage integriert wird, die sie verwendet. Abfragen in SQL-Funktionen werden zusammen kompiliert, so als ob der Funktionsteil eine einzelne Abfrage wäre. Die Kompilierung erfolgt jedes Mal, wenn eine Anweisung, die diese Funktion verwendet, kompiliert wird. Im Unterschied zu den Abläufen in SQL-Prozeduren werden prozedurale Anweisungen in SQL-Funktionen nicht auf einer anderen Ebene als die Datenflussanweisungen ausgeführt. Daher findet kein Kontextwechsel statt, wenn die Steuerung von einer prozeduralen Anweisung zu einer Datenflussanweisung oder umgekehrt übergeht.

Sofern keine Nebenwirkungen durch Ihre Logik auftreten, verwenden Sie eine SQL-Funktion.

Aufgrund des Unterschieds in der Kompilierung zwischen SQL PL in Prozeduren und Inline-SQL-PL in Funktionen, ist in der Regel davon auszugehen, dass ein Stück prozeduralen Codes in einer Funktion schneller ausgeführt wird als in einer Prozedur, sofern der Code nur SQL-Daten abfragt und keine Datenänderungen vornimmt, das heißt, keine Nebenwirkungen auf die Daten in der Datenbank bzw. auf Daten außerhalb der Datenbank hat.

Dies ist jedoch nur dann von Vorteil, wenn alle Anweisungen, die Sie ausführen müssen, in SQL-Funktionen unterstützt werden. SQL-Funktionen können keine

SQL-Anweisungen enthalten, die eine Änderung der Datenbank bewirken. Darüber hinaus steht auch nur eine Teilmenge von SQL PL als Inline-SQL-PL für Funktionen zur Verfügung. Es ist zum Beispiel nicht möglich, CALL-Anweisungen auszuführen, Cursor zu deklarieren oder Ergebnismengen in SQL-Funktionen zurückzugeben.

Das folgende Beispiel zeigt eine SQL-Prozedur mit SQL PL, die sich zur Leistungsverbesserung für die Umwandlung in eine SQL-Funktion anbieten würde:

```
CREATE PROCEDURE GetPrice (IN Vendor CHAR&(20&),
                           IN Pid INT, OUT price DECIMAL(10,3))
LANGUAGE SQL
BEGIN
  IF Vendor eq; ssq;Vendor 1ssq;
    THEN SET price eq; (SELECT ProdPrice
                        FROM V1Table
                        WHERE Id = Pid);
  ELSE IF Vendor eq; ssq;Vendor 2ssq;
    THEN SET price eq; (SELECT Price FROM V2Table
                        WHERE Pid eq; GetPrice.Pid);
  END IF;
END
```

Der Code sähe als SQL-Funktion umgeschrieben folgendermaßen aus:

```
CREATE FUNCTION GetPrice (Vendor CHAR(20), Pid INT)
RETURNS  DECIMAL(10,3)
LANGUAGE SQL
BEGIN
  DECLARE price DECIMAL(10,3);
  IF Vendor = 'Vendor 1'
    THEN SET price = (SELECT ProdPrice
                      FROM V1Table
                      WHERE Id = Pid);
  ELSE IF Vendor = 'Vendor 2'
    THEN SET price = (SELECT Price FROM V2Table
                WHERE Pid = GetPrice.Pid);
  END IF;
  RETURN price;
END
```

Beachten Sie, dass sich der Aufruf einer Funktion von dem einer Prozedur unterscheidet. Verwenden Sie zum Aufrufen der Funktion die Anweisung VALUES oder rufen Sie die Funktion anstellen auf, an denen ein Ausdruck zulässig ist, wie zum Beispiel in einer Anweisung SELECT oder SET. Alle folgenden Methoden zum Aufrufen der neuen Funktionen sind zum Beispiel gültig:

```
VALUES (GetPrice('IBM', 324))

SELECT VName FROM Vendors WHERE GetPrice(Vname, Pid) < 10

SET price = GetPrice(Vname, Pid)
```

Vermeiden Sie mehrere Anweisungen in einer SQL PL-Prozedur, wenn eine Anweisung ausreicht.

Obwohl es im Allgemeinen als sinnvoller Lösungsansatz gilt, möglichst kurze SQL-Anweisungen zu schreiben, wird diese Empfehlung in der Praxis schnell vernachlässigt. Betrachten Sie zum Beispiel die folgenden SQL-Anweisungen:

```
INSERT INTO tab_comp VALUES (item1, price1, qty1);
INSERT INTO tab_comp VALUES (item2, price2, qty2);
INSERT INTO tab_comp VALUES (item3, price3, qty3);
```

Diese Anweisungen können wie folgt zu einer Anweisung zusammengefasst werden:

```
INSERT INTO tab_comp VALUES (item1, price1, qty1),
                            (item2, price2, qty2),
                            (item3, price3, qty3);
```

Die mehrzeilige Einfügung erfordert ungefähr nur ein Drittel des Zeitaufwands zur Ausführung der drei ursprünglichen Anweisungen. Isoliert betrachtet, könnte diese Verbesserung vernachlässigbar erscheinen. Wenn das Codefragment jedoch wiederholt ausgeführt wird, zum Beispiel in einer Schleife oder im Hauptteil eines Triggers, kann sich eine beträchtliche Verbesserung ergeben.

Analog kann eine Folge von Anweisungen SET umgeschrieben werden. Zum Beispiel:

```
SET A = expr1;
SET B = expr2;
SET C = expr3;
```

Diese Anweisungen können durch eine einzige Anweisung VALUES ersetzt werden:

```
VALUES expr1, expr2, expr3 INTO A, B, C;
```

Diese Transformation bewahrt die Semantik der ursprünglichen Anweisungsfolge, wenn keine Abhängigkeiten zwischen je zwei Anweisungen bestehen. Betrachten Sie dazu das folgende Beispiel:

```
SET A = monthly_avg * 12;
SET B = (A / 2) * correction_factor;
```

Wenn diese beiden Anweisungen wie folgt konvertiert werden:

```
VALUES (monthly_avg * 12, (A / 2) * correction_factor) INTO A, B;
```

bleibt die ursprüngliche Semantik nicht erhalten, weil die Ausdrücke vor dem Schlüsselwort INTO 'parallel' ausgewertet werden. Das bedeutet, dass der Wert, der der Variablen *B* zugewiesen wird, nicht auf dem Wert basiert, der der Variablen *A* zugewiesen wird, was jedoch die beabsichtigte Semantik der ursprünglichen Anweisungen war.

Reduzieren Sie mehrere SQL-Anweisungen auf einen einzigen SQL-Ausdruck.

Ebenso wie andere Programmiersprachen stellt die Sprache SQL zwei Typen von Bedingungskonstrukten zur Verfügung: prozedurale (Anweisungen IF und CASE) und funktionale (CASE-Ausdrücke). In den meisten Fällen können beide Typen verwendet werden, um eine Verarbeitung zu formulieren, wobei die Verwendung des einen oder des anderen eher eine Frage des Geschmacks ist. Trotzdem ist Logik, die mit CASE-Ausdrücken arbeitet, nicht nur kompakter, sondern auch effizienter als Logik, die mit CASE- oder IF-Anweisungen geschrieben ist.

Betrachten Sie das folgende SQL PL-Codefragment:

```
IF (Price <= MaxPrice) THEN
  INSERT INTO tab_comp(Id, Val) VALUES(Oid, Price);
ELSE
  INSERT INTO tab_comp(Id, Val) VALUES(Oid, MaxPrice);
END IF;
```

Die Bedingung der Klausel IF dient lediglich zur Entscheidung, welcher Wert in die Spalte 'tab_comp.Val' eingefügt wird. Zur Vermeidung des Kontextwechsels zwischen der prozeduralen Ebene und der Datenflussebene, kann dieselbe Logik durch eine Anweisung INSERT mit einem CASE-Ausdruck formuliert werden:

```
INSERT INTO tab_comp(Id, Val)
       VALUES(Oid,
            CASE
               WHEN (Price <= MaxPrice) THEN Price
               ELSE MaxPrice
            END);
```

Beachten Sie, dass CASE-Ausdrücke in jedem Kontext verwendet werden können, in dem ein Skalarwert erwartet wird. Insbesondere können solche Ausdrücke auf der rechten Seite von Zuweisungen verwendet werden. Zum Beispiel:

```
IF (Name IS NOT NULL) THEN
  SET ProdName = Name;
ELSEIF (NameStr IS NOT NULL) THEN
  SET ProdName = NameStr;
ELSE
  SET ProdName = DefaultName;
END IF;
```

Diese Anweisungsfolge kann in folgender Weise umgeschrieben werden:

```
SET ProdName = (CASE
                  WHEN (Name IS NOT NULL) THEN Name
                  WHEN (NameStr IS NOT NULL) THEN NameStr
                  ELSE  DefaultName
                END);
```

Tatsächlich lässt dieses spezielle Beispiel eine noch bessere Lösung zu:

```
SET ProdName = COALESCE(Name, NameStr, DefaultName);
```

Unterschätzen Sie nicht, welchen Nutzen es haben kann, sich die Zeit zur Analyse und Überarbeitung des eigenen SQL-Codes zu nehmen. Die Leistungsvorteile machen die investierte Zeit für das Analysieren und Umschreiben Ihrer Prozedur mehrfach bezahlt.

Nutzen Sie die mengenorientierte Set-at-a-Time-Semantik von SQL.

Prozedurale Konstrukte wie Schleifen, Zuweisungen und Cursor ermöglichen es, Verarbeitungen zu formulieren, die bei Verwendung von reinen SQL-DML-Anweisungen nicht möglich wären. Wenn prozedurale Anweisungen zur Verfügung stehen, besteht jedoch die Gefahr, dass sie verwendet werden, selbst wenn die betreffende Verarbeitung tatsächlich nur durch SQL-DML-Anweisungen ausgedrückt werden kann. Wie bereits zuvor erwähnt, kann die Leistung einer prozeduralen Verarbeitung um Größenordnungen schlechter sein als eine äquivalente Verarbeitung, die mithilfe von DML-Anweisungen formuliert ist. Betrachten Sie das folgende Codefragment:

```
DECLARE cur1 CURSOR FOR SELECT col1, col2 FROM tab_comp;
OPEN cur1;
FETCH cur1 INTO v1, v2;
WHILE SQLCODE <> 100 DO
  IF (v1 > 20) THEN
    INSERT INTO tab_sel VALUES (20, v2);
  ELSE
```

```
    INSERT INTO tab_sel VALUES (v1, v2);
  END IF;
  FETCH cur1 INTO v1, v2;
END WHILE;
```

Zunächst kann der Schleifenrumpf verbessert werden, indem die im vorangehenden Abschnitt "Reduzieren Sie mehrere SQL-Anweisungen auf einen einzigen SQL-Ausdruck" erläuterte Transformation angewendet wird:

```
DECLARE cur1 CURSOR FOR SELECT col1, col2 FROM tab_comp;
OPEN cur1;
FETCH cur1 INTO v1, v2;
WHILE SQLCODE <> 100 DO
  INSERT INTO tab_sel VALUES (CASE
                                WHEN v1 > 20 THEN 20
                                ELSE v1
                              END, v2);
  FETCH cur1 INTO v1, v2;
END WHILE;
```

Bei näherer Prüfung lässt sich jedoch feststellen, dass der gesamte Codeblock in eine Anweisung INSERT mit einem Subselect umgeschrieben werden kann:

```
INSERT INTO tab_sel (SELECT (CASE
                               WHEN col1 > 20 THEN 20
                               ELSE col1
                             END),
                             col2
                     FROM tab_comp);
```

In der ursprünglichen Formulierung gab es einen Kontextwechsel zwischen der prozeduralen Ebene und der Datenflussebene für jede Zeile in den Anweisungen SELECT. In der letzten Formulierung gibt es überhaupt keinen Kontextwechsel, und das Optimierungsprogramm hat eine Möglichkeit, die gesamte Verarbeitung global zu optimieren.

Andererseits wäre diese beträchtliche Vereinfachung nicht möglich gewesen, wenn jede der Anweisungen INSERT eine andere Zieltabelle hätte, wie im folgenden Beispiel gezeigt:

```
DECLARE cur1 CURSOR FOR SELECT col1, col2 FROM tab_comp;
OPEN cur1;
FETCH cur1 INTO v1, v2;
WHILE SQLCODE <> 100 DO
  IF (v1 > 20) THEN
    INSERT INTO tab_default VALUES (20, v2);
  ELSE
    INSERT INTO tab_sel VALUES (v1, v2);
  END IF;
  FETCH cur1 INTO v1, v2;
END WHILE;
```

Allerdings kann die Set-at-a-Time-Spezifik von SQL auch in diesem Fall genutzt werden:

```
INSERT INTO tab_sel (SELECT col1, col2
                     FROM tab_comp
                     WHERE col1 <= 20);
INSERT INTO tab_default (SELECT col1, col2
                         FROM tab_comp
                         WHERE col1 > 20);
```

Wenn Sie nach Möglichkeiten zur Verbesserung der Leistung vorhandener Prozedurlogik suchen, wird sich wahrscheinlich jeder Zeitaufwand, der zur Beseitigung von Cursorschleifen verwendet wird, bezahlt machen.

Versorgen Sie das DB2-Optimierungsprogramm mit aktuellen Informationen.

Wenn eine Prozedur erstellt wird, werden die einzelnen SQL-Abfragen der Prozedur in Abschnitte in einem Paket kompiliert. Das DB2-Optimierungsprogramm wählt einen Ausführungsplan für eine Abfrage unter anderen Gesichtspunkten auch nach Tabellenstatistiken (z. B. Tabellengrößen oder die relative Häufigkeit von Datenwerten in einer Spalte) und Indizes aus, die bei der Kompilierung der Abfrage verfügbar sind. Wenn Tabellen signifikante Änderungen aufweisen, kann es sinnvoll sein, DB2 erneut Statistiken zu diesen Tabellen erfassen zu lassen. Wenn Statistiken aktualisiert oder neue Indizes erstellt werden, kann es außerdem sinnvoll sein, die Pakete, die SQL-Prozeduren zugeordnet sind, die die Tabellen verwenden, durch Rebind erneut zu binden, um DB2 die Erstellung von Plänen zu ermöglichen, die die neuesten Statistiken und Indizes nutzen.

Tabellenstatistiken können mithilfe des Befehls RUNSTATS aktualisiert werden. Für den Rebind des Pakets, das einer SQL-Prozedur zugeordnet ist, können Sie die integrierte Prozedur REBIND_ROUTINE_PACKAGE verwenden, die in DB2 Version 8.1 verfügbar ist. Mit dem folgenden Befehl wird zum Beispiel ein Rebind des Pakets für die Prozedur MYSCHEMA.MYPROC durchgeführt:

```
CALL SYSPROC.REBIND_ROUTINE_PACKAGE('P', 'MYSCHEMA.MYPROC', 'ANY')
```

Dabei gibt 'P' an, dass das Paket einer Prozedur entspricht. 'ANY' gibt an, dass alle Funktionen und Typen im SQL-Pfad bei der Funktions- und Typauflösung berücksichtigt werden. Weitere Informationen finden Sie im Eintrag zum Befehl REBIND im Handbuch 'Command Reference'.

Verwenden Sie Datenfeldgruppen (Arrays).

Mit Datenfeldgruppen (Arrays) können Sie Datensammlungen effizient zwischen Anwendungen und gespeicherten Prozeduren übergeben sowie transiente Datensammlungen in SQL-Prozeduren speichern und bearbeiten, ohne die relationalen Tabellen verwenden zu müssen. Operatoren, die für Feldgruppen in SQL-Prozeduren verfügbar sind, ermöglichen ein effizientes Speichern und Abrufen von Daten. Anwendungen, die Feldgruppen mäßiger Größe erstellen, erfahren eine wesentlich bessere Leistung als Anwendungen, die sehr große Feldgruppen (im Maßstab mehrerer Megabyte) erstellen, da die gesamte Feldgruppe im Hauptspeicher gespeichert wird. Weitere Informationen finden Sie unter *Zugehörige Links*.

Kapitel 22. Abfragezugriffspläne

Der SQL- und XQuery-Compilerprozess

Der SQL- und XQuery-Compiler führt mehrere Schritte aus, um einen Zugriffsplan zu erstellen, der ausgeführt werden kann. Diese Schritte werden in der folgenden Abbildung dargestellt und in den anschließenden Abschnitten beschrieben. Beachten Sie, dass einige Schritte nur für Abfragen in einer föderierten Datenbank durchgeführt werden.

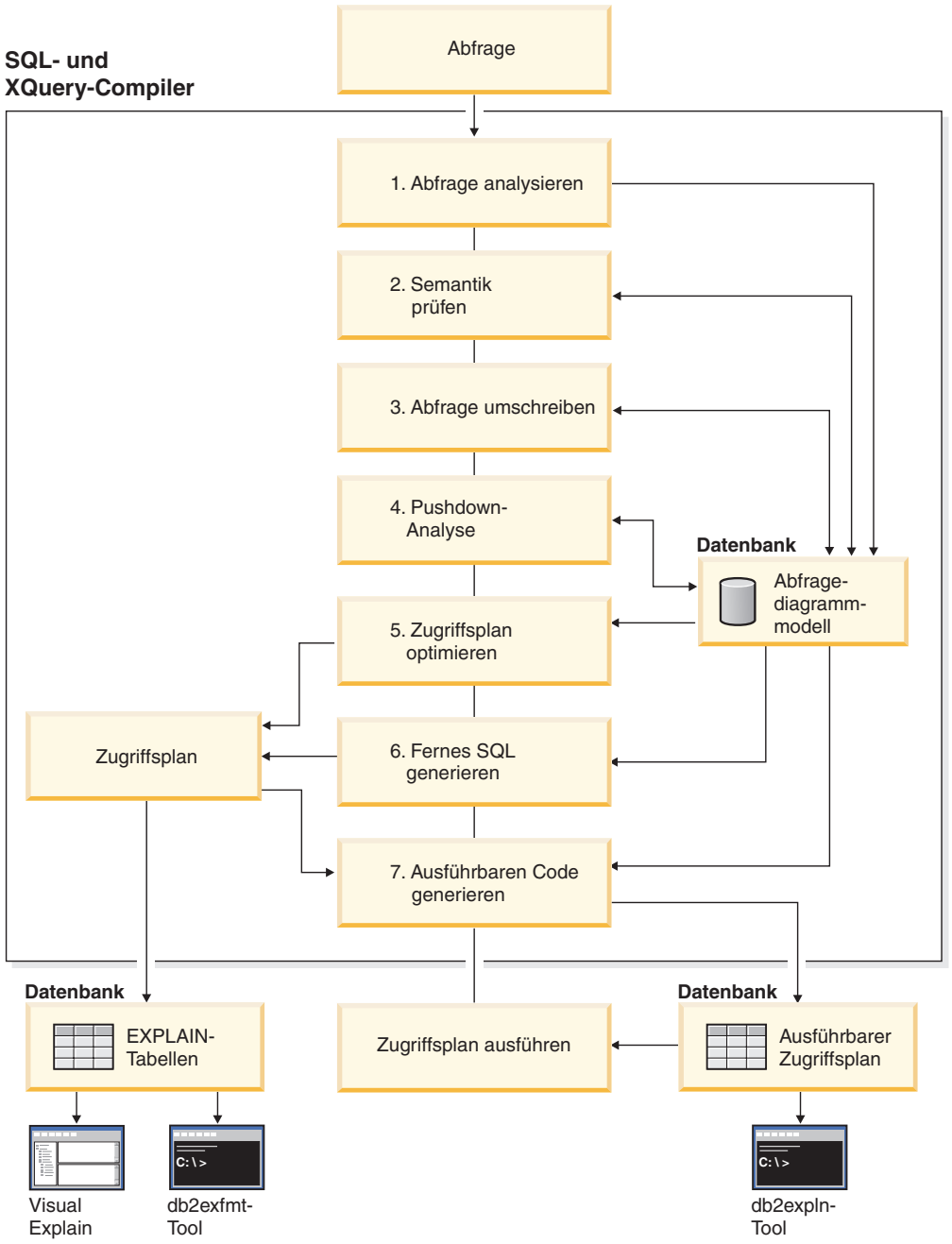


Abbildung 22. Vom SQL- und XQuery-Compiler ausgeführte Schritte

Abfragediagrammmodell

Das *Abfragediagrammmodell* ist eine interne, im Speicher befindliche Datenbank, welche die Abfrage im Verlauf der in den folgenden Schritten beschriebenen Verarbeitung darstellt:

1. **Abfrage analysieren**

   Der SQL- und XQuery-Compiler analysiert die Abfrage, um die Gültigkeit der Syntax zu überprüfen. Wenn Syntaxfehler festgestellt werden, stoppt der Abfragecompiler die Verarbeitung und gibt den entsprechenden Fehler an die

Anwendung zurück, die die Abfrage übergeben hat. Wenn die Analyse abgeschlossen ist, wird eine interne Darstellung der Abfrage erstellt und im Abfragediagrammmodell gespeichert.

2. **Semantik prüfen**

   Der Compiler stellt sicher, dass keine Inkonsistenzen zwischen Teilen der Anweisung bestehen. Ein einfaches Beispiel für eine Semantikprüfung ist die Überprüfung durch den Compiler, ob eine für die Skalarfunktion YEAR angegebene Spalte mit dem Datentyp DATETIME (Datum/Uhrzeit) definiert wurde.

   Der Compiler fügt außerdem die Bedingungssemantik in das Abfragediagrammmodell ein, zu der die Auswirkungen der referenziellen Integritätsbedingungen, der Prüfungen auf Integritätsbedingungen in Tabellen, der Trigger und der Sichten gehören. Das Abfragediagrammmodell enthält die gesamte Semantik der Abfrage, einschließlich der Abfrageblöcke, Unterabfragen, Korrelationen, abgeleiteten Tabellen, Ausdrücke, Datentypen, Datentypumsetzungen, Codepageumsetzungen und der Verteilungsschlüssel.

3. **Abfrage umschreiben**

   Der Compiler verwendet die im Abfragediagrammmodell gespeicherte globale Semantik, um die Abfrage in eine Form umzusetzen, die leichter optimiert werden kann, und speichert das Ergebnis im Abfragediagrammmodell.

   Zum Beispiel kann der Compiler ein Vergleichselement verschieben und somit die Ebene ändern, auf der es angewandt wird, um dadurch potenziell die Abfrageleistung zu erhöhen. Diese Art der Verschiebung einer Operation wird als allgemeine Vergleichselementverschiebung (*General Predicate Pushdown*) bezeichnet. In einer Umgebung mit partitionierten Datenbanken sind die folgenden Abfrageoperationen etwas rechenintensiver:

   - Spaltenberechnungen (Aggregation)
   - Umverteilen von Zeilen
   - Korrelierte Unterabfragen, d. h. Unterabfragen, die einen Verweis auf eine Spalte einer Tabelle enthalten, die sich außerhalb der Unterabfrage befindet

   Für einige Abfragen in einer Umgebung mit partitionierten Datenbanken kann eine Dekorrelierung im Rahmen des Umschreibens der Abfrage erfolgen.

4. **Pushdown-Analyse (föderierte Datenbanken)**

   Die Hauptfunktion dieses Schrittes ist, dem Optimierungsprogramm eine Empfehlung zu liefern, ob eine Operation an eine Datenquelle verschoben und fern ausgewertet werden kann, was als *Pushdown* bezeichnet wird. Diese Art von Pushdown-Aktivität ist für Datenquellenabfragen spezifisch und bildet eine Erweiterung zu den allgemeinen Operationen der Vergleichselementverschiebung.

   Dieser Schritt wird nur ausgeführt, wenn Abfragen für föderierte Datenbanken ausgeführt werden.

5. **Zugriffsplan optimieren**

   Mit dem Abfragediagrammmodell als Eingabe generiert die Optimierungskomponente des Compilers zahlreiche alternative Ausführungspläne zur Erfüllung der Abfrage. Das Optimierungsprogramm schätzt den Ausführungsaufwand mithilfe der Statistiken für Tabellen, Indizes, Spalten und Funktionen für jeden der verschiedenen Pläne ab. Dann wählt es den Plan mit dem geringsten geschätzten Ausführungsaufwand aus. Das Optimierungsprogramm verwendet das Abfragediagrammmodel, um die Abfragesemantik zu analysieren und Informationen zu einer Vielzahl von Faktoren, einschließlich Indizes, Basistabellen, abgeleitete Tabellen, Unterabfragen, Korrelationen und Rekursion, zu erhalten.

Das Optimierungsprogramm kann außerdem eine andere Art von Verschiebeoperation (Pushdown) in Betracht ziehen, nämlich für *Spaltenberechnungen und Sortierungen*. Die Leistung kann erhöht werden, wenn die Auswertung dieser Operationen an die Komponente der Datenverwaltungsservices (Data Management Services) verschoben werden kann.

Das Optimierungsprogramm berücksichtigt auch, ob es Pufferpools verschiedener Größen gibt, wenn es die Auswahl der Seitengröße festlegt. Der Faktor, dass die Umgebung eine partitionierte Datenbank enthält, wird ebenso berücksichtigt wie die Möglichkeit, den ausgewählten Plan für eine potenzielle abfrageinterne Parallelität in einer symmetrischen Multiprozessorumgebung (SMP-Umgebung) auszulegen. Diese Informationen werden vom Optimierungsprogramm zur Auswahl des am besten geeigneten Zugriffsplans für die Abfrage verwendet.

Die Ausgabe dieses Schritts des Compilers ist ein Zugriffsplan. Dieser Zugriffsplan stellt die Informationen bereit, die in den EXPLAIN-Tabellen erfasst werden. Die Informationen, die zur Generierung des Zugriffsplans dienten, können mit einer Momentaufnahme der EXPLAIN-Einrichtung erfasst werden.

6. **Fernes SQL generieren (föderierte Datenbanken)**

   Der endgültige Plan, der vom Optimierungsprogramm gewählt wird, kann aus einer Reihe von Schritten bestehen, die an einer fernen Datenquelle ausgeführt werden. Für Operationen, die an der jeweiligen Datenquelle ausgeführt werden, erstellt der Schritt der Generierung von fernem SQL eine effiziente SQL-Anweisung, die auf der SQL-Version der Datenquelle beruht.

7. **„Ausführbaren" Code generieren**

   Im letzten Schritt verwendet der Compiler den Zugriffsplan und das Abfragediagrammmodell, um einen ausführbaren Zugriffsplan oder Abschnitt für die Abfrage zu erstellen. Bei dieser Generierung des Codes werden Informationen des Abfragediagrammmodells verwendet, um eine Wiederholung der Ausführung von Ausdrücken zu vermeiden, die für eine Abfrage nur einmal berechnet werden müssen. Diese Art der Optimierung ist beispielsweise für Umwandlungen von Codepages und die Verwendung von Hostvariablen möglich.

   Damit eine Abfrageoptimierung bzw. -reoptimierung für statische und dynamische SQL- und XQuery-Anweisungen, die Hostvariablen, Sonderregister oder Parametermarken enthalten, ausgeführt werden kann, binden Sie das Paket mit der Bindeoption REOPT. Wenn diese Bindeoption verwendet wird, wird der Zugriffspfad für eine SQL- oder XQuery-Anweisung, die zu diesem Paket gehört und Hostvariablen, Parametermarken oder Sonderregister enthält, mit den Werten dieser Variablen und nicht mit Standardschätzwerten, die der Compiler auswählt, optimiert. Diese Optimierung erfolgt bei der Ausführung der Abfrage, wenn die Werte verfügbar sind.

   Informationen über die Zugriffspläne für statische SQL- und XQuery-Anweisungen werden in den Systemkatalogtabellen gespeichert. Wenn das Paket ausgeführt wird, verwendet der Datenbankmanager die in den Systemkatalogtabellen gespeicherten Informationen, um festzulegen, wie auf die Daten zugegriffen werden soll, und Ergebnisse für eine Abfrage bereitzustellen. Diese Informationen werden vom Tool *db2expln* verwendet.

Anmerkung: Führen Sie RUNSTATS in angemessenen Intervallen für Tabellen aus, die häufig geändert werden. Das Optimierungsprogramm benötigt aktuelle Statistikinformationen zu den Tabellen und ihren Daten, um die effizientesten Zugriffspläne zu generieren. Führen Sie einen Rebind für Ihre Anwendung durch, um die aktualisierten Statistiken zu nutzen. Wenn das Dienstprogramm RUNSTATS nicht ausgeführt wird oder das Optimierungsprogramm annimmt, dass RUNSTATS für leere oder fast leere Tabellen ausgeführt wurde, kann das

Optimierungsprogramm entweder Standardwerte verwenden oder versuchen, bestimmte Statistikdaten mithilfe der Anzahl der Dateiseiten (FPAGES) abzuleiten, die zum Speichern der Tabelle auf dem Plattenspeicher verwendet werden. Die Gesamtanzahl belegter Blöcke wird in der Spalte ACTIVE_BLOCKS gespeichert.

Methoden zum Umschreiben von Abfragen und Beispiele

Während der Phase des Umschreibens der Abfrage setzt der Abfragecompiler SQL- und XQuery-Anweisungen in Formate um, die leichter optimiert werden und infolgedessen die möglichen Zugriffspfade verbessern können. Das Umschreiben von Abfragen ist besonders wichtig für komplexe Abfragen, zu denen auch Abfragen mit zahlreichen Unterabfragen oder Joins zählen. Tools zur Generierung von Abfragen erstellen häufig diese sehr komplexen Arten von Abfragen.

Zur Beeinflussung der Anzahl von Regeln für das Umschreiben von Abfragen, die auf eine SQL- oder XQuery-Anweisung angewendet werden, ändern Sie die Optimierungsklasse. Einige der Ergebnisse dieses Umschreibens der Abfrage können Sie mithilfe der EXPLAIN-Einrichtung oder über Visual Explain anzeigen.

Abfragen können durch die folgenden drei Hauptmethoden umgeschrieben werden:

- **Zusammenfügen von Operationen**

  Zur Generierung der Abfrage in ein Format, das möglichst wenige Operationen, insbesondere SELECT-Operationen, aufweist, schreibt der SQL- und XQuery-Compiler Abfragen um, damit Abfrageoperationen zusammengefügt werden können. Die folgenden Beispiele zeigen einige der Operationen, die zusammengefügt werden können:

  – Beispiel - Zusammenfügen von Sichten

    Eine SELECT-Anweisung, die Sichten verwendet, kann die Joinfolge der Tabelle einschränken und zudem überflüssige Joins von Tabellen nach sich ziehen. Wenn die Sichten während des Umschreibens der Abfrage zusammengefügt werden, können diese Einschränkungen aufgehoben werden.

  – Beispiel - Umsetzungen von Unterabfragen in Joins

    Wenn eine SELECT-Anweisung eine Unterabfrage enthält, kann die Auswahl der Reihenfolgeverarbeitung der Tabelle eingeschränkt sein.

  – Beispiel - Eliminierung überflüssiger Joins

    Während des Umschreibens der Abfrage können überflüssige Joins entfernt werden, um die SELECT-Anweisung zu vereinfachen.

  – Beispiel - Gemeinsame Spaltenberechnungen

    Wenn eine Abfrage verschiedene Funktionen verwendet, kann durch Umschreiben die Anzahl der erforderlichen Berechnungen reduziert werden.

- **Verschieben von Operationen**

  Zur Generierung der Abfrage mit der kleinstmöglichen Anzahl an Operationen und Vergleichselementen schreibt der Compiler Abfragen um, um Abfrageoperationen zu verschieben. Die folgenden Beispiele zeigen einige Operationen, die verschoben werden können:

  – Beispiel - Eliminierung von DISTINCT

    Während des Umschreibens einer Abfrage kann das Optimierungsprogramm den Zeitpunkt verschieben, zu dem die DISTINCT-Operation durchgeführt wird, um den Aufwand für diese Operation zu verringern. In dem bereitgestellten erweiterten Beispiel wird die DISTINCT-Operation vollständig entfernt.

– Beispiel - Allgemeine Verschiebung von Vergleichselementen (Pushdown)

  Während des Umschreibens von Abfragen kann das Optimierungsprogramm die Reihenfolge der angewandten Vergleichselemente ändern, sodass die Vergleichselemente, die die Auswahl in größerem Maße einschränken, zum frühstmöglichen Zeitpunkt angewandt werden.

– Beispiel - Dekorrelierung

  In einer Umgebung mit partitionierten Datenbanken erfordert das Verschieben von Ergebnismengen zwischen den Datenbankpartitionen hohen Aufwand. Den Umfang des Broadcastbetriebs (Rundsenden) an andere Datenbankpartitionen oder die Anzahl der Broadcastvorgänge zu reduzieren, oder beides, gehört zu den Zielen des Umschreibens von Abfragen.

- **Übersetzen von Vergleichselementen**

  Der SQL- und XQuery-Compiler schreibt Abfragen um, damit vorhandene Vergleichselemente für eine bestimmte Abfrage in ein optimiertes Format umgesetzt werden. Die folgenden Beispiele zeigen einige der Vergleichselemente, die übersetzt werden können:

  – Beispiel - Hinzufügen implizierter Vergleichselemente

    Während des Umschreibens können Vergleichselemente der Abfrage hinzugefügt werden, um dem Optimierungsprogramm die Möglichkeit zu geben, weitere Tabellenjoins bei der Auswahl des günstigsten Zugriffsplans für die Abfrage in Betracht zu ziehen.

  – Beispiel - Transformation von OR zu IN

    Während des Umschreibens einer Abfrage kann für einen effizienteren Zugriffsplan ein Vergleichselement OR in ein Vergleichselement IN umgesetzt werden. Der SQL- und XQuery-Compiler kann auch ein Vergleichselement IN in ein Vergleichselement OR umsetzen, wenn diese Transformation einen günstigeren Zugriffsplan generieren würde.

Beispiel für das Umschreiben durch den Compiler: Zusammenfügen von Sichten

Nehmen Sie zum Beispiel an, Sie haben Zugriff auf die beiden folgenden Sichten der Tabelle EMPLOYEE, von denen die eine die Mitarbeiter mit einer hohen Ausbildungsstufe (EDLEVEL) und die andere die Mitarbeiter mit einem Gehalt (SALARY) über 35.000 Dollar zeigt:

```
CREATE VIEW EMP_EDUCATION (EMPNO, FIRSTNME, LASTNAME, EDLEVEL) AS
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL
  FROM EMPLOYEE
 WHERE EDLEVEL > 17
CREATE VIEW EMP_SALARIES (EMPNO, FIRSTNAME, LASTNAME, SALARY) AS
SELECT EMPNO, FIRSTNME, LASTNAME, SALARY
  FROM EMPLOYEE
 WHERE SALARY > 35000
```

Jetzt wird beispielsweise die folgende Abfrage ausgeführt, um die Mitarbeiter, die eine hohe Ausbildungsstufe haben und deren Gehalt über 35.000 Dollar liegt, aufzulisten:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMP_EDUCATION E1,
       EMP_SALARIES  E2
 WHERE E1.EMPNO = E2.EMPNO
```

Während der Phase des Umschreibens könnten diese beiden Sichten zusammengefügt werden, um folgende Abfrage zu erstellen:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

Durch das Zusammenfügen der Anweisungen SELECT der beiden Sichten mit der vom Benutzer geschriebenen Anweisung SELECT kann das Optimierungsprogramm mehr Möglichkeiten bei der Auswahl des Zugriffsplans in Betracht ziehen. Darüber hinaus kann die Abfrage noch weiter umgeschrieben werden, wenn die beiden zusammengefügten Sichten dieselbe Basistabelle verwenden.

Beispiel - Umsetzungen von Unterabfragen in Joins

Angenommen, der SQL- und XQuery-Compiler verfügt über eine Abfrage mit einer Unterabfrage wie die folgende:

```
SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO
FROM EMPLOYEE
WHERE WORKDEPT IN
      (SELECT DEPTNO
         FROM DEPARTMENT
        WHERE DEPTNAME = 'OPERATIONS')
```

Diese Unterabfrage wird vom SQL-Compiler in eine Joinabfrage der folgenden Form umgewandelt:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME, PHONENO
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNAME = 'OPERATIONS'
```

Im Allgemeinen ist ein Join in der Ausführung wesentlich effektiver als eine Unterabfrage.

Beispiel - Eliminierung überflüssiger Joins

Manche geschriebenen oder generierten Abfragen enthalten unnötige Joins. Abfragen wie die folgende könnten auch während des Umschreibens einer Abfrage generiert werden.

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

In dieser Abfrage kann der SQL- und XQuery-Compiler den Join eliminieren und die Abfrage wie folgt vereinfachen:

```
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL, SALARY
  FROM EMPLOYEE
 WHERE EDLEVEL > 17
   AND SALARY  > 35000
```

Im folgenden Beispiel wird davon ausgegangen, dass zwischen den Beispieltabellen EMPLOYEE und DEPARTMENT eine referenzielle Integritätsbedingung über die Abteilung (WORKDEPT/DEPNO) vorhanden ist. Zuerst wird eine Sicht erstellt.

```
CREATE VIEW PEPLVIEW
   AS SELECT FIRSTNME, LASTNAME, SALARY, DEPTNO, DEPTNAME, MGRNO
         FROM EMPLOYEE E DEPARTMENT D
         WHERE E.WORKDEPT = D.DEPTNO
```

Eine Abfrage wie die folgende:

```
SELECT LASTNAME, SALARY
   FROM PEPLVIEW
```

wird dann geändert in:

```
SELECT LASTNAME, SALARY
   FROM EMPLOYEE
   WHERE WORKDEPT NOT NULL
```

Beachten Sie bei dieser Situation, dass Benutzer die Abfrage eventuell nicht umschreiben können, selbst wenn sie wüssten, dass dies möglich ist, weil sie keinen Zugriff auf die zugrunde liegenden Tabellen haben. Sie haben eventuell nur Zugriff auf die oben gezeigte Sicht. Daher muss diese Art von Optimierung im Datenbankmanager ausgeführt werden.

Redundanz in Joins mit referenzieller Integrität ist in folgenden Fällen wahrscheinlich:

- Sichten sind mit Joins definiert.
- Abfragen werden automatisch generiert.

  Es gibt zum Beispiel automatisierte Tools in Abfragemanagern, die das Schreiben optimierter Abfragen durch Benutzer verhindern.

Beispiel - Gemeinsame Spaltenberechnungen

Bei Verwendung mehrerer Funktionen in einer Abfrage können zahlreiche Berechnungen entstehen, die zeitintensiv sind. Durch Reduzieren der für die Abfrage erforderlichen Anzahl von Berechnungen kann der Zugriffsplan verbessert werden. Der SQL- und XQuery-Compiler verfügt über eine Abfrage, die mehrere Funktionen verwendet, wie zum Beispiel:

```
SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  AVG(SALARY+BONUS+COMM) AS OAVG,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE;
```

wird vom SQL-Compiler wie folgt umgewandelt:

```
SELECT OSUM,
  OSUM/OCOUNT
  OCOUNT
FROM (SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE) AS SHARED_AGG;
```

Durch dieses Umschreiben benötigt die Abfrage statt 2 Summen und 2 Zählern nur noch 1 Summe und 1 Zähler.

Beispiel für das Umschreiben durch den Compiler: Eliminierung von DISTINCT

Für das folgende Abfragebeispiel wird angenommen, dass die Spalte EMPNO als Primärschlüssel der Tabelle EMPLOYEE definiert wurde:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME
    FROM EMPLOYEE
```

Diese Abfrage würde so umgeschrieben, dass die Klausel DISTINCT entfernt wird:

```
SELECT EMPNO, FIRSTNME, LASTNAME
    FROM EMPLOYEE
```

Da hier der Primärschlüssel ausgewählt wird, weiß der SQL- und XQuery-Compiler, dass jede zurückgegebene Zeile bereits eindeutig ist. In diesem Fall wird das Schlüsselwort DISTINCT nicht benötigt. Wenn die Abfrage nicht umgeschrieben wird, erstellt das Optimierungsprogramm einen Plan, der die nötigen Verarbeitungsschritte, wie zum Beispiel eine Sortierung, enthält, um sicherzustellen, dass die Spalten eindeutig sind.

Beispiel - Allgemeines Verschieben von Vergleichselementen (Pushdown)

Durch das Ändern der Ebene, auf der Vergleichselemente normalerweise angewendet werden, lässt sich eventuell eine Leistungsverbesserung erreichen. Zum Beispiel sei die folgende Sicht gegeben, die eine Liste aller Mitarbeiter der Abteilung „D11" zusammenstellt:

```
CREATE VIEW D11_EMPLOYEE
 (EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM)
AS SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM
     FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
```

Es wird nun die folgende Abfrage aufgesetzt:

```
SELECT FIRSTNME, PHONENO
  FROM D11_EMPLOYEE
 WHERE LASTNAME = 'BROWN'
```

In der Phase des Umschreibens verschiebt der Compiler das Vergleichselement `LASTNAME = 'BROWN'` nach unten in die Sicht D11_EMPLOYEE. Dadurch kann das Vergleichselement früher und möglicherweise effektiver angewandt werden. Die tatsächliche Abfrage, die in diesem Beispiel zur Ausführung kommen könnte, sieht folgendermaßen aus:

```
SELECT FIRSTNME, PHONENO
  FROM EMPLOYEE
 WHERE LASTNAME = 'BROWN'
   AND WORKDEPT = 'D11'
```

Das Verschieben von Vergleichselementen ist nicht auf Sichten beschränkt. Beispiele für andere Situationen, in denen Vergleichselemente verschoben werden können, sind Klauseln UNION, GROUP BY und abgeleitete Tabellen (verschachtelte Tabellenausdrücke oder allgemeine Tabellenausdrücke).

Beispiel - Dekorrelierung

In einer Umgebung mit partitionierten Datenbanken kann der SQL- und XQuery-Compiler die folgende Abfrage umschreiben:

Lokalisieren aller Mitarbeiter, die in der Programmierung arbeiten und unterbezahlt sind.

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
  FROM EMPLOYEE E, PROJECT P
 WHERE P.EMPNO = E.EMPNO
   AND P.PROJNAME LIKE '%PROGRAMMING%'
   AND E.SALARY+E.BONUS+E.COMM <
     (SELECT AVG(E1.SALARY+E1.BONUS+E1.COMM)
```

```
FROM EMPLOYEE E1, PROJECT P1
WHERE P1.PROJNAME LIKE '%PROGRAMMING%'
  AND P1.PROJNO = A.PROJNO
  AND E1.EMPNO = P1.EMPNO)
```

Da die Abfrage korreliert ist und wahrscheinlich weder PROJECT noch EMPLOYEE über die Spalte PROJNO partitioniert sind, wird möglicherweise jedes Projekt im Broadcastbetrieb an jede Datenbankpartition übermittelt. Außerdem müsste die Unterabfrage viele Male ausgewertet werden.

Der SQL- und XQuery-Compiler kann die Abfrage wie folgt umschreiben:

- Die eindeutige Liste (DISTINCT) der Mitarbeiter (Employees) ermitteln, die an Programmierungsprojekten arbeiten, und diese DIST_PROJS nennen. Diese Liste muss mit DISTINCT erstellt werden, damit sichergestellt wird, dass die Spaltenberechnung nur einmal pro Projekt erfolgt:

  ```
  WITH DIST_PROJS(PROJNO, EMPNO) AS
  (SELECT DISTINCT PROJNO, EMPNO
   FROM PROJECT P1
   WHERE P1.PROJNAME LIKE '%PROGRAMMING%')
  ```

- Die eindeutige Liste der Mitarbeiter, die an Programmierungsprojekten arbeiten mit der Mitarbeitertabelle verknüpfen, um die durchschnittliche Entlohnung je Projekt (AVG_PER_PROJ) zu ermitteln:

  ```
  AVG_PER_PROJ(PROJNO, AVG_COMP) AS
  (SELECT P2.PROJNO, AVG(E1.SALARY+E1.BONUS+E1.COMM)
   FROM EMPLOYEE E1, DIST_PROJS P2
   WHERE E1.EMPNO = P2.EMPNO
   GROUP BY P2.PROJNO)
  ```

- Die neue Abfrage würde wie folgt aussehen:

  ```
  SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
         E.SALARY+E.BONUS+E.COMM AS COMPENSATION
   FROM PROJECT P, EMPLOYEE E, AVG_PER_PROG A
  WHERE P.EMPNO = E.EMPNO
    AND P.PROJNAME LIKE '%PROGRAMMING%'
    AND P.PROJNO = A.PROJNO
    AND E.SALARY+E.BONUS+E.COMM < A.AVG_COMP
  ```

Die umgeschriebene Abfrage berechnet AVG_COMP je Projekt (AVG_PRE_PROJ) und kann das Ergebnis an alle Datenbankpartitionen übermitteln, in denen die Tabelle EMPLOYEE enthalten ist.

Beispiel für das Umschreiben durch den Compiler: implizierte Vergleichselemente

Die folgende Abfrage erstellt eine Liste der Manager, deren Abteilung an Abteilung „E01" berichten, und der Projekte, für die die Manager verantwortlich sind:

```
SELECT DEPT.DEPTNAME DEPT.MGRNO, EMP.LASTNAME, PROJ.PROJNAME
    FROM DEPARTMENT DEPT,
              EMPLOYEE   EMP,
              PROJECT    PROJ
  WHERE DEPT.ADMRDEPT = 'E01'
      AND DEPT.MGRNO = EMP.EMPNO
      AND EMP.EMPNO  = PROJ.RESPEMP
```

Beim Umschreiben der Abfrage wird das folgende implizierte Vergleichselement hinzugefügt:

```
DEPT.MGRNO = PROJ.RESPEMP
```

Als Ergebnis dieses Umschreibens kann das Optimierungsprogramm weitere Joins in die Berechnung mit einbeziehen, wenn es versucht, den günstigsten Zugriffsplan für die Abfrage auszuwählen.

Neben der oben gezeigten Ausnutzung der Transitivität der Vergleichselemente werden durch das Umschreiben auch zusätzliche lokale Vergleichselemente aufgrund der durch Gleichheitsvergleichselemente implizierten Transitivität abgeleitet. Zum Beispiel stellt die folgende Abfrage eine Liste der Namen der Abteilungen (deren Abteilungsnummer größer als „E00" ist) und der Mitarbeiter, die in diesen Abteilungen arbeiten, zusammen.

```
SELECT EMPNO, LASTNAME, FIRSTNAME, DEPTNO, DEPTNAME
    FROM EMPLOYEE EMP,
                DEPARTMENT DEPT
  WHERE EMP.WORKDEPT = DEPT.DEPTNO
      AND DEPT.DEPTNO > 'E00'
```

Dieser Abfrage wird in der Phase des Umschreibens das folgende implizierte Vergleichselement hinzugefügt:

```
EMP.WORKDEPT > 'E00'
```

Das Ergebnis dieses Umschreibens besteht darin, dass das Optimierungsprogramm die Anzahl der Zeilen, die verknüpft werden müssen, verringern kann.

Beispiel - Transformationen von OR zu IN

Nehmen Sie an, eine Klausel OR verknüpft zwei oder mehr einfache Gleichheitsvergleichselemente für dieselbe Spalte, wie im folgenden Beispiel:

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO = 'D11'
    OR DEPTNO = 'D21'
    OR DEPTNO = 'E21'
```

Wenn es für die Spalte DEPTNO keinen Index gibt, erlaubt die Umwandlung der Klausel OR in das folgende Vergleichselement IN eine effektivere Verarbeitung der Abfrage:

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

Anmerkung: In einigen Fällen kann der Datenbankmanager ein IN-Vergleichselement in eine Gruppe von Klauseln OR umwandeln, sodass logische OR-Joins für Indizes (Index ORing) durchgeführt werden können.

Vergleichselementtypologie und Zugriffspläne

Eine Benutzeranwendung fordert eine Reihe von Zeilen aus der Datenbank mithilfe einer Abfrageanweisung an, in der Qualifikationsmerkmale für die gewünschten Zeilen angegeben werden, die als Ergebnismenge zurückgegeben werden sollen. Diese Qualifikationsbedingungen stehen in der Regel in der WHERE-Klausel der Abfrage. Solche Qualifikationsbedingungen werden als *Vergleichselemente* bezeichnet. Vergleichselemente können in vier Kategorien zusammengefasst werden, die dadurch bestimmt sind, wie und wann das jeweilige Vergleichselement im Auswertungsprozess verwendet wird. Die Kategorien werden im Folgenden in der Reihenfolge von der höchsten bis zur niedrigsten Leistung geordnet aufgelistet:

1. Bereichsbegrenzende Vergleichselemente

2. Bei Indexsuchen als Suchargument verwendbare Vergleichselemente (Index SARGable)
3. Bei Datensuchen als Suchargument verwendbare Vergleichselemente (Data SARGable)
4. Restvergleichselemente

Anmerkung: Die Bezeichnung *SARGable* ist aus dem Begriff *s*earch *arg*ument abgeleitet.

Die folgende Tabelle enthält eine Übersicht über die Kategorien von Vergleichselementen. In den nachfolgenden Abschnitten werden die einzelnen Kategorien eingehender beschrieben.

Tabelle 63. Zusammenfassung der Merkmale der Vergleichselementkategorien

Merkmal	Vergleichselementkategorie			
	Bereichs-begrenzend	**Indexsuch-argument**	**Datensuch-argument**	**Restvergleichs-element**
Verringern der Index-Ein-/Ausgabe	Ja	Nein	Nein	Nein
Verringern der Datenseitenein-/ausgabe	Ja	Ja	Nein	Nein
Verringern der Anzahl intern übergebener Zeilen	Ja	Ja	Ja	Nein
Verringern der Anzahl der den Bedingungen entsprechenden Zeilen	Ja	Ja	Ja	Ja

Bereichsbegrenzende und bei Indexsuchen als Suchargumente verwendbare Vergleichselemente

Bereichsbegrenzende Vergleichselemente begrenzen den Bereich einer Indexsuche. Sie geben Start- und Stoppschlüsselwerte für die Indexsuche an. Bei Indexsuchen als Suchargument verwendbare Vergleichselemente können nicht den Bereich einer Suche begrenzen, aber sie können mithilfe des Index ausgewertet werden, da die im Vergleichselement verwendeten Spalten Teil des Indexschlüssels sind. Betrachten Sie zum Beispiel den folgenden Index:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

Betrachten Sie dazu eine Abfrage, welche die folgende WHERE-Klausel enthält:

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

Die ersten beiden Vergleichselemente (NAME = :hv1, DEPT = :hv2) sind bereichsbegrenzende Vergleichselemente, während YEARS > :hv5 ein bei Indexsuchen als Suchargument verwendbares Vergleichselement ist.

Das Optimierungsprogramm verwendet die Indexdaten bei der Auswertung dieser Vergleichselemente, anstatt die Basistabelle zu lesen. Diese in Indizes als Suchargumente verwendbaren (*Index SARGable*) Vergleichselemente verringern die Anzahl der Zeilen, die aus der Tabelle gelesen werden müssen, sie beeinflussen jedoch nicht die Anzahl von Indexseiten, auf die zugegriffen wird.

Vergleichselemente für XML-Daten, die in XMLEXISTS- und XMLTABLE-Ausdrücken auftreten, werden ebenfalls von Suchvorgängen des Datenoperators XSCAN unterstützt. Einige dieser Vergleichselemente werden auch von Indexbereichssuchen unterstützt.

Bei Datensuchen als Suchargument verwendbare Vergleichselemente

Vergleichselemente, die nicht vom Indexmanager ausgewertet werden können, sondern nur von den Datenverwaltungsservices, werden als bei Datensuchen verwendbare (*Data SARGable*) Vergleichselemente bezeichnet. Für solche Vergleichselemente ist in der Regel ein Zugriff auf einzelne Zeilen einer Tabelle erforderlich. Bei Bedarf rufen die Datenverwaltungsservices die zur Auswertung des Vergleichselements benötigten Spalten und andere Spalten ab, um die Spalten für die SELECT-Liste, die nicht aus dem Index abgerufen werden konnten, zur Verfügung zu stellen.

Betrachten Sie zum Beispiel einen einzelnen Index, der für die Tabelle PROJECT definiert ist:

```
INDEX IX0:  PROJNO ASC
```

Für die folgende Abfrage wird dann das Vergleichselement `DEPTNO = 'D11'` als bei Datensuchen als Suchargument verwendbar betrachtet.

```
SELECT PROJNO, PROJNAME, RESPEMP
        FROM PROJECT
      WHERE DEPTNO = 'D11'
      ORDER BY PROJNO
```

Restvergleichselemente

Restvergleichselemente erfordern mehr E/A-Aufwand als der Zugriff auf eine Tabelle. Sie können folgende Merkmale aufweisen:

- Sie verwenden korrelierte Unterabfragen.
- Sie verwenden quantifizierte Unterabfragen mit den Klauseln ANY, ALL, SOME oder IN.
- Sie Lesen LONG VARCHAR- oder LOB-Daten, die in einer von der Tabelle getrennten Datei gespeichert sind.

Solche Vergleichselemente werden von den Services für relationale Daten (Relational Data Services) ausgewertet.

Manchmal müssen Vergleichselemente, die nur auf den Index angewandt wurden, erneut angewandt werden, wenn auf die Datenseite zugegriffen wird. Zum Beispiel wenden Zugriffspläne mit OR-Joins oder AND-Joins von Indizes die Vergleichselemente immer ein weiteres Mal als Restvergleichselemente an, wenn auf die Datenseite zugegriffen wird.

Compilerphasen für Abfragen auf föderierte Datenbanken

Pushdown-Analyse für föderierte Datenbanken

Für Abfragen in föderierten Datenbanken führt das Optimierungsprogramm eine Pushdown-Analyse durch, um zu ermitteln, ob eine Operation an einer fernen Datenquelle durchgeführt werden kann. Bei dieser Operation kann es sich um eine Funktion, wie zum Beispiel einen relationalen Operator, eine System- oder Benutzerfunktion oder einen SQL-Operator, zum Beispiel GROUP BY, ORDER BY usw., handeln.

Anmerkung: Obwohl der SQL-Compiler von DB2 über zahlreiche Informationen zur SQL-Unterstützung der Datenquelle verfügt, müssen diese Daten möglicherweise mit der Zeit angepasst werden, da Datenquellen aufgerüstet und/oder angepasst werden können. In solchen Fällen müssen Erweiterungen DB2 durch Ändern der lokalen Kataloginformationen bekannt gemacht werden. Verwenden Sie DDL-Anweisungen von DB2 (z. B. CREATE FUNCTION MAPPING und ALTER SERVER), um den Katalog zu aktualisieren.

Wenn Funktionen nicht an die ferne Datenquelle verschoben werden können, können sie sich erheblich auf die Abfrageleistung auswirken. Betrachten Sie die Konsequenzen für den Fall, dass ein selektives Vergleichselement lokal anstatt an der Datenquelle ausgewertet werden muss. Eine solche Auswertung würde DB2 zwingen, die gesamte Tabelle von der fernen Datenquelle abzurufen und anschließend lokal über das Vergleichselement zu filtern. Netzwerkbegrenzungen und eine bedeutende Tabellengröße könnten zu einer Beeinträchtigung der Leistung führen.

Operatoren, die nicht an die Datenquelle verschoben werden, können sich ebenfalls bedeutsam auf die Abfrageleistung auswirken. Wenn zum Beispiel der Operator GROUP BY ferne Daten lokal zusammenfassen soll, muss DB2 möglicherweise ebenfalls die gesamte Tabelle von der fernen Datenquelle abrufen.

Nehmen Sie zum Beispiel an, dass der Kurzname N1 auf die Tabelle EMPLOYEE an einer Datenquelle unter DB2 für OS/390 oder z/OS verweist. Nehmen Sie weiter an, dass die Tabelle 10.000 Zeilen hat und dass eine der Spalten die Familiennamen (Lastname) und eine andere die Gehälter (Salary) enthält. Betrachten Sie die folgende Anweisung:

```
SELECT LASTNAME, COUNT(*)  FROM N1
   WHERE LASTNAME  > 'B' AND SALARY > 50000
   GROUP BY LASTNAME;
```

In Abhängigkeit davon, ob die Sortierfolgen unter DB2 und DB2 für OS/390 oder z/OS übereinstimmen, werden verschiedene Möglichkeiten in Betracht gezogen:

- Wenn die Sortierfolgen übereinstimmen, kann das Abfragevergleichselement höchstwahrscheinlich per Pushdown-Aktion nach DB2 für OS/390 oder z/OS verschoben werden. Eine Filterung und Gruppierung der Ergebnisse an der Datenquelle ist in der Regel effizienter, als die gesamte Tabelle in DB2 zu kopieren und die Operationen lokal auszuführen. Für die oben gezeigte Abfrage können die Operationen für das Vergleichselement und für den Operator GROUP BY an der Datenquelle stattfinden.
- Wenn die Sortierfolgen nicht übereinstimmen, kann das gesamte Vergleichselement nicht an der Datenquelle ausgewertet werden. Allerdings kann das Optimierungsprogramm entscheiden, den Teil `SALARY > 50000` des Vergleichselements an die Datenquelle zu verschieben. Der Vergleich des Wertebereichs muss immer noch in DB2 ausgeführt werden.

- Wenn die Sortierfolgen übereinstimmen und dem Optimierungsprogramm bekannt ist, dass der lokale DB2-Server sehr schnell ist, kann das Optimierungsprogramm entscheiden, dass die lokale Ausführung der Operation `GROUP BY` in DB2 das günstigste Verfahren (mit dem geringsten Aufwand) ist. Das Vergleichselement wird an der Datenquelle ausgeführt. Dies wäre ein Beispiel für die Pushdown-Analyse in Kombination mit globaler Optimierung.

Im Allgemeinen besteht das Ziel darin, sicherzustellen, dass das Optimierungsprogramm Funktionen und Operatoren an den Datenquellen auswertet. Zahlreiche Faktoren haben Einfluss auf die Entscheidung, ob eine Funktion oder ein SQL-Operator an der fernen Datenquelle ausgewertet wird. Die zu bewertenden Faktoren werden in folgende drei Gruppen klassifiziert:

- Servermerkmale
- Kurznamenmerkmale
- Abfragemerkmale

Servermerkmale mit Auswirkung auf die Pushdown-Möglichkeiten

Bestimmte datenquellenspezifische Faktoren können sich auf die Pushdown-Möglichkeiten auswirken. Diese Faktoren sind im Allgemeinen deswegen vorhanden, weil DB2 eine sehr facettenreiche SQL-Programmversion unterstützt. Diese SQL-Version bietet u. U. mehr Funktionalität als die SQL-Version, die von einem Server unterstützt wird, auf den durch eine Abfrage zugegriffen wird. DB2 kann diesen Funktionsmangel auf dem Datenserver kompensieren, allerdings kann dies dazu führen, dass die Operation in DB2 ausgeführt werden muss.

SQL-Leistungsspektrum: Jede Datenquelle unterstützt eine Variante der SQL-Version und verschiedene Funktionalitätsebenen. Betrachten Sie zum Beispiel die GROUP BY-Liste. Die meisten Datenquellen unterstützen den Operator GROUP BY. Jedoch begrenzen einige die Anzahl von Elementen in der GROUP BY-Liste oder haben Einschränkungen hinsichtlich der Zulässigkeit bestimmter Ausdrücke in der GROUP BY-Liste. Wenn es eine Einschränkung für die ferne Datenquelle gibt, muss DB2 die Operation GROUP BY möglicherweise lokal ausführen.

SQL-Einschränkungen: Jede Datenquelle kann unterschiedliche SQL-Einschränkungen haben. Zum Beispiel fordern einige Datenquellen, dass Parametermarken Werte an ferne SQL-Anweisungen binden. Daher müssen die Einschränkungen für Parametermarken überprüft werden, um sicherzustellen, dass jede Datenquelle solche Bindeverfahren unterstützen kann. Wenn DB2 keine gute Methode zum Binden eines Werts für eine Funktion finden kann, muss diese Funktion lokal ausgewertet werden.

SQL-Begrenzungen: Zwar ermöglicht DB2 die Verwendung größerer Ganzzahlen (Integer) im Vergleich zu den fernen Datenquellen, aber es können keine Werte, die ferne Grenzwerte überschreiten, in Anweisungen eingebettet werden, die an Datenquellen gesendet werden. Daher muss eine Funktion oder ein Operator, die bzw. der mit einer solchen Konstanten operiert, lokal ausgewertet werden.

Serverspezifische Faktoren: In diese Kategorie fallen verschiedene Faktoren. Ein Beispiel ist, ob NULL-Werte als höchster oder niedrigster Wert sortiert werden oder von der Reihenfolge abhängig sind. Wenn Nullwerte in einer Datenquelle anders als von DB2 sortiert werden, können ORDER BY-Operationen für einen Nullwertausdruck nicht fern ausgewertet werden.

Sortierfolge: Das Abrufen von Daten für lokale Sortierungen und Vergleiche setzt in der Regel die Leistung herab. Daher sollten Sie in Erwägung ziehen, die föderierte Datenbank so zu konfigurieren, dass sie dieselbe Sortierfolge wie die Datenquellen verwendet. Wenn Sie eine föderierte Datenbank zur Verwendung derselben Sortierfolge konfigurieren, die von einer Datenquelle verwendet wird, und die Serveroption *collating_sequence* auf den Wert 'Y' setzen, kann das Optimierungsprogramm in Betracht ziehen, viele Abfrageoperationen an die Datenquelle zu verschieben, wenn sich dadurch die Leistung verbessert.

Die folgenden Operationen können an eine Datenquelle verschoben werden, wenn die Sortierfolgen übereinstimmen:

- Vergleiche von Zeichendaten oder numerischen Daten
- Vergleichselemente für Zeichenbereiche
- Sortierungen

Unerwartete Ergebnisse kommen eventuell dann zustande, wenn die Wertigkeit von Nullzeichen zwischen der föderierten Datenbank und der Datenquelle unterschiedlich ist. Vergleichsanweisungen können unerwartete Ergebnisse liefern, wenn Sie Anweisungen an eine Datenquelle übergeben, auf der die Groß-/Kleinschreibung nicht unterschieden wird. Die Wertigkeiten der Zeichen ″I″ und ″i″ sind bei einer Datenquelle ohne Unterscheidung der Groß-/Kleinschreibung identisch. Bei DB2 wird standardmäßig die Groß-/Kleinschreibung beachtet; außerdem werden den Zeichen unterschiedliche Wertigkeiten zugeordnet.

Zur Verbesserung der Leistung lässt der Server der föderierten Datenbank zu, dass Sortierungen und Vergleiche an der Datenquelle stattfinden. Zum Beispiel werden in DB2 für OS/390 oder z/OS Sortierungen, die durch Klauseln ORDER BY definiert werden, mithilfe einer Sortierfolge implementiert, die auf einer EBCDIC-Codepage (Extended Binary-Coded Decimal Interchange Code) basiert. Um den Server der föderierten Datenbank zum Abrufen von Daten einer Datenquelle unter DB2 für OS/390 oder z/OS zu verwenden und die Daten an der Datenquelle mithilfe von Klauseln des Typs ORDER BY zu sortieren, konfigurieren Sie die föderierte Datenbank so, dass sie eine vordefinierte, auf der EBCDIC-Codepage basierte Sortierfolge verwendet.

Wenn die Sortierfolgen der föderierten Datenbank und der Datenquelle voneinander abweichen, ruft DB2 die Daten für die föderierte Datenbank ab. Da Benutzer die Abfrageergebnisse nach der für den Server der föderierten Datenbank definierten Sortierfolge geordnet erwarten, stellt Server der föderierten Datenbank durch ein lokales Sortieren der Daten sicher, dass diese Erwartung erfüllt wird. Übergeben Sie Ihre Abfrage im Durchgriffsmodus (Pass-through) oder definieren Sie die Abfrage in einer Datenquellensicht, wenn Sie die Daten nach der Sortierfolge der Datenquelle geordnet abrufen müssen.

Serveroptionen: Verschiedene Serveroptionen können die Pushdown-Möglichkeiten beeinflussen. Prüfen Sie insbesondere Ihre Einstellungen für die Serveroptionen *collating_sequence*, *varchar_no_trailing_blanks* und *pushdown*.

Faktoren der DB2-Typenzuordnung und -Funktionszuordnung: Die von DB2 definierten lokalen Standardtypenzuordnungen sind dazu vorgesehen, genügend Pufferspeicherplatz für jeden Datentyp einer Datenquelle bereitzustellen, wodurch Datenverlust vermieden wird. Benutzer können Typenzuordnung für eine bestimmte Datenquelle an die Anforderungen bestimmter Anwendungen anzupassen. Wenn Sie zum Beispiel auf eine Spalte des Datentyps DATE einer Oracle-

Datenquelle zugreifen, die standardmäßig dem DB2-Datentyp TIMESTAMP zugeordnet wird, können Sie den lokalen Datentyp in den DB2-Datentyp DATE ändern.

In den folgenden drei Fällen kann DB2 Funktionen kompensieren, die von einer Datenquelle nicht unterstützt werden:

- Die Funktion ist an der fernen Datenquelle nicht vorhanden.
- Die Funktion ist vorhanden, jedoch verletzen die Merkmale des Operanden die Einschränkungen der Funktion. Ein Beispiel für diesen Fall ist der relationale Operator IS NULL. Die meisten Datenquellen unterstützen ihn, aber einige haben möglicherweise Einschränkungen, wie zum Beispiel, dass nur ein Spaltenname auf der linken Seite des Operators IS NULL zulässig ist.
- Die Funktion liefert möglicherweise ein anderes Ergebnis, wenn sie fern ausgewertet wird. Ein Beispiel für diesen Fall ist der Operator '>' (größer als). Für Datenquellen mit abweichenden Sortierfolgen kann der Operator 'größer als' andere Ergebnisse liefern als bei einer lokalen Auswertung durch DB2.

Kurznamenmerkmale mit Auswirkung auf die Pushdown-Möglichkeiten

Die folgenden kurznamenspezifischen Faktoren können sich auf die Pushdown-Möglichkeiten auswirken.

Lokaler Datentyp einer Kurznamenspalte: Stellen Sie sicher, dass der lokale Datentyp einer Spalte nicht verhindert, dass ein Vergleichselement an der Datenquelle ausgewertet werden kann. Verwenden Sie die Standarddatentypenzuordnungen, um einen möglichen Überlauf zu vermeiden. Jedoch wird ein Joinvergleichselement zwischen zwei Spalten unterschiedlicher Längen möglicherweise nicht bei der Datenquelle ausgewertet, deren Joinspalte kürzer ist, in Abhängigkeit davon, wie DB2 die längere Spalte bindet. Dieser Umstand kann sich auf die Anzahl der Möglichkeiten auswirken, die das DB2-Optimierungsprogramm in einer Joinsequenz auswerten kann. Zum Beispiel erhalten Spalten einer Oracle-Datenquelle, die mit dem Datentyp INTEGER bzw. INT erstellt wurden, den Typ NUMBER(38). Eine Kurznamenspalte für diesen Oracle-Datentyp erhält den lokalen Datentyp FLOAT, weil die Werte einer Ganzzahl in DB2 den Bereich von 2**31 bis (-2**31)-1 umfassen, was in etwa dem Typ NUMBER(9) entspricht. In diesem Fall können Joins zwischen einer DB2-Spalte des Typs INTEGER und einer Oracle-Spalte des Typs INTEGER nicht in der DB2-Datenquelle stattfinden (kürzere Joinspalte). Wenn jedoch der Wertebereich dieser Oracle-Spalte des Typs INTEGER in dem DB2-Datentyp INTEGER untergebracht werden kann, ändern Sie den lokalen Datentyp der Spalte mit der Anweisung ALTER NICKNAME, sodass der Join an der DB2-Datenquelle stattfinden kann.

Spaltenoptionen: Verwenden Sie die SQL-Anweisung ALTER NICKNAME, um Spaltenoptionen für Kurznamen hinzuzufügen oder zu ändern.

Verwenden Sie die Option *varchar_no_trailing_blanks* zur Angabe einer Spalte, die keine folgenden Leerzeichen enthält. Der Pushdown-Analyseschritt des Compilers berücksichtigt diese Information bei der Prüfung aller Operationen für Spalten, die so markiert sind. Aufgrund dieser Angabe kann DB2 ein anderes, aber äquivalentes Format eines Vergleichselements generieren, das in einer fernen, an eine Datenquelle gesendete SQL-Anweisung zu verwenden ist. Ein Benutzer bemerkt vielleicht, dass an der Datenquelle ein anderes Vergleichselement ausgewertet wird, das Nettoergebnis sollte jedoch äquivalent sein.

Verwenden Sie die Option *numeric_string* zur Angabe, ob die Werte in der betreffenden Spalte immer Ziffern ohne folgende Leerzeichen sind.

In der folgenden Tabelle werden diese Optionen beschrieben.

Tabelle 64. Spaltenoptionen und zugehörige Einstellungen

Option	Gültige Einstellungen		Standard-einstellung
numeric_string	**'Y'**	Der Wert 'Y' (Yes) gibt an, dass diese Spalte nur Zeichenfolgen mit numerischen Daten enthält. WICHTIG: Wenn die Spalte nur numerische Zeichenfolgen mit folgenden Leerzeichen enthält, geben Sie 'Y' nicht an.	'N'
	'N'	Der Wert 'N' (No) gibt an, dass diese Spalte nicht auf Zeichenfolgen mit numerischen Daten beschränkt ist.	
	Wenn Sie die Spaltenoption numeric_string auf den Wert 'Y' setzen, teilen Sie dem Optimierungsprogramm mit, dass diese Spalte keine Leerzeichen enthält, die einen störenden Einfluss auf das Sortieren der Daten dieser Spalte haben könnten. Diese Option ist in solchen Fällen nützlich, in denen die Sortierfolge einer Datenquelle von der Sortierfolge in DB2 abweicht. Mit dieser Option markierte Spalten werden nicht von der lokalen Auswertung (der Datenquelle) aufgrund unterschiedlicher Sortierfolgen ausgeschlossen.		
varchar_no_trailing_blanks	Gibt an, ob diese Datenquelle eine VARCHAR-Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen verwendet. Für Zeichenfolgen variabler Länge, die keine folgenden Leerzeichen enthalten, liefert die Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen einiger DBMSs die gleichen Ergebnisse wie die Vergleichssemantik von DB2. Wenn Sie sicher sind, dass alle VARCHAR-Spalten von Tabellen und Sichten an einer Datenquelle keine folgenden Leerzeichen enthalten, können Sie in Betracht ziehen, diese Serveroption auf den Wert 'Y' für eine Datenquelle zu setzen. Diese Option wird häufig für Datenquellen von Oracle verwendet. Stellen Sie sicher, dass Sie alle Objekte berücksichtigen, die Kurznamen haben können, einschließlich Sichten.		'N'
	'Y'	Diese Datenquelle verfügt über eine Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen, die der Vergleichssemantik von DB2 ähnlich ist.	
	'N'	Diese Datenquelle verfügt über keine mit DB2 vergleichbare Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen.	

Abfragemerkmale mit Auswirkung auf die Pushdown-Möglichkeiten

Eine Abfrage kann einen SQL-Operator enthalten, der Kurznamen aus verschiedenen Datenquellen mit einbezieht. Die Operation muss in DB2 stattfinden, um die Ergebnisse aus zwei angegebenen Datenquellen zu kombinieren, die nur einen Operator verwenden, wie zum Beispiel einen Gruppenoperator (z. B. UNION). Der Operator kann nicht direkt an der fernen Datenquelle ausgewertet werden.

Richtlinien zur Analyse, wo eine Abfrage für föderierte Datenbanken ausgewertet wird

DB2 stellt zwei Dienstprogramme bereit, damit Sie sehen, wo Abfragen ausgewertet werden:

- Visual Explain. Starten Sie dieses Programm über den Befehl **db2cc**. Dieses Programm dient zum Anzeigen des Zugriffsplandiagramms. Die Ausführungsposition für jeden Operator ist der detaillierten Anzeige für einen Operator zu entnehmen.

Wenn eine Abfrage an eine Datenquelle verschoben wird (Pushdown), sollten Sie einen Operator RETURN sehen. Der Operator RETURN ist ein Standardoperator von DB2. Für eine SELECT-Anweisung, die Daten aus einem Kurznamen auswählt, sehen Sie außerdem einen Operation SHIP. Der Operator SHIP ist für Operationen mit föderierten Datenbanken spezifisch. Er ändert das Servermerkmal des Datenflusses und trennt lokale Operatoren von fernen Operatoren. Die SELECT-Anweisung wird mithilfe der von der Datenquelle unterstützten SQL-Version generiert. Sie kann jede für die Datenquelle gültige Abfrage enthalten.

Wenn eine INSERT-, DELETE- oder UPDATE-Abfrage vollständig an die ferne Datenbank verschoben werden kann (Pushdown), wird eventuell keine SHIP-Anweisung im Zugriffsplan ausgewiesen. Alle fern ausgeführten INSERT-, DELETE- oder UPDATE-Anweisungen werden jedoch für den Operator RETURN gezeigt. Wenn eine Abfrage jedoch nicht vollständig verschoben werden kann, zeigt der Operator SHIP, welche Operationen fern durchgeführt wurden.

- SQL-EXPLAIN. Starten Sie dieses Programm mit dem Befehl **db2expln** oder **dynexpln**. Dieses Programm dient zur Erstellung einer Textausgabe des Zugriffsplans.

Verstehen, warum eine Abfrage an einer Datenquelle oder in DB2 ausgewertet wird

Berücksichtigen Sie die folgenden Schlüsselfragen, wenn Sie Methoden zur Ausweitung der Pushdown-Möglichkeiten untersuchen:

- Warum wird dieses Vergleichselement nicht fern ausgewertet?

  Diese Frage erhebt sich, wenn ein Vergleichselement sehr selektiv ist und daher zum Filtern von Zeilen herangezogen werden und den Netzwerkverkehr verringern könnte. Die ferne Auswertung von Vergleichselementen wirkt sich auch darauf aus, ob ein Join zwischen zwei Tabellen derselben Datenquelle fern ausgewertet werden kann.

  Zu untersuchende Bereiche sind:

  – Vergleichselemente mit Unterabfragen. Enthält dieses Vergleichselement eine Unterabfrage, die sich auf eine andere Datenquelle bezieht? Enthält dieses Vergleichselement eine Unterabfrage mit einem SQL-Operator, der von dieser Datenquelle nicht unterstützt wird? Nicht alle Datenquellen unterstützen Gruppenoperatoren in einem Vergleichselement einer Unterabfrage.
  – Funktionen in Vergleichselementen. Enthält dieses Vergleichselement eine Funktion, die von dieser fernen Datenquelle nicht ausgewertet werden kann? Relationale Operatoren werden als Funktionen klassifiziert.
  – Bindeanforderungen von Vergleichselementen. Erfordert dieses Vergleichselement, wenn es fern ausgewertet werden soll, das Einbinden eines bestimmten Werts? Ist dies der Fall, würde der Wert die SQL-Einschränkungen an dieser Datenquelle verletzen?
  – Globale Optimierung. Das Optimierungsprogramm kann entschieden haben, dass der Aufwand bei lokaler Verarbeitung geringer ist.

- Warum wird der Operator GROUP BY nicht fern ausgewertet?

  Es gibt verschiedene Bereiche, die überprüft werden können:

  – Wird die Eingabe für den Operator GROUP BY fern ausgewertet? Ist dies zu verneinen, untersuchen Sie die Eingabe.
  – Besitzt die Datenquelle Einschränkungen für diesen Operator? Beispiele hierfür sind:

- Begrenzte Anzahl von GROUP BY-Elementen
- Begrenzte Byteanzahl kombinierter GROUP BY-Elemente
- Spaltenspezifikation nur für die GROUP BY-Liste

– Unterstützt die Datenquelle diesen SQL-Operator?
– Globale Optimierung. Das Optimierungsprogramm kann entschieden haben, dass der Aufwand bei lokaler Verarbeitung geringer ist.
– Enthält die Operatorklausel GROUP BY einen Zeichenausdruck? Ist dies der Fall, überprüfen Sie, ob die ferne Datenquelle die gleichen Einstellungen für die Beachtung der Groß-/Kleinschreibung wie DB2 verwendet.

• Warum wird der Gruppenoperator nicht fern ausgewertet?

  Es gibt verschiedene Bereiche, die überprüft werden können:

  – Werden beide Operanden vollständig an derselben Datenquelle ausgewertet? Ist dies nicht der Fall, obwohl es der Fall sein sollte, untersuchen Sie jeden Operanden.
  – Besitzt die Datenquelle Einschränkungen für diesen Gruppenoperator? Sind zum Beispiel große Objekte oder Langfelder gültige Eingaben für diesen speziellen Gruppenoperator?

• Warum wird die Operation ORDER BY nicht fern ausgeführt?

  Betrachten Sie Folgendes:

  – Wird die Eingabe für die Operation ORDER BY fern ausgewertet? Ist dies zu verneinen, untersuchen Sie die Eingabe.
  – Enthält die Klausel ORDER BY einen Zeichenausdruck? Wenn ja, besitzt die ferne Datenquelle nicht die gleiche Sortierfolge oder Einstellung zur Beachtung der Groß-/Kleinschreibung wie DB2?
  – Besitzt die Datenquelle Einschränkungen für diesen Operator? Gibt es beispielsweise eine begrenzte Anzahl von ORDER BY-Elementen? Schränkt die Datenquelle die Spaltenangaben für die ORDER BY-Liste ein?

Generierung von fernem SQL und globale Optimierung in föderierten Datenbanken

Für eine Abfrage auf eine föderierte Datenbank, die relationale Kurznamen verwendet, kann die Zugriffsstrategie vorsehen, die Originalabfrage in eine Gruppe ferner Abfrageeinheiten zu zerlegen und anschließend die Ergebnisse zu kombinieren. Diese Generierung von fernem SQL hilft bei der Herstellung einer global optimalen Zugriffsstrategie für eine Abfrage.

Das Optimierungsprogramm verwendet die Ausgabe der Pushdown-Analyse zur Entscheidung, ob die jeweilige Operation lokal in DB2 oder fern in einer Datenquelle ausgewertet wird. Es stützt die Entscheidung auf die Ausgabe des Aufwandsmodells, das nicht nur den Aufwand für die Auswertung der Operation, sondern auch den Aufwand für die Übertragung der Daten oder Nachrichten zwischen DB2 und den Datenquellen berücksichtigt.

Obwohl das Ziel die Erstellung einer optimierten Abfrage ist, wird die Ausgabe aus der globalen Optimierung und somit auch die Leistung der Abfrage von den folgenden Hauptfaktoren beeinflusst.

• Servermerkmale
• Kurznamenmerkmale

Servermerkmale und Optionen mit Auswirkung auf die globale Optimierung

Die folgenden Serverfaktoren einer Datenquelle können sich auf die globale Optimierung auswirken:

- Relatives Verhältnis der CPU-Geschwindigkeiten

  Geben Sie mithilfe der Serveroption *cpu_ratio* an, wie schnell oder langsam die CPU-Geschwindigkeit der Datenquelle im Verhältnis zur CPU von DB2 ist. Ein niedriger Faktor bedeutet, dass die CPU des Computers der Datenquelle schneller als die CPU des DB2-Computers ist. Wenn der Faktor niedrig ist, zieht das DB2-Optimierungsprogramm mit größerer Wahrscheinlichkeit in Betracht, CPU-intensive Operationen in die Datenquelle zu verschieben (Pushdown).

- Relatives Verhältnis der E/A-Geschwindigkeiten

  Geben Sie mithilfe der Serveroption *io_ratio* an, um wie viel schneller oder langsamer die E/A-Geschwindigkeit des Datenquellensystems im Verhältnis zum DB2-System ist. Ein niedriger Faktor bedeutet, dass die E/A-Geschwindigkeit der Workstation der Datenquelle schneller als die E/A-Geschwindigkeit der DB2-Workstation ist. Wenn der Faktor niedrig ist, zieht das DB2-Optimierungsprogramm in Betracht, E/A-intensive Operationen in die Datenquelle zu verschieben (Pushdown).

- Übertragungsgeschwindigkeit zwischen DB2 und der Datenquelle

  Geben Sie mithilfe der Serveroption *comm_rate* die Netzwerkkapazität an. Langsame Geschwindigkeiten, die eine langsame Netzkommunikation zwischen DB2 und der Datenquelle bedeuten, veranlassen das DB2-Optimierungsprogramm, die Anzahl der Nachrichten zu reduzieren, die an die und von dieser Datenquelle gesendet werden. Wenn die Geschwindigkeit auf den Wert 0 gesetzt wird, erstellt das Optimierungsprogramm einen Zugriffsplan, der nur minimalen Netzwerkverkehr erfordert.

- Sortierfolge der Datenquelle

  Geben Sie mithilfe der Serveroption *collating_sequence* an, ob die Sortierfolge einer Datenquelle mit der lokalen DB2-Datenbanksortierfolge übereinstimmt. Wenn die Option nicht auf den Wert 'Y' gesetzt ist, betrachtet das Optimierungsprogramm die von dieser Datenquelle abgerufenen Daten als unsortiert.

- Ferne Planhinweise

  Geben Sie mithilfe der Serveroption *plan_hints* an, dass Planhinweise generiert werden oder an einer Datenquelle verwendet werden sollen. Standardmäßig sendet DB2 keine Planhinweise an die Datenquelle.

  Planhinweise sind Anweisungsfragmente, die zusätzliche Informationen für Optimierungsprogramme von Datenquellen bereitstellen. Für bestimmte Abfragen können diese Informationen die Leistung verbessern. Die Planhinweise können das Optimierungsprogramm der Datenquelle bei der Entscheidung unterstützen, ob ein Index, und wenn ja, welcher, zu verwenden ist oder nach welcher Reihenfolge beim Join von Tabellen vorzugehen ist.

  Wenn Planhinweise aktiviert sind, enthält die Abfrage, die an die Datenquelle gesendet wird, zusätzliche Informationen. Zum Beispiel könnte eine Anweisung, die an ein Oracle-Optimierungsprogramm mit Planhinweisen gesendet wird, folgendermaßen aussehen:

  ```
  SELECT /*+ INDEX (tabelle1, t1index)*/
           col1
           FROM tabelle1
  ```

  Der Planhinweis ist hier die Zeichenfolge `/*+ INDEX (tabelle1, t1index)*/`.

- Informationen in der Wissensbasis des DB2-Optimierungsprogramms

  DB2 verfügt über eine Wissensbasis für das Optimierungsprogramm, die Daten über native Datenquellen enthält. Das DB2-Optimierungsprogramm generiert keine fernen Zugriffspläne, die nicht von bestimmten Datenbankverwaltungs-

systemen (DBMSs) generiert werden können. D. h., DB2 vermeidet die Generierung von Plänen, die von Optimierungsprogrammen an fernen Datenquellen nicht verstanden bzw. akzeptiert werden.

Kurznamenmerkmale mit Auswirkung auf die globale Optimierung

Die folgenden kurznamenspezifischen Faktoren können sich auf die globale Optimierung auswirken.

Indexinformationen: DB2 kann Informationen über Indizes an Datenquellen zur Optimierung von Abfragen heranziehen. Daher ist es wichtig, dass die für DB2 verfügbaren Indexinformationen aktuell sind. Die Indexinformationen für Kurznamen werden zu Anfang bei der Erstellung des Kurznamens erfasst. Indexinformationen werden für Sichtkurznamen nicht gesammelt.

Erstellen von Indexspezifikationen für Kurznamen: Sie können eine Indexspezifikation für einen Kurznamen erstellen. Indexspezifikationen erstellen eine Indexdefinition (keinen tatsächlichen Index) im Katalog, die vom DB2-Optimierungsprogramm verwendet werden kann. Indexspezifikationen werden mit der Anweisung CREATE INDEX SPECIFICATION ONLY erstellt. Die Syntax für die Erstellung einer Indexspezifikation für einen Kurznamen ist der Syntax zur Erstellung eines Index für eine lokale Tabelle ähnlich.

Ziehen Sie die Erstellung von Indexspezifikationen in folgenden Fällen in Betracht:

- DB2 kann während der Kurznamenerstellung keine Indexinformationen von einer Datenquelle abrufen.
- Sie wünschen einen Index für einen Sichtkurznamen.
- Sie möchten das DB2-Optimierungsprogramm zur Verwendung eines bestimmten Kurznamens als innere Tabelle eines Joins mit Verschachtelungsschleife veranlassen. Der Benutzer kann einen Index für die Joinspalte erstellen, falls keiner vorhanden ist.

Überlegen Sie vor dem Ausführen von Anweisungen CREATE INDEX für einen Kurznamen einer Sicht, ob Sie den Index benötigen. Wenn die Sicht eine einfache SELECT-Abfrage auf eine Tabelle mit einem Index ist, kann die Erstellung lokaler Indizes für den Kurznamen, die mit den Indizes für die Tabelle an der Datenquelle übereinstimmen, die Abfrageleistung erheblich erhöhen. Wenn Indizes jedoch lokal für Sichten erstellt werden, die keine einfachen SELECT-Anweisungen enthalten, zum Beispiel eine Sicht, die durch den Join zweier Tabellen erstellt wird, kann die Abfrageleistung sinken. Wenn Sie zum Beispiel einen Index für eine Sicht erstellen, die auf dem Join zweier Tabellen basiert, kann das Optimierungsprogramm diese Sicht möglicherweise als inneres Element in einem Join mit Verschachtelungsschleife (Nested Loop Join) wählen. Die Abfrage wird eine geringe Leistung erzielen, weil in diesem Fall der Join mehrere Male ausgewertet wird. Eine Alternative wäre, Kurznamen für jede einzelne Tabelle, auf die in der Sicht der Datenquelle verwiesen wird, zu erstellen und eine lokale Sicht unter DB2 zu definieren, die auf beide Kurznamen verweist.

Katalogstatistiken: Katalogstatistiken beschreiben die allgemeine Größe von Kurznamen und den Wertebereich in den zugehörigen Spalten. Das Optimierungsprogramm verwendet diese Statistikdaten, wenn es den Pfad mit dem geringsten Aufwand zur Verarbeitung von Abfragen berechnet, die Kurznamen enthalten. Die statistischen Daten über Kurznamen werden in den gleichen Katalogsichten wie Tabellenstatistiken gespeichert.

Obwohl DB2 die in einer Datenquelle gespeicherten statistischen Daten abrufen kann, kann es Aktualisierungen an vorhandenen statistischen Daten in Datenquellen nicht automatisch erkennen. Überdies kann DB2 Änderungen in Objektdefinitionen oder strukturelle Änderungen, wie das Hinzufügen einer Spalte, an Objekten von Datenquellen nicht verarbeiten. Wenn die Statistik- bzw. Strukturdaten für ein Objekt geändert wurden, haben Sie zwei Möglichkeiten:

- Führen Sie an der Datenquelle das Programm aus, das dem Dienstprogramm RUNSTATS entspricht. Löschen Sie anschließend den aktuellen Kurznamen und erstellen Sie ihn neu. Verwenden Sie diese Methode, wenn sich Strukturinformationen geändert haben.
- Aktualisieren Sie die Statistiken in der Sicht SYSSTAT.TABLES manuell. Diese Methode erfordert zwar weniger Schritte, funktioniert jedoch nicht, wenn sich Strukturinformationen geändert haben.

Globale Analyse von Abfragen auf föderierte Datenbanken

Die beiden folgenden Dienstprogramme sind zum Anzeigen globaler Zugriffspläne verfügbar:

- Visual Explain. Starten Sie dieses Programm über die Steuerzentrale oder führen Sie den Befehl *db2cc* aus, der die Steuerzentrale startet. Verwenden Sie Visual Explain zum Anzeigen des Zugriffsplandiagramms für eine Abfrage. Die Ausführungsposition für jeden Operator ist der detaillierten Anzeige für einen Operator zu entnehmen. Sie können darüber hinaus die ferne SQL-Anweisung, die für die jeweilige Datenquelle generiert wurde, je nach Art der Abfrage dem Operator SHIP oder RETURN entnehmen. Durch eine Untersuchung der Details für jeden Operator können Sie die Anzahl der Zeilen feststellen, die vom DB2-Optimierungsprogramm als Eingabe für den jeweiligen Operator und als Ausgabe aus diesem Operator geschätzt wird. Weiterhin können Sie den für die Ausführung des jeweiligen Operators geschätzten Aufwand, einschließlich des Kommunikationsaufwands, ablesen.
- SQL-EXPLAIN. Starten Sie das Programm mit dem Befehl *db2expln* oder *dynexpln*. SQL-EXPLAIN dient zur Erstellung einer Textausgabe des Zugriffsplans. Obwohl SQL-EXPLAIN keine Informationen über den Aufwand bietet, können Sie der Ausgabe den Zugriffsplan entnehmen, der von dem fernen Optimierungsprogramm für die Datenquellen generiert wurde, die von der fernen EXPLAIN-Funktion unterstützt werden.

Verstehen der Entscheidungen der DB2-Optimierung

Berücksichtigen Sie bei der Untersuchung von Leistungsverbesserungen die folgenden Fragen und Schlüsselbereiche der Optimierung:

- Warum wird ein Join zwischen zwei Kurznamen derselben Datenquelle nicht fern an der Datenquelle ausgewertet?

  Zu untersuchende Bereiche sind:

  – Joinoperationen. Werden sie von der Datenquelle unterstützt?
  – Joinvergleichselemente. Können die Joinvergleichselemente an der fernen Datenquelle ausgewertet werden? Wenn dies zu verneinen ist, untersuchen Sie das Joinvergleichselement.
  – Anzahl der Zeilen im Joinergebnis (mit Visual Explain). Ergibt sich aus dem Join eine wesentlich größere Gruppe von Zeilen als durch die Kombination der beiden Kurznamen? Ist die jeweilige Anzahl sinnvoll? Wenn dies zu verneinen ist, ziehen Sie eine manuelle Aktualisierung der Kurznamenstatistik (SYSSTAT.TABLES) in Betracht.
- Warum wird der Operator GROUP BY nicht fern ausgewertet?

Zu untersuchende Bereiche sind:

- Operatorsyntax. Stellen Sie sicher, dass der Operator an der fernen Datenquelle ausgewertet werden kann.
- Anzahl von Zeilen. Prüfen Sie die geschätzte Anzahl von Zeilen in der Eingabe und Ausgabe des Operators GROUP BY mithilfe von Visual Explain. Liegt die eine Anzahl sehr nahe an der anderen? Ist dies zu bejahen, betrachtet das DB2-Optimierungsprogramm es als effizienter, diesen Operator GROUP BY lokal auszuwerten. Ist die jeweilige Anzahl zudem sinnvoll? Wenn dies zu verneinen ist, ziehen Sie eine manuelle Aktualisierung der Kurznamenstatistik (SYSSTAT.TABLES) in Betracht.

- Warum wird die Anweisung nicht vollständig durch die ferne Datenquelle ausgewertet?

  Das DB2-Optimierungsprogramm führt eine aufwandsorientierte Optimierung durch. Auch wenn die Pushdown-Analyse ergibt, dass jeder Operator an der fernen Datenquelle ausgewertet werden kann, stützt sich das Optimierungsprogramm bei der Generierung eines global optimalen Plans auf die Aufwandsschätzung. Es gibt eine Vielzahl von Faktoren, die in diesen Plan einfließen können. Es ist beispielsweise möglich, dass eine ferne Datenquelle zwar jede einzelne Operation in der ursprünglichen Abfrage ausführen kann, aber die CPU-Geschwindigkeit der Datenquelle wesentlich langsamer ist als die CPU-Geschwindigkeit des DB2-Systems, sodass es vorteilhafter ist, die Operationen in DB2 lokal auszuführen. Wenn die Ergebnisse nicht zufrieden stellend sind, überprüfen Sie die Serverstatistikdaten in SYSCAT.SERVEROPTIONS.

- Warum zeigt ein vom Optimierungsprogramm generierter Plan, der vollständig an einer fernen Datenquelle ausgewertet wird, eine wesentlich schlechtere Leistung als die Originalabfrage bei direkter Ausführung an der fernen Datenquelle?

  Zu untersuchende Bereiche sind:

  - Die vom DB2-Optimierungsprogramm generierte ferne SQL-Anweisung. Überprüfen Sie, ob sie mit der Originalabfrage identisch ist. Suchen Sie nach Änderungen in der Reihenfolge der Vergleichselemente. Ein gutes Abfrageoptimierungsprogramm sollte nicht von der Reihenfolge der Vergleichselemente einer Abfrage abhängig sein. Leider sind nicht alle DBMS-Optimierungsprogramme identisch, sodass das Optimierungsprogramm der fernen Datenquelle vielleicht aufgrund der Reihenfolge der Vergleichselemente in der Eingabe einen anderen Plan generiert. Wenn dies der Fall ist, liegt das Problem bei dem fernen Optimierungsprogramm. Sie können in diesem Fall entweder eine Änderung der Reihenfolge der Vergleichselemente in der Eingabe für DB2 in Betracht ziehen oder die Serviceorganisation der fernen Datenquelle um Unterstützung bitten.

    Prüfen Sie darüber hinaus auf Ersetzungen von Vergleichselementen. Ein gutes Abfrageoptimierungsprogramm sollte nicht von der Ersetzung äquivalenter Vergleichselemente abhängig sein. Leider sind nicht alle DBMS-Optimierungsprogramme identisch, sodass das Optimierungsprogramm der fernen Datenquelle vielleicht aufgrund des Vergleichselements in der Eingabe einen anderen Plan generiert. Zum Beispiel können einige Optimierungsprogramme keine Anweisungen für Vergleichselemente unter Verwendung der Transitivität generieren.
  - Die Anzahl der zurückgegebenen Zeilen. Diese Anzahl kann mithilfe von Visual Explain festgestellt werden. Wenn die Abfrage eine große Anzahl von Zeilen zurückgibt, kann der Netzwerkverkehr zum potenziellen Engpass werden.

– Zusätzliche Funktionen. Enthält die ferne SQL-Anweisung im Vergleich zur Originalabfrage zusätzliche Funktionen? Es können einige zusätzliche Funktionen zur Umsetzung von Datentypen generiert werden. Stellen Sie sicher, dass diese erforderlich sind.

Datenzugriffsmethoden

Bei der Kompilierung einer SQL- oder XQuery-Anweisung schätzt das Abfrageoptimierungsprogramm den Ausführungsaufwand der verschiedenen Methoden ab, die die Anforderung erfüllen würden. Auf der Grundlage dieser Schätzungen wählt das Optimierungsprogramm einen optimalen Zugriffsplan aus. Ein *Zugriffsplan* gibt die Reihenfolge von Operationen an, die erforderlich sind, um eine SQL- oder XQuery-Anweisung auszuführen. Wenn ein Anwendungsprogramm gebunden wird, wird ein *Paket* erstellt. Dieses Paket enthält Zugriffspläne für alle statischen SQL- und XQuery-Anweisungen in dem entsprechenden Anwendungsprogramm. Die Zugriffspläne für dynamische SQL- und XQuery-Anweisungen werden zum Zeitpunkt der Ausführung der Anwendung erstellt.

Es gibt zwei Methoden für den Zugriff auf Daten in einer Tabelle:

- Sequenzielles Durchsuchen der gesamten Tabelle
- Lokalisieren bestimmter Tabellenzeilen durch einen vorherigen Zugriff auf einen Index für die Tabelle

Zur Erstellung der von der Abfrage angeforderten Ergebnisse werden Zeilen in Abhängigkeit von den Bedingungen des Vergleichselements ausgewählt, das in der Regel in einer WHERE-Klausel angegeben wird. Die ausgewählten Zeilen in den Tabellen, auf die zugegriffen wird, werden verknüpft, um die Ergebnismenge zu erstellen, und die Ergebnismenge wird unter Umständen weiter verarbeitet, indem die Ausgabe gruppiert oder sortiert wird.

Datenzugriff über Indexsuchen

Als *Indexsuche* wird der Vorgang bezeichnet, bei dem der Datenbankmanager zu folgenden Zwecken auf einen Index zugreift:

- Eingrenzen der Menge der den Bedingungen entsprechenden Zeilen (durch Durchsuchen der Zeilen in einem bestimmten Bereich des Index) vor dem Zugriff auf die Basistabelle. Der *Suchbereich* des Index (der Start- und der Stoppunkt der Suche) wird durch die Werte in der Abfrage festgelegt, mit denen Indexspalten verglichen werden.
- Sortieren der Ausgabe.
- Direktes Abrufen der angeforderten Spaltendaten. Wenn sich alle angeforderten Daten im Index befinden, braucht kein Zugriff auf die Basistabelle stattzufinden. Dies wird als *reiner Indexzugriff* bezeichnet.

Wenn Indizes mit der Option ALLOW REVERSE SCANS erstellt wurden, können Indexsuchen auch in entgegengesetzter Richtung zu der Richtung durchgeführt werden, in der die Indizes definiert wurden.

Anmerkung: Das Optimierungsprogramm wählt eine Tabellensuche, wenn kein entsprechender Index erstellt wurde oder eine Indexsuche aufwendiger wäre. Eine Indexsuche kann aufwendiger sein, wenn die Tabelle klein ist, das Indexclusterverhältnis niedrig ist oder die Abfrage die meisten der Tabellenzeilen anfordert. Ob der Zugriffsplan eine Tabellensuche oder eine Indexsuche verwendet, lässt sich mithilfe der EXPLAIN-Einrichtung feststellen.

Indexsuchen zur Begrenzung eines Bereichs

Zur Bestimmung, ob ein Index für eine bestimmte Abfrage verwendet werden kann, wertet das Optimierungsprogramm jede Spalte des Index beginnend bei der ersten Spalte aus, um zu überprüfen, ob sie zur Erfüllung von Gleichheitsvergleichselementen und anderen Vergleichselementen in der WHERE-Klausel verwendet werden kann. Ein *Vergleichselement* ist ein Element einer Suchbedingung in einer Klausel WHERE, das eine Vergleichsoperation definiert oder impliziert. Vergleichselemente können zur Begrenzung des Bereichs einer Indexsuche zu folgenden Zwecken verwendet werden:

- Testen auf Gleichheit mit einer Konstante, einer Hostvariablen, einem Ausdruck, der zu einer Konstanten ausgewertet wird, oder einem Schlüsselwort.
- Testen auf „IS NULL" oder „IS NOT NULL".
- Testen auf Gleichheit mit einer einfachen Unterabfrage, d. h. mit einer Unterabfrage, die weder ANY, ALL oder SOME noch einen Verweis über eine korrelierte Spalte auf den unmittelbar übergeordneten Abfrageblock (d. h. auf die SELECT-Anweisung, für die diese Unterabfrage ein Subselect ist) enthält.
- Testen auf strenge oder einschließende Ungleichheit.

Die folgenden Beispiele veranschaulichen, wann ein Index zur Begrenzung eines Bereichs verwendet werden könnte:

- Betrachten Sie einen Index mit der folgenden Definition:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

  In diesem Fall könnten die folgenden Vergleichselemente zur Begrenzung des Bereichs bei der Suche im Index IX1 verwendet werden:

```
WHERE  NAME = :hv1
  AND  DEPT = :hv2
```

  oder

```
WHERE  MGR  = :hv1
  AND  NAME = :hv2
  AND  DEPT = :hv3
```

  Beachten Sie, dass in der zweiten WHERE-Klausel die Vergleichselemente nicht in derselben Reihenfolge angegeben werden müssen, in der die Schlüsselspalten im Index erscheinen. In den Beispielen werden zwar Hostvariablen (hv = Hostvariable) verwendet, jedoch hätten andere Variablen wie Parametermarken, Ausdrücke oder Konstanten dieselbe Wirkung.

- Betrachten Sie einen einzelnen Index, der mit dem Parameter ALLOW REVERSE SCANS erstellt wurde. Solche Indizes unterstützen Suchoperationen in der Richtung, die bei der Erstellung des Index definiert wurde, und Suchoperationen in entgegengesetzter oder umgekehrter Richtung. Die Anweisung könnte etwa wie folgt aussehen:

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

  In diesem Fall wird der Index (`iname`) nach den absteigenden (DESCending) Werten der Spalte `cname` gebildet. Durch Zulassen von umgekehrten Suchoperationen kann eine Suche in aufsteigender Folge durchgeführt werden, obwohl der Index für die Spalte für Suchoperationen in absteigender Folge definiert ist. Die tatsächliche Verwendung des Index in beiden Richtungen wird nicht von Ihnen, sondern vom Optimierungsprogramm bei der Erstellung und Auswahl von Zugriffsplänen gesteuert.

In der folgenden Klausel WHERE würden nur die Vergleichselemente für NAME und DEPT verwendet, um den Bereich der Indexsuche einzugrenzen, jedoch nicht die Vergleichselemente für SALARY und YEARS:

```
WHERE  NAME   = :hv1
  AND  DEPT   = :hv2
  AND  SALARY = :hv4
  AND  YEARS  = :hv5
```

Der Grund dafür ist der, dass es eine Schlüsselspalte (MGR) gibt, die diese Spalten von den ersten beiden Indexschlüsselspalten trennt, sodass die Reihenfolge nicht gewährleistet wäre. Wenn aber der Bereich einmal durch die Vergleichselemente NAME = :hv1 und DEPT = :hv2 festgelegt ist, können die verbleibenden Vergleichselemente an den verbleibenden Indexschlüsselspalten ausgewertet werden.

Indexsuchen zum Testen von Ungleichheit

Bestimmte Ungleichheitsvergleichselemente können den Bereich einer Indexsuche begrenzen. Es gibt zwei Typen von Ungleichheitsvergleichselementen:

- Vergleichselemente für strenge Ungleichheit

  Die Operatoren für strenge (ausschließende) Ungleichheit, die für bereichsbegrenzende Vergleichselemente verwendet werden, sind > (größer als) und < (kleiner als).

  Nur eine Spalte mit Vergleichselementen für strenge Ungleichheit wird zur Begrenzung des Bereichs für eine Indexsuche berücksichtigt. Im folgenden Beispiel können die Vergleichselemente für die Spalten NAME und DEPT verwendet werden, um den Bereich einzugrenzen, aber das Vergleichselement für die Spalte MGR nicht.

  ```
  WHERE  NAME   = :hv1
    AND  DEPT   > :hv2
    AND  DEPT   < :hv3
    AND  MGR    < :hv4
  ```

- Vergleichselemente für einschließende Ungleichheit

  Die folgenden Operatoren für einschließende Ungleichheit können für Vergleichselemente verwendet werden, die einen Bereich begrenzen:

  – >= und <=
  – BETWEEN
  – LIKE

  Zur Begrenzung eines Bereichs für eine Indexsuche werden mehrere Spalten mit Vergleichselementen einschließender Ungleichheit berücksichtigt. Im folgenden Beispiel können alle Vergleichselemente zur Eingrenzung des Bereichs der Indexsuche verwendet werden:

  ```
  WHERE  NAME   = :hv1
    AND  DEPT  >= :hv2
    AND  DEPT  <= :hv3
    AND  MGR   <= :hv4
  ```

  Zur weiteren Verdeutlichung dieses Beispiels seien die folgenden Werte angenommen: `:hv2 = 404`, `:hv3 = 406` und `:hv4 = 12345`. Der Datenbankmanager durchsucht den Index für die Abteilungen (DEPT) 404 und 405 ganz, bricht aber die Suche in Abteilung 406 ab, wenn er auf den ersten Manager trifft, der eine Personalnummer (Spalte MGR) über dem Wert 12345 hat.

Indexsuchen zum Sortieren von Daten

Wenn die Abfrage eine sortierte Ausgabe erfordert, kann ein Index zum Sortieren der Daten verwendet werden, wenn die ordnenden Spalten nacheinander, angefangen bei der ersten Indexschlüsselspalte, im Index vertreten sind. Das Ordnen oder Sortieren kann aus Operationen wie ORDER BY, DISTINCT, GROUP BY, Unterabfrage mit „= ANY", Unterabfrage mit „> ALL", Unterabfrage mit „> ALL", INTERSECT bzw. EXCEPT sowie UNION resultieren. Eine Ausnahme ist der Fall, wenn die Indexschlüsselspalten auf Gleichheit mit „konstanten Werten" verglichen werden, das heißt mit einem beliebigen Ausdruck, der zu einer Konstanten ausgewertet wird. In diesem Fall können die ordnenden Spalten andere als die ersten Indexschlüsselspalten sein.

Betrachten Sie die folgende Abfrage:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

Für diese Abfrage könnte der Index verwendet werden, um die Zeilen zu sortieren, da die Spalten NAME und DEPT immer dieselben Werte haben und so geordnet sind. Das heißt, die gezeigten Klauseln WHERE und ORDER BY sind äquivalent mit folgenden Klauseln:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY NAME, DEPT, MGR
```

Ein eindeutiger Index kann auch zum Verkürzen einer Sortieranforderung verwendet werden. Betrachten Sie die folgende Indexdefinition und die ORDER BY-Klausel:

```
UNIQUE INDEX IX0:  PROJNO  ASC
SELECT PROJNO, PROJNAME, DEPTNO
  FROM PROJECT
ORDER BY PROJNO, PROJNAME
```

Eine zusätzliche Sortierung über die Spalte PROJNAME ist nicht erforderlich, da der Index IX0 bereits sicherstellt, dass die Werte der Spalte PROJNO eindeutig sind. Diese Eindeutigkeit sorgt dafür, dass es nur einen Wert für PROJNAME für jeden Wert von PROJNO gibt.

Arten des Indexzugriffs

In einigen Fällen kann das Optimierungsprogramm feststellen, dass alle Daten, die eine Abfrage aus einer Tabelle anfordert, aus einem Index für die Tabelle abgerufen werden können. In anderen Fällen kann das Optimierungsprogramm auch mehrere Indizes für den Zugriff auf Tabellen verwenden. Im Fall von Bereichsclustertabellen kann der Zugriff auf Daten über einen „virtuellen" Index erfolgen, der die Positionen von Datensätzen berechnet.

Reiner Indexzugriff

In einigen Fällen können alle angeforderten Daten aus dem Index abgerufen werden, ohne auf die Tabelle zuzugreifen. Dies wird als *reiner Indexzugriff* bezeichnet.

Betrachten Sie die folgende Indexdefinition zur Illustration des reinen Indexzugriffs:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

Die folgende Abfrage kann nur durch den Zugriff auf den Index ohne Lesen der Basistabelle erfüllt werden:

```
SELECT NAME, DEPT, MGR, SALARY
  FROM EMPLOYEE
 WHERE NAME = 'SMITH'
```

Häufig sind erforderliche Spalten jedoch nicht im Index vertreten. Um die Daten für diese Spalten abzurufen, müssen Tabellenzeilen gelesen werden. Damit das Optimierungsprogramm die Möglichkeit hat, einen reinen Indexzugriff zu wählen, erstellen Sie einen eindeutigen Index mit INCLUDE-Spalten. Betrachten Sie zum Beispiel die folgende Indexdefinition:

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
   INCLUDE (DEPT, MGR, SALARY, YEARS)
```

Dieser Index gewährleistet die Eindeutigkeit der Datenwerte in der Spalte NAME und speichert und pflegt darüber hinaus Daten für die Spalten DEPT, MGR, SALARY und YEARS. Dadurch kann die folgende Abfrage nur durch einen Zugriff auf den Index erfüllt werden:

```
SELECT NAME, DEPT, MGR, SALARY
   FROM EMPLOYEE
   WHERE NAME='SMITH'
```

Wenn Sie das Hinzufügen von INCLUDE-Spalten für einen Index in Betracht ziehen, müssen Sie jedoch abwägen, ob der zusätzliche Speicherplatzbedarf und der Pflegeaufwand gerechtfertigt sind. Wenn Abfragen, die durch einen solchen Index erfüllt werden können, nur selten ausgeführt werden, ist der Zusatzaufwand möglicherweise nicht gerechtfertigt.

Zugriff auf mehrere Indizes

Das Optimierungsprogramm kann entscheiden, mehrere Indizes für dieselbe Tabelle zu durchsuchen, um die Vergleichselemente einer WHERE-Klausel zu erfüllen. Betrachten Sie zum Beispiel die beiden folgenden Indexdefinitionen:

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

Die folgenden Vergleichselemente können durch die Verwendung der beiden Indizes erfüllt werden:

```
WHERE DEPT = :hv1
  OR (JOB   = :hv2
  AND YEARS >= :hv3)
```

Das Durchsuchen des Index IX2 liefert eine Liste von Zeilen-IDs (RIDs), die das Vergleichselement `DEPT = :hv1` erfüllen. Das Durchsuchen des Index IX3 liefert eine Liste der RIDs, die das Vergleichselement `JOB = :hv2 AND YEARS >= :hv3` erfüllen. Diese beiden Listen von RIDs werden kombiniert und doppelte Werte entfernt, bevor auf die Tabelle zugegriffen wird. Diese Methode wird als *logischer OR-Join von Indizes* (Index ORing) bezeichnet.

Der OR-Join von Indizes kann auch für Vergleichselemente mit der Klausel IN wie in folgendem Beispiel verwendet werden:

```
WHERE DEPT IN (:hv1, :hv2, :hv3)
```

Während der Zweck des OR-Joins von Indizes in der Eliminierung doppelter RIDs liegt, besteht das Ziel des *logischen AND-Joins von Indizes* (Index ANDing) darin,

gemeinsame RIDs zu finden. Der AND-Join von Indizes kann auftreten, wenn Anwendungen mehrere Indizes für entsprechende Spalten innerhalb derselben Tabelle erstellen und eine Abfrage mit mehreren AND-Vergleichselementen für die Tabelle ausgeführt wird. Mehrere Indexsuchen für alle mit Indizes versehenen Spalten in einer solchen Abfrage ergeben Werte, mit denen in einem Hashverfahren Bitzuordnungen erstellt werden. Die zweite Bitzuordnung wird zur Prüfung der ersten Bitzuordnung verwendet, um die gesuchten Zeilen zu generieren, die zur Erstellung der endgültigen zurückzugebenden Datenmenge abgerufen werden.

Betrachten Sie zum Beispiel die beiden folgenden Indexdefinitionen:

```
INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

Die folgenden Vergleichselemente könnten mithilfe dieser beiden Indizes aufgelöst werden:

```
WHERE SALARY BETWEEN 20000 AND 30000
AND COMM BETWEEN 1000 AND 3000
```

In diesem Beispiel generiert das Durchsuchen des Index `IX4` eine Bitzuordnung, die das Vergleichselement `SALARY BETWEEN 20000 AND 30000` erfüllt. Das Durchsuchen von `IX5` und Prüfen gegen die Bitzuordnung für `IX4` liefert eine Liste von RIDs, die beide Vergleichselemente erfüllen. Dies wird als „dynamischer AND-Join über Bitzuordnungen" bezeichnet. Diese Methode wird nur angewandt, sofern die Tabelle ausreichend Kardinalität hat und die Spalten genügend Werte in dem gesuchten Bereich oder genügend Duplizität haben, wenn Gleichheitsvergleichselemente verwendet werden.

Zur Umsetzung der Leistungsvorteile dynamischer Bitzuordnungen beim Durchsuchen mehrerer Indizes kann es notwendig sein, den Wert des Konfigurationsparameters für die Größe des Sortierspeichers (*sortheap*) der Datenbank und des Konfigurationsparameters für den Schwellenwert für Sortierspeicher (*sheapthres*) des Datenbankmanagers zu ändern.

Bei Verwendung dynamischer Bitzuordnungen in Zugriffsplänen wird zusätzlicher Sortierspeicher benötigt. Wenn der Wert von *sheapthres* relativ nahe am Wert von *sortheap* liegt (d. h. weniger als ein Faktor von 2- oder 3-mal pro gleichzeitiger Abfrage), steht dynamischen Bitzuordnungen mit Zugriff über mehrere Indizes wesentlich weniger Speicher zur Verfügung als das Optimierungsprogramm veranschlagt hat. Die Lösung besteht darin, den Wert von *sheapthres* relativ zu *sortheap* zu erhöhen.

Anmerkung: Das Optimierungsprogramm verwendet beim Zugriff auf eine einzelne Tabelle keine Kombination aus AND- und OR-Joins von Indizes.

Indexzugriff in Bereichsclustertabellen

Im Gegensatz zu Standardtabellen benötigt eine Bereichsclustertabelle keinen physischen Index, der einer Zeile einen Schlüsselwert wie in einem traditionellen B-Baumstrukturindex zuordnet. Stattdessen greift er auf die sequenzielle Anordnung des Spaltenwertebereichs zurück und verwendet eine Zuordnungsfunktion, um die Position einer bestimmten Zeile in einer Tabelle zu generieren. Im einfachsten Beispiel einer solchen Zuordnung ist der erste Schlüsselwert im Bereich die erste Zeile in der Tabelle, der zweite Wert im Bereich die zweite Zeile in der Tabelle usw.

Das Optimierungsprogramm verwendet das Merkmal des Bereichsclustering der Tabelle, um Zugriffspläne auf der Basis eines perfekt angeordneten Index zu generieren, der lediglich den Aufwand zur Berechnung der Bereichsclusterfunktion erfordert. Das Clustering von Zeilen innerhalb der Tabelle ist gewährleistet, da Bereichsclustertabellen ihre Reihenfolge nach den ursprünglichen Schlüsselwerten beibehalten.

Indexzugriff und Clusterverhältnisse

Bei der Auswahl des Zugriffsplans schätzt das Optimierungsprogramm die Anzahl der E/A-Operationen ab, die zum Einlesen der benötigten Seiten von der Platte in den Pufferpool erforderlich sind. Diese 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 E/A-Operationen anfallen.

Für Indexsuchen wird das Optimierungsprogramm durch Informationen aus den Systemkatalogtabellen (SYSCAT.INDEXES) bei der Abschätzung des Ein-/Ausgabeaufwands zum Lesen von Datenseiten in den Pufferpool unterstützt. Dabei werden Informationen aus den folgenden Spalten der Tabelle SYSCAT.INDEXES verwendet:

- Die Informationen der Spalte CLUSTERRATIO geben den Grad an, zu dem die Tabellendaten in Relation zu diesem Index in Clustern zusammengefasst sind (Clusterbildung). Je höher der Wert, desto besser sind die Zeilen in der Reihenfolge des Indexschlüssels geordnet. Wenn Tabellenzeilen nahe an der Reihenfolge des Indexschlüssels vorliegen, können Zeilen von einer Datenseite gelesen werden, während sich die Seite im Puffer befindet. Wenn der Wert dieser Spalte -1 ist, verwendet das Optimierungsprogramm die Informationen der Spalten PAGE_FETCH_PAIRS und CLUSTERFACTOR, wenn diese verfügbar sind.
- Die Spalte PAGE_FETCH_PAIRS enthält Paare von Zahlen, die zusammen mit CLUSTERFACTOR-Informationen jeweils ein Modell für die Anzahl der E/A-Operationen zum Lesen der Datenseiten in Pufferpools verschiedener Größen angeben. Für diese Spalten werden Daten nur erfasst, wenn Sie das Dienstprogramm RUNSTATS für den Index mit der Klausel DETAILED ausführen.

Wenn keine Statistiken zur Clusterbildung verfügbar sind, verwendet das Optimierungsprogramm Standardwerte, die von einem geringen Grad an Clusterbildung der Daten bezüglich des Index ausgehen.

Der Grad, zu dem die Daten in Bezug auf einen Index in Clustern zusammengefasst sind, kann bedeutende Auswirkungen auf die Leistung haben, sodass einer der Indizes für eine Tabelle auf einem Grad nahe an 100% Clusterbildung gehalten werden sollte.

Im Allgemeinen kann nur ein Index eine 100-prozentige Clusterbildung aufweisen. Eine Ausnahme bilden nur solche Fälle, in denen die Schlüssel eines anderen Index eine Obermenge der Schlüssel des Clusterindex sind oder in denen es eine De-Facto-Korrelation zwischen den Schlüsselspalten der beiden Indizes gibt.

Bei der Reorganisation einer Tabelle können Sie einen Index angeben, über den die Zeilen in Clustern angeordnet werden und versucht wird, diese Clusterbildung bei der Verarbeitung von Einfügungen beizubehalten. Da Aktualisierungen und Einfügungen die Clusterbildung in Bezug auf den Index verringern können, müssen Sie die Tabelle eventuell in regelmäßigen Abständen reorganisieren. Um die Häufigkeit der Reorganisation einer Tabelle zu verringern, die aufgrund von INSERT-, UPDATE- und DELETE-Anweisungen häufig geändert wird, können Sie beim Ändern einer Tabelle (ALTER TABLE) den Parameter PCTFREE verwenden.

Dieser Parameter ermöglicht es, weitere Einfügungen so in die vorhandenen Daten einzugliedern, dass die Clusterbildung erhalten bleibt.

Joins

Ein *Join* ist der Prozess der Kombination von Informationen aus mindestens zwei Tabellen auf der Grundlage eines gemeinsamen Geltungsbereichs der Informationen. Zeilen aus der einen Tabelle werden mit Zeilen aus einer anderen Tabelle paarweise verbunden, wenn die Informationen in den entsprechenden Zeilen die Joinbedingung erfüllen.

Betrachten Sie zum Beispiel die beiden folgenden Tabellen:

Tabelle1

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

Tabelle2

PROJ_ID	NAME
1	Sam
3	Joe
4	Mary
1	Sue
2	Mike

Für den Join von Tabelle1 und Tabelle2 an den Stellen, an denen die ID-Spalten die gleichen Werte haben, verwenden Sie die folgende SQL-Anweisung:

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

Diese Anweisung liefert die folgende Menge von Ergebniszeilen:

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

Abhängig davon, ob ein Joinvergleichselement vorhanden ist und welche Verarbeitungsaufwände anhand der Tabellen- und Indexstatistiken ermittelt werden, wählt das Optimierungsprogramm eine der folgenden Joinmethoden aus:

- Join mit Verschachtelungsschleife (Nested-Loop Join)
- Mischjoin (Merge Join)
- Hash-Join

Beim Join zweier Tabellen wird die eine Tabelle als äußere Tabelle und die andere als innere Tabelle ausgewählt. Auf die äußere Tabelle wird zuerst zugegriffen, und sie wird nur einmal durchsucht. Ob die innere Tabelle mehrere Male durchsucht wird, hängt von der Art des Joins und den vorhandenen Indizes ab. Auch wenn in einer Abfrage mehr als zwei Tabellen verknüpft werden, verknüpft das Optimierungsprogramm jeweils nur zwei Tabellen gleichzeitig. Falls erforderlich, werden temporäre Tabellen zum Speichern von Zwischenergebnissen erstellt.

Sie können explizite Joinoperatoren wie INNER oder LEFT OUTER JOIN angeben, um festzulegen, wie Tabellen im Join verwendet werden. Bevor Sie jedoch eine Abfrage auf diese Art ändern, sollten Sie das Optimierungsprogramm ermitteln

lassen, wie die Tabellen zu verknüpfen sind. Anschließend analysieren Sie die Abfrageleistung und entscheiden, ob Joinoperatoren hinzugefügt werden sollten.

Joinmethoden

Das Optimierungsprogramm kann eine von drei Grundstrategien wählen, wenn Abfragen einen Join von Tabellen erfordern.

- Join mit Verschachtelungsschleife (Nested-Loop Join)
- Mischjoin (Merge Join)
- Hash-Join

Diese Methoden werden in den folgenden Abschnitten beschrieben.

Join mit Verschachtelungsschleife

Ein Join mit Verschachtelungsschleife (Nested-Loop Join) wird auf eine der beiden folgenden Arten ausgeführt:

- Durchsuchen der inneren Tabelle für jede Zeile der äußeren Tabelle, auf die zugegriffen wird

  Betrachten Sie zum Beispiel die Spalte A in den Tabellen T1 und T2 mit folgenden Werten:

Äußere Tabelle T1: Spalte A	Innere Tabelle T2: Spalte A
2	3
3	2
3	2
	3
	1

  Zur Durchführung eines Joins mit Verschachtelungsschleife geht der Datenbankmanager in folgenden Schritten vor:

  1. Lesen der ersten Zeile aus T1. Der Wert für A ist „2".
  2. Durchsuchen von T2, bis ein übereinstimmender Wert („2") gefunden wird, und anschließender Join der beiden Zeilen.
  3. Durchsuchen von T2, bis der nächste übereinstimmende Wert („2") gefunden wird, und anschließender Join der beiden Zeilen.
  4. Durchsuchen von T2 bis zum Ende der Tabelle.
  5. Zurückkehren zu T1 und Lesen der nächsten Zeile („3").
  6. Durchsuchen von T2 von der ersten Zeile an, bis ein übereinstimmender Wert („3") gefunden wird, und anschließender Join der beiden Zeilen.
  7. Durchsuchen von T2, bis der nächste übereinstimmende Wert („3") gefunden wird, und anschließender Join der beiden Zeilen.
  8. Durchsuchen von T2 bis zum Ende der Tabelle.
  9. Zurückkehren zu T1 und Lesen der nächsten Zeile („3").
  10. Durchsuchen von T2 wie vorher und Join aller übereinstimmenden Zeilen („3").

- Durchführen einer Indexsuche für die innere Tabelle für jede Zeile der äußeren Tabelle, auf die zugegriffen wird

  Diese Methode kann für die angegebenen Vergleichselemente verwendet werden, wenn es ein Vergleichselement der folgenden Form gibt:

  ```
  ausdr(äußere_tabelle.spalte)  relop  innere_tabelle.spalte
  ```

Dabei ist `relop` ein relativer Operator (z. B. =, >, >=, < oder <=) und `ausdr` ist ein gültiger Ausdruck für die äußere Tabelle. Betrachten Sie die folgenden Beispiele:

```
OUTER.C1 + OUTER.C2 <= INNER.C1
OUTER.C4 < INNER.C3
```

Diese Methode könnte die Anzahl der Zeilen, auf die in der inneren Tabelle für jeden Zugriff auf die äußere Tabelle zugegriffen wird, wesentlich verringern, obwohl dies von einer Reihe von Faktoren abhängig ist, zu denen auch die Selektivität des Joinvergleichselements zählt.

Bei der Beurteilung eines Joins mit Verschachtelungsschleife entscheidet das Optimierungsprogramm auch, ob die äußere Tabelle sortiert wird oder nicht, bevor der Join durchgeführt wird. Durch Sortieren der äußeren Tabelle nach den Werten der Joinspalten kann die Anzahl der Leseoperationen zum Zugriff auf Seiten auf der Platte für die innere Tabelle verringert werden, da es wahrscheinlicher wird, dass sich die Seiten bereits im Pufferpool befinden. Wenn der Join einen Index mit einem hohen Grad der Clusterbildung verwendet, um auf die innere Tabelle zuzugreifen, und wenn die äußere Tabelle sortiert wurde, kann die Anzahl von Indexseiten, auf die zugegriffen wird, minimiert werden.

Wenn darüber hinaus das Optimierungsprogramm erwartet, dass durch den Join eine spätere Sortierung aufwendiger wird, kann es entscheiden, die Sortierung vor dem Join durchzuführen. Eine spätere Sortierung könnte erforderlich sein, um die Klauseln GROUP BY, DISTINCT, ORDER BY oder einen Mischjoin zu unterstützen.

Mischjoin

Ein Mischjoin (engl. *Merge Join*, *Merge Scan Join* oder *Sort Merge Join*) erfordert ein Vergleichselement der Form tabelle1.spalte = tabelle2.spalte. Ein solches Vergleichselement wird als *Equijoin-Vergleichselement* bezeichnet. Ein Mischjoin erfordert entweder über einen Indexzugriff oder durch eine Sortierung geordnete Eingaben für die Joinspalten. Ein Mischjoin kann nicht verwendet werden, wenn die Joinspalte eine LONG-Feldspalte oder eine LOB-Spalte ist.

Bei einem Mischjoin werden die Tabellen, die verknüpft werden, gleichzeitig durchsucht. Die äußere Tabelle des Mischjoins wird nur einmal durchsucht. Die innere Tabelle wird auch nur einmal durchsucht, sofern in der äußeren Tabelle keine sich wiederholenden Werte auftreten. Wenn wiederholte Werte auftreten, wird eine Gruppe von Zeilen der inneren Tabelle eventuell noch einmal durchsucht. Betrachten Sie zum Beispiel die Spalte A in den Tabellen T1 und T2 mit folgenden Werten:

Äußere Tabelle T1: Spalte A	Innere Tabelle T2: Spalte A
2	1
3	2
3	2
	3
	3

Zur Durchführung eines Mischjoins geht der Datenbankmanager in folgenden Schritten vor:

1. Lesen der ersten Zeile aus T1. Der Wert für A ist „2".
2. Durchsuchen von T2, bis ein übereinstimmender Wert gefunden wird, und anschließender Join der beiden Zeilen.

3. Fortsetzen des Durchsuchens von T2, solange die Spalten übereinstimmen, und dabei Join der Zeilen.
4. Wenn der Wert „3" in T2 gelesen wird, Zurückkehren zu T1 und Lesen der nächsten Zeile.
5. Der nächste Wert in T1 ist „3", der mit T2 übereinstimmt, also Join der Zeilen.
6. Fortsetzen des Durchsuchens von T2, solange die Spalten übereinstimmen, und dabei Join der Zeilen.
7. Erreichen des Endes von T2.
8. Zurückkehren zu T1, um die nächste Zeile zu lesen. Beachten Sie, dass der nächste Wert in T1 derselbe ist wie der vorige Wert aus T1, sodass T2 noch einmal, angefangen vom ersten Wert „3" in T2, durchsucht wird. Der Datenbankmanager speichert diese Position.

Hash-Join

Eine Hash-Join erfordert ein oder mehrere Vergleichselemente der Form tabelle1.spalteX = tabelle2.spalteY, wobei die Spaltentypen übereinstimmen müssen. Spalten des Typs CHAR müssen die gleiche Länge aufweisen. Bei Spalten des Typs DECIMAL muss die Genauigkeit und die Anzahl der Kommastellen übereinstimmen. Spalten des Typs DECFLOAT müssen die gleiche Genauigkeit (Stellenanzahl) aufweisen. Der Spaltentyp kann keine LONG-Feldspalte oder LOB-Spalte sein.

Zunächst wird die als INNER designierte Tabelle (innere Tabelle) durchsucht, wobei die Zeilen in Speicherpuffer kopiert werden, die dem Sortierzwischenspeicher entnommen werden, der wiederum durch den Datenbankkonfigurationsparameter *sortheap* definiert ist. Die Speicherpuffer werden auf der Grundlage eines Hashwerts, der auf den Spalten der Joinvergleichselemente berechnet wird, in Abschnitte unterteilt. Wenn die Größe der inneren Tabelle (INNER) die Größe des Sortierspeichers überschreitet, werden Puffer aus ausgewählten Abschnitten in temporäre Tabellen geschrieben.

Wenn die innere Tabelle verarbeitet wurde, wird die zweite, als OUTER designierte Tabelle (äußere Tabelle) durchsucht und ihre Zeilen werden mit den Zeilen aus der inneren Tabelle (INNER) abgeglichen, indem zuerst der Hashwert, der für die Spalten der Joinvergleichselemente errechnet wurde, verglichen wird. Wenn der Hashwert für die OUTER-Zeilenspalte mit dem Hashwert der INNER-Zeilenspalte übereinstimmt, werden die tatsächlichen Werte der Joinelementspalten verglichen.

Zeilen der äußeren Tabelle (OUTER), die Teilen der Tabelle entsprechen, die nicht in eine temporäre Tabelle geschrieben wurden, werden sofort mit den Zeilen der inneren Tabelle im Speicher abgeglichen. Wenn der entsprechende Teil der inneren Tabelle in eine temporäre Tabelle geschrieben wurde, wird die OUTER-Zeile ebenfalls in eine temporäre Tabelle geschrieben. Schließlich werden abgeglichene Paare von Tabellenteilen aus den temporären Tabellen gelesen, die Hashwerte ihrer Zeilen abgeglichen und die Joinvergleichselemente geprüft.

Zur Erzielung der vollen Leistungsvorteile des Hash-Joins müssen Sie möglicherweise den Wert des Konfigurationsparameters *sortheap* der Datenbank und des Konfigurationsparameters *sheapthres* des Datenbankmanagers ändern.

Die Leistung des 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 *sheapthres* verfügbar ist, und optimieren dann den Parameter **sortheap**. Erhöhen Sie den Wert dieses

Parameters, bis Sie möglichst viele Hashschleifen und Überläufe auf den Plattenspeicher vermeiden, jedoch nicht den durch den Parameter *sheapthres* definierten Grenzwert erreichen.

Die Erhöhung des Werts für *sortheap* sollte außerdem die Leistung von Abfragen verbessern, die mehrere Sortierungen erfordern.

Strategien zur Auswahl optimaler Joins

Das Optimierungsprogramm nutzt verschiedene Methoden zur Auswahl einer optimalen Joinstrategie für eine Abfrage. Zu diesen Methoden zählen die folgenden Suchstrategien, die durch die Optimierungsklasse der Abfrage bestimmt werden:

- Schnelle Joinaufzählung (Greedy Join Enumeration)
  - Effizient im Hinblick auf Speicherbedarf und Zeit.
  - Aufzählung in einer Richtung, d. h., wenn eine Joinmethode für zwei Tabellen ausgewählt ist, wird sie im Laufe der weiteren Optimierung nicht mehr geändert.
  - Eventuell wird nicht der optimale Zugriffsplan gewählt, wenn viele Tabellen verknüpft werden. Wenn eine Abfrage nur zwei oder drei Tabellen verknüpft, ist der Zugriffsplan, der durch die schnelle Joinaufzählung ausgewählt wird, derselbe wie der Zugriffsplan, der durch die dynamisch programmierte Joinaufzählung (Dynamic Programming Join Enumeration) ausgewählt wird. Dies gilt insbesondere dann, wenn die Abfrage viele entweder explizit angegebene oder implizit durch die Anwendung der Transitivität generierte Joinvergleichselemente für dieselbe Spalte hat.
- Dynamisch programmierte Joinaufzählung (Dynamic Programming Join Enumeration)
  - Der Bedarf an Speicherplatz und Zeit wächst exponenziell mit zunehmender Anzahl der verknüpften Tabellen.
  - Effiziente und erschöpfende Suche nach dem besten Zugriffsplan.
  - Ähnlich der Strategie, die von DB2 für OS/390 oder z/OS verwendet wird.

Der Algorithmus für die Joinaufzählung spielt die entscheidende Rolle bei der Bestimmung der Anzahl von Zugriffsplankombinationen, die das Optimierungsprogramm prüft.

Sternschemajoins

Die Tabellen, auf die in einer Abfrage verwiesen wird, sind fast immer durch Joinvergleichselemente miteinander verbunden. Wenn zwei Tabellen ohne Joinvergleichselement verknüpft werden, wird das kartesische Produkt der beiden Tabellen gebildet. Bei einem kartesischen Produkt wird jede ausgewählte Zeile der ersten Tabelle mit jeder ausgewählten Zeile der zweiten Tabelle verknüpft. Das Ergebnis ist eine Tabelle, die aus dem Kreuzprodukt in der Größe der beiden Tabellen besteht und in der Regel sehr groß ist. Da ein solcher Plan wahrscheinlich keine gute Leistung zulässt, vermeidet das Optimierungsprogramm sogar die Aufwandsabschätzung für einen Plan dieser Art.

Die einzigen Ausnahmen bilden die Fälle, in denen die Optimierungsklasse auf 9 gesetzt wird oder der Sonderfall eines Sternschemas vorliegt. Ein *Sternschema* (engl. star schema) enthält eine zentrale Tabelle, die als Fakttabelle bezeichnet wird, und andere Tabellen, die als Dimensionstabellen bezeichnet werden. Die Dimensionstabellen haben, ungeachtet der Abfrage, jeweils nur einen einzigen Join, über den sie mit der Fakttabelle verbunden sind. Jede Dimensionstabelle enthält zusätzliche

Werte, die die Informationen über eine bestimmte Spalte in der Fakttabelle erweitern. Eine typische Abfrage besteht aus mehreren lokalen Vergleichselementen, die auf Werte in den Dimensionstabellen verweisen, und enthält Joinvergleichselemente, die die Dimensionstabellen mit der Fakttabelle verbinden. Für solche Abfragen kann es vorteilhaft sein, das kartesische Produkt mehrerer kleiner Dimensionstabellen zu berechnen und erst anschließend auf die umfangreiche Fakttabelle zuzugreifen. Diese Technik ist dann von Vorteil, wenn mehrere Joinvergleichselemente einem mehrspaltigen Index entsprechen.

DB2 kann Abfragen erkennen, die für Datenbanken durchgeführt werden, die in Sternschemata aufgebaut sind und über mindestens zwei Dimensionstabellen verfügen, und kann den Suchbereich vergrößern, um mögliche Pläne zur Berechnung des kartesischen Produkts von Dimensionstabellen zu berücksichtigen. Wenn der Plan mit den kartesischen Produkten den niedrigsten geschätzten Aufwand verursacht, wird er vom Optimierungsprogramm ausgewählt.

Die oben behandelte Strategie des Sternschemajoins basiert auf der Annahme, dass Primärschlüsselindizes im Join verwendet werden. Eine andere Situation liegt vor, wenn Fremdschlüsselindizes verwendet werden. Wenn die Fremdschlüsselspalten in der Fakttabelle einspaltige Indizes sind und es eine relativ hohe Selektivität über alle Dimensionstabellen hinweg gibt, könnte die folgende Methode des Sternjoins zur Anwendung kommen:

1. Verarbeiten jeder Dimensionstabelle wie folgt:
   - Durchführen eines Semi-Joins zwischen der Dimensionstabelle und dem Fremdschlüsselindex für die Fakttabelle
   - Dynamisches Erstellen einer Bitzuordnung durch ein Hashverfahren für die Werte der Zeilen-IDs (RID)
2. Anwenden von AND-Vergleichselementen auf die vorige Bitzuordnung für jede Bitzuordnung
3. Feststellen der verbliebenen RIDs, nachdem die letzte Bitzuordnung verarbeitet wurde
4. Optionales Sortieren dieser RIDs
5. Abrufen einer Zeile aus der Basistabelle
6. Erneuter Join der Fakttabelle mit jeder der zugehörigen Dimensionstabellen, dabei Zugreifen auf die Spalten in Dimensionstabellen, die für die SELECT-Klausel benötigt werden
7. Erneutes Anwenden der Restvergleichselemente

Diese Methode erfordert keine mehrspaltigen Indizes. Explizite referenzielle Integritätsbedingungen zwischen Fakttabelle und Dimensionstabellen sind nicht erforderlich, obwohl die Beziehung zwischen Fakttabelle und Dimensionstabellen doch auf diese Weise realisiert werden sollte.

Die Bitzuordnungen, die von Sternjointechniken dynamisch erstellt und verwendet werden, erfordern Sortierspeicher, dessen Größe durch den Datenbankkonfigurationsparameter *sortheap* definiert wird.

Joins mit frühem Verlassen

Das Optimierungsprogramm kann einen Join mit frühem Verlassen (Early Out Join) auswählen, wenn es erkennt, dass jede Zeile aus der einen Tabelle maximal mit einer Zeile aus der anderen Tabelle verknüpft werden muss.

Ein Join mit frühem Verlassen ist bei einem Joinvergleichselement mit der Schlüsselspalte (bzw. den Schlüsselspalten) einer der Tabellen möglich. Betrachten Sie zum Beispiel die folgende Abfrage, die die Namen von Mitarbeitern (Employee) und ihren direkten Vorgesetzten (Manager) zurückgibt.

```
SELECT EMPLOYEE.NAME AS EMPLOYEE_NAME, MANAGER.NAME AS MANAGER_NAME
FROM EMPLOYEE AS EMPLOYEE, EMPLOYEE AS MANAGER
WHERE EMPLOYEE.MANAGER_ID=MANAGER.ID
```

Unter der Voraussetzung, dass die Spalte ID eine Schlüsselspalte in der Tabelle EMPLOYEE ist und das jeder Mitarbeiter höchstens einen Manager hat, kann der oben angegebene Join die Suche nach einer weiteren übereinstimmenden Zeile in der Tabelle MANAGER vermeiden.

Ein Join mit frühem Verlassen ist auch möglich, wenn die Abfrage eine Klausel DISTINCT enthält. Betrachten Sie zum Beispiel die folgende Abfrage, die die Namen von Automobilherstellern (MAKE) zurückgibt, die ein Modell anbieten, das für über 30.000 $ verkauft wird.

```
SELECT DISTINCT MAKE.NAME
FROM MAKE, MODEL
WHERE MAKE.MAKE_ID=MODEL.MAKE_ID AND MODEL.PRICE>30000
```

Für jeden Hersteller (MAKE) muss lediglich festgestellt werden, ob eines der von ihm hergestellten Modelle für über 30.000 $ verkauft wird. Eine Verknüpfung eines Herstellers mit allen von ihm hergestellten Modellen, die für über 30.000 $ verkauft werden, ist für die Richtigkeit des Abfrageergebnisses nicht erforderlich.

Ein Join mit frühem Verlassen ist auch möglich, wenn der Join Daten für eine Klausel GROUP BY mit einer Spaltenfunktion MIN oder MAX liefert. Betrachten Sie zum Beispiel die folgende Abfrage, die Börsenkürzel (STOCK SYMBOL) mit dem jüngsten Datum vor dem Jahr 2000 zurückgibt, für die der Schlusspreis (CLOSE) einer bestimmten Aktie mindestens 10 % höher als der Eröffnungspreis (OPEN) ist:

```
SELECT DAILYSTOCKDATA.SYMBOL, MAX(DAILYSTOCKDATA.DATE) AS DATE
FROM SP500, DAILYSTOCKDATA
WHERE SP500.SYMBOL=DAILYSTOCKDATA.SYMBOL AND DAILYSTOCKDATA.DATE<'01/01/2000' AND
      DAILYSTOCKDATA.CLOSE / DAILYSTOCKDATA.OPEN >= 1.1
GROUP BY DAILYSTOCKDATA.SYMBOL
```

Die 'qualifizierte Menge' sei definiert als die Menge von Zeilen aus der Tabelle DAILYSTOCKDATA, die die Datums- und Preisvorgaben erfüllt und mit einem bestimmten Börsenkürzel aus der Tabelle SP500 durch einen Join verknüpft wird. Wenn die qualifizierte Menge aus der Tabelle DAILYSTOCKDATA für jede Börsenkürzelzeile aus der Tabelle SP500 über das Datum (DATE) absteigend sortiert ist, muss nur die erste Zeile aus der qualifizierten Menge für jedes Kürzel zurückgegeben werden, da diese erste Zeile das jüngste Datum für das jeweilige Börsenkürzel enthält. Die übrigen Zeilen in der qualifizierten Menge sind für die Richtigkeit des Abfrageergebnisses nicht erforderlich.

Zusammengesetzte Tabellen

Wenn das Ergebnis des Joins zweier Tabellen eine neue Tabelle ist, die in diesem Fall als *zusammengesetzte* Tabelle bezeichnet wird, wird diese Tabelle in der Regel zur äußeren Tabelle eines weiteren Joins mit einer anderen inneren Tabelle. Ein solcher Join wird als „Composite-Outer-Join" (zusammengesetzter Outer Join) bezeichnet. In einigen Fällen, besonders bei Verwendung der schnellen Joinaufzählung (Greedy Join Enumeration), ist es sinnvoll, das Ergebnis des Joins zweier Tabellen zur inneren Tabelle eines späteren Joins zu machen. Wenn die innere

Tabelle eines Joins aus dem Ergebnis des Joins zweier oder mehrerer Tabellen besteht, wird ein solcher Plan als „Composite-Inner-Join" (zusammengesetzter Inner Join) bezeichnet. Betrachten Sie zum Beispiel die folgende Abfrage:

```
SELECT COUNT(*)
FROM T1, T2, T3, T4
WHERE T1.A = T2.A AND
      T3.A = T4.A AND
      T2.Z = T3.Z
```

Hier könnte es von Vorteil sein, die Tabellen T1 und T2 zu verknüpfen (T1xT2), anschließend die Tabellen T3 und T4 zu verknüpfen (T3xT4) und schließlich das Ergebnis des ersten Joins als äußere Tabelle und das Ergebnis des zweiten Joins als innere Tabelle auszuwählen. Im daraus resultierenden Plan ((T1xT2) x (T3xT4)) ist das Ergebnis des Joins (T3xT4) eine zusammengesetzte innere Tabelle. Abhängig von der Abfrageoptimierungsklasse belegt das Optimierungsprogramm die maximale Anzahl von Tabellen, die als innere Tabelle eines Joins verwendet werden können, mit unterschiedlichen Einschränkungen. Composite-Inner-Joins sind bei den Optimierungsklassen 5, 7 und 9 zulässig.

Replizierte MQTs in Umgebungen mit partitionierten Datenbanken

Replizierte MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle) verbessern die Leistung häufig ausgeführter Joins in einer partitionierten Datenbankumgebung, indem sie der Datenbank die Möglichkeit geben, vorberechnete Werte der Tabellendaten zu pflegen.

Betrachten Sie ein Beispiel einer Abfrage und einer replizierten MQT. Dazu gelten folgende Annahmen:

- Die Tabelle SALES (Verkauf) befindet sich im Mehrpartitionstabellenbereich REGIONTABLESPACE und wird über die Spalte REGION unterteilt.
- Die Tabellen EMPLOYEE (Mitarbeiter) und DEPARTMENT (Abteilung) sind in einer Datenbankpartitionsgruppe mit Einzelpartition.

Erstellen Sie eine replizierte MQT auf der Basis der Informationen in der Tabelle EMPLOYEE.

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM   EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

Führen Sie zur Aktualisierung der replizierten MQT die folgende Anweisung aus:

```
REFRESH TABLE R_EMPLOYEE;
```

Anmerkung: Nach der Verwendung der Anweisung REFRESH sollten Sie RUNSTATS für die replizierte Tabelle in gleicher Weise wie für die anderen Tabellen ausführen.

Im folgenden Beispiel werden der Verkauf nach Mitarbeiter, die Summe für die Abteilung und die Gesamtsumme berechnet:

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

Anstatt der Tabelle EMPLOYEE, die nur in einer Datenbankpartition ist, verwendet der Datenbankmanager die Tabelle R_EMPLOYEE, die in jede der Datenbankpartitionen repliziert wird, in der die Tabelle SALES gespeichert ist. Der Leistungsvorteil ergibt sich daraus, dass die Mitarbeiterinformationen zur Berechnung des Joins nicht zu jeder Datenbankpartition über das Netzwerk gesendet werden müssen.

Replizierte MQTs in zusammengefassten Joins

Replizierte MQTs können auch bei der Kollokation von Joins helfen. Wenn beispielsweise ein Sternschema eine große Fakttabelle enthält, die sich über zwanzig Knoten erstreckt, sind die Joins zwischen der Fakttabelle und den Dimensionstabellen am effizientesten, wenn diese Tabellen durch Kollokation zusammenfasst werden. Wenn sich alle Tabellen in derselben Datenbankpartitionsgruppe befinden, ist höchstens eine Dimensionstabelle korrekt für einen zusammengefassten Join (d. h. Join von durch Kollokation zusammengefassten Tabellen) partitioniert. Die anderen Dimensionstabellen können nicht in einem zusammengefassten Join verwendet werden, weil die Joinspalten der Fakttabelle nicht dem Verteilungsschlüssel der Fakttabelle entsprechen.

Betrachten Sie zum Beispiel eine Tabelle mit dem Namen FACT (C1, C2, C3, ...), die über Spalte C1 unterteilt ist, und eine Tabelle mit dem Namen DIM1 (C1, dim1a, dim1b, ...), die über C1 unterteilt ist, und eine Tabelle DIM2 (C2, dim2a, dim2b, ...), die über C2 unterteilt ist, usw.

In diesem Fall können Sie erkennen, dass der Join zwischen FACT und DIM1 perfekt ist, da das Vergleichselement DIM1.C1 = FACT.C1 durch Kollokation zusammengefasst vorliegt. Beide Tabellen werden über die Spalte C1 unterteilt.

Der Join mit DIM2 mit dem Vergleichselement WHERE DIM2.C2 = FACT.C2 kann jedoch nicht durch Kollokation zusammengefasst werden, da FACT über die Spalte C1 unterteilt ist und nicht über die Spalte C2. In diesem Fall kann es sinnvoll sein, die Tabelle DIM2 in der Datenbankpartitionsgruppe der Fakttabelle zu replizieren, sodass der Join lokal in jeder Datenbankpartition erfolgen kann.

Anmerkung: Diese Erörterung replizierter MQTs bezieht sich auf datenbankinterne Replikation. Datenbankübergreifende Replikation erfordert Subskriptionen, Steuertabellen und Daten in verschiedenen Datenbanken und auf verschiedenen Betriebssystemen.

Wenn Sie eine replizierte MQT erstellen, kann die Quellentabelle eine Tabelle auf einem einzelnen Knoten oder eine Tabelle auf mehreren Knoten in einer Datenbankpartitionsgruppe sein. In den meisten Fällen ist die replizierte Tabelle klein und kann in einer Datenbankpartitionsgruppe mit einem Knoten untergebracht werden. Sie können die zu replizierende Datenmenge einschränken, indem Sie nur einen Teil der Spalten der Tabelle angeben, die Zeilenzahl mithilfe der verwendeten Vergleichselemente begrenzen oder beide Methoden anwenden. Für das Funktionieren replizierter MQTs ist die Option zur Datenerfassung nicht erforderlich.

Eine replizierte MQT kann auch in einer Datenbankpartitionsgruppe mit mehreren Knoten erstellt werden, sodass Kopien der Quellentabelle in allen Datenbankpartitionen erstellt werden. Joins zwischen einer großen Fakttabelle und den Dimensionstabellen finden in dieser Art von Umgebung mit größerer Wahrscheinlichkeit lokal statt, als wenn die Quellentabelle an alle Datenbankpartitionen per Broadcast übertragen werden muss.

Indizes für replizierte Tabellen werden nicht automatisch erstellt. Sie können Indizes erstellen, die sich von denen für die Quellentabelle unterscheiden. Zur Vermeidung von Verletzungen von Integritätsbedingungen, die für die Quellentabellen nicht vorhanden sind, können Sie jedoch keine eindeutigen Indizes erstellen oder Integritätsbedingungen für die replizierten Tabellen definieren. Integritätsbedingungen sind auch dann nicht zulässig, wenn die gleichen Integritätsbedingungen für die Quellentabelle gelten.

Auf replizierte Tabellen kann in einer Abfrage direkt verwiesen werden, jedoch können Sie nicht das Vergleichselement NODENUMBER() mit einer replizierten Tabelle verwenden, um die Tabellendaten in einer bestimmten Partition abzurufen.

Verwenden Sie die EXPLAIN-Einrichtung, um festzustellen, ob eine replizierte MQT von dem Zugriffsplan für eine Abfrage genutzt wurde. Ob der vom Optimierungsprogramm ausgewählte Zugriffsplan die replizierte MQT verwendet, hängt von den Informationen ab, die verknüpft werden müssen. Die replizierte MQT könnte zum Beispiel dann nicht verwendet werden, wenn das Optimierungsprogramm feststellt, dass es weniger aufwendig wäre, die ursprüngliche Quellentabelle an die anderen Datenbankpartitionen der Datenbankpartitionsgruppe per Broadcast rundzusenden.

Joinstrategien in partitionierten Datenbanken

In mancher Hinsicht unterscheiden sich die Joinstrategien in einer Umgebung mit partitionierten Datenbanken von denen in einer Umgebung mit nicht partitionierten Datenbanken. Zur Verbesserung der Leistung können zusätzliche Techniken auf die Standardjoinmethoden angewandt werden.

Ein Aspekt bei Tabellen, die häufig an Joins in einer Umgebung mit partitionierten Datenbanken beteiligt sind, ist die Tabellenkollokation. Die Tabellenkollokation stellt eine Methode in einer Umgebung mit partitionierten Datenbanken dar, mit der Daten aus einer Tabelle mithilfe eines gleichen Verteilungsschlüssels zusammen mit den Daten aus einer anderen Tabelle in derselben Datenbankpartition angeordnet werden. Wenn die Daten einmal in dieser Weise zusammengefasst sind, können zu verknüpfende Daten an einer Abfrage beteiligt sein, ohne im Rahmen der Aktivitäten für die Abfrage von einer Datenbankpartition in eine andere übertragen werden zu müssen. Nur die Ergebnismenge eines Joins wird an den Koordinatorknoten übergeben.

Tabellenwarteschlangen

In den Beschreibungen von Jointechniken in einer Umgebung mit partitionierten Datenbanken wird die folgende Terminologie verwendet:

- Tabellenwarteschlange (wird manchmal auch als *TQ* bezeichnet)

  Ein Mechanismus zur Übertragung von Zeilen zwischen Datenbankpartitionen oder auch zwischen Prozessoren in einer Datenbank mit einer Einzelpartition.
- Gezielt übertragene Tabellenwarteschlange (wird manchmal auch als *DTQ* bezeichnet)

Eine Tabellenwarteschlange, in der Zeilen durch ein Hashverfahren gezielt an eine der empfangenden Datenbankpartitionen geleitet werden.

- Broadcast-Tabellenwarteschlange (wird manchmal auch als *BTQ* bezeichnet)

  Eine Tabellenwarteschlange, in der Zeilen ohne Hashverfahren an alle empfangenden Datenbankpartitionen gesendet werden.

Eine Tabellenwarteschlange wird zu folgenden Zwecken verwendet:

- Übertragen von Tabellendaten von einer Datenbankpartition zu einer anderen bei Verwendung partitionsübergreifender Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung partitionsinterner Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung einer einzigen Datenbankpartition

Jede Tabellenwarteschlange überträgt die Daten in eine einzige Richtung. Der Compiler entscheidet, wo Tabellenwarteschlangen erforderlich sind, und nimmt sie in den Plan auf. Bei der Ausführung des Plans werden die Tabellenwarteschlangen durch die Verbindungen zwischen den Datenbankpartitionen initiiert. Die Tabellenwarteschlangen werden am Ende der Prozesse wieder geschlossen.

Es gibt verschiedene Arten von Tabellenwarteschlangen:

- *Asynchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als asynchron bezeichnet, weil sie Zeilen vor einer FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, vorablesen. Wenn eine FETCH-Anweisung abgesetzt wird, wird die Zeile aus der Tabellenwarteschlange abgerufen.

  Asynchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung verwenden. Wenn Sie Zeilen nur mit FETCH abrufen, ist eine asynchrone Tabellenwarteschlange schneller.
- *Synchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als synchron bezeichnet, weil sie für jede FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, eine Zeile lesen. In jeder Datenbankpartition wird der Cursor in die nächste Zeile gesetzt, die aus der jeweiligen Datenbankpartition zu lesen ist.

  Synchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung nicht angeben. In einer Umgebung mit partitionierten Datenbanken verwendet der Datenbankmanager synchrone Tabellenwarteschlangen, wenn Zeilen aktualisiert werden.
- *Tabellenwarteschlangen für Mischjoin.*

  Bei dieser Art von Tabellenwarteschlange wird die Reihenfolge gewahrt.
- *Reguläre Tabellenwarteschlangen.*

  Diese Tabellenwarteschlangen sind nicht für Mischjoins geeignet. Bei ihnen bleibt die Reihenfolge nicht erhalten.
- *Empfangende Tabellenwarteschlangen.*

  Diese Tabellenwarteschlangen werden mit korrelierten Unterabfragen verwendet. Die Korrelationswerte werden an die Unterabfrage übergeben, und die Ergebnisse mithilfe dieser Art von Tabellenwarteschlange an den übergeordneten Abfrageblock zurückgeliefert.

Joinmethoden in Umgebungen mit partitionierten Datenbanken

Die folgenden Abbildungen veranschaulichen Joinmethoden in einer Umgebung mit partitionierten Datenbanken.

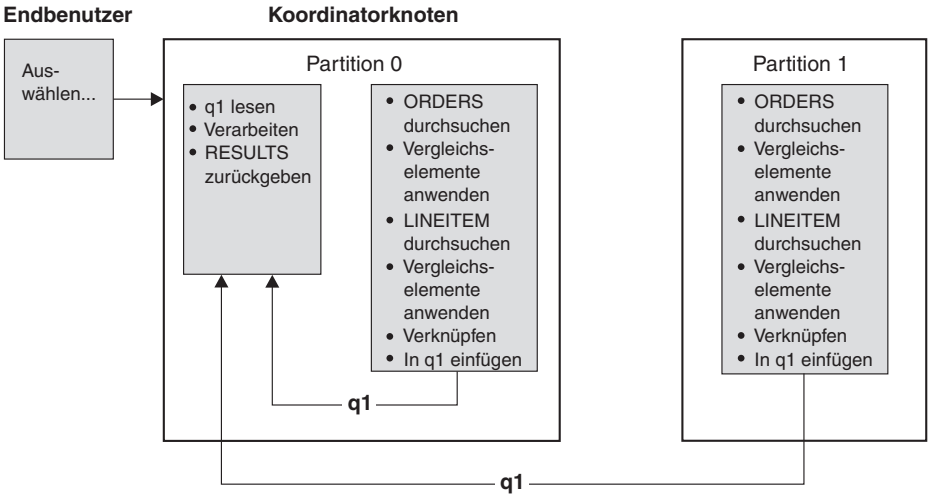
Anmerkung: In den Abbildungen beziehen sich q1, q2 und q3 (für „Queue") auf die Tabellenwarteschlangen in den Beispielen. Die Tabellen, auf die zugegriffen wird, sind für den Zweck dieser Beispielszenarios auf zwei Datenbankpartitionen verteilt. Die Pfeile zeigen die Richtung an, in der die Tabellenwarteschlangen übertragen bzw. gesendet werden. Der Koordinatorknoten entspricht der Datenbankpartition 0.

Zusammengefasste Joins

Ein zusammengefasster Join findet lokal in der Datenbankpartition statt, in der sich die Daten befinden. Die Datenbankpartition sendet die Daten an die anderen Datenbankpartitionen, wenn der Join abgeschlossen ist. Damit das Optimierungsprogramm einen zusammengefassten Join in Erwägung ziehen kann, müssen die verknüpften Tabellen durch Kollokation zusammengefasst sein, und alle Paare mit dem entsprechenden Verteilungsschlüssel müssen Teil der Gleichheitsvergleichselemente sein.

Die folgende Abbildung zeigt ein Beispiel.

Anmerkung: Replizierte MQTs erhöhen die Wahrscheinlichkeit zusammengefasster Joins.

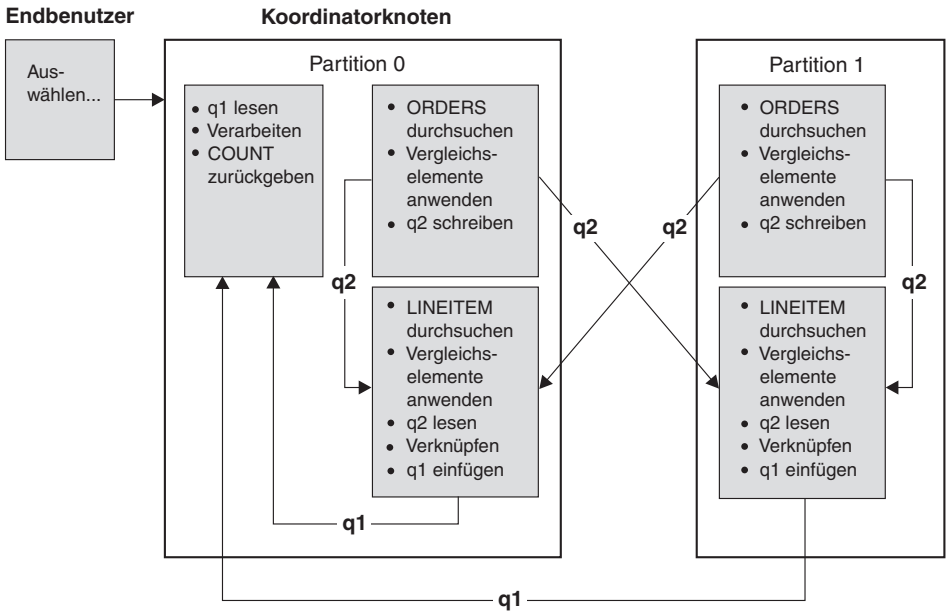


Die beiden Tabellen LINEITEM und ORDERS werden über die Spalte ORDERKEY partitioniert. Der Join erfolgt lokal in jeder Datenbankpartition. In diesem Beispiel wird folgendes Vergleichselement für den Join angenommen: ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 23. Beispiel für einen zusammengefassten Join

Broadcast-Outer-Table-Joins

Broadcast-Outer-Table-Joins (Joins mit rundgesendeter äußerer Tabelle) sind eine parallele Joinstrategie, die angewandt werden kann, wenn es zwischen den zu verknüpfenden Tabellen keine Equijoin-Vergleichselemente gibt. Sie kann außerdem in solchen Fällen verwendet werden, in denen sie die Methode mit dem geringsten Aufwand darstellt. Zum Beispiel könnte ein Broadcast-Outer-Table-Join stattfinden, wenn eine sehr umfangreiche und eine sehr kleine Tabelle am Join beteiligt sind, von denen keine auf die Spalten, auf die die Joinvergleichselemente angewandt werden, verteilt ist. Anstatt beide Tabellen zu teilen, kann es günstiger sein, die kleinere Tabelle an alle Partitionen mit der größeren Tabelle per Broadcast rundzusenden. Die folgenden Abbildungen geben ein Beispiel.

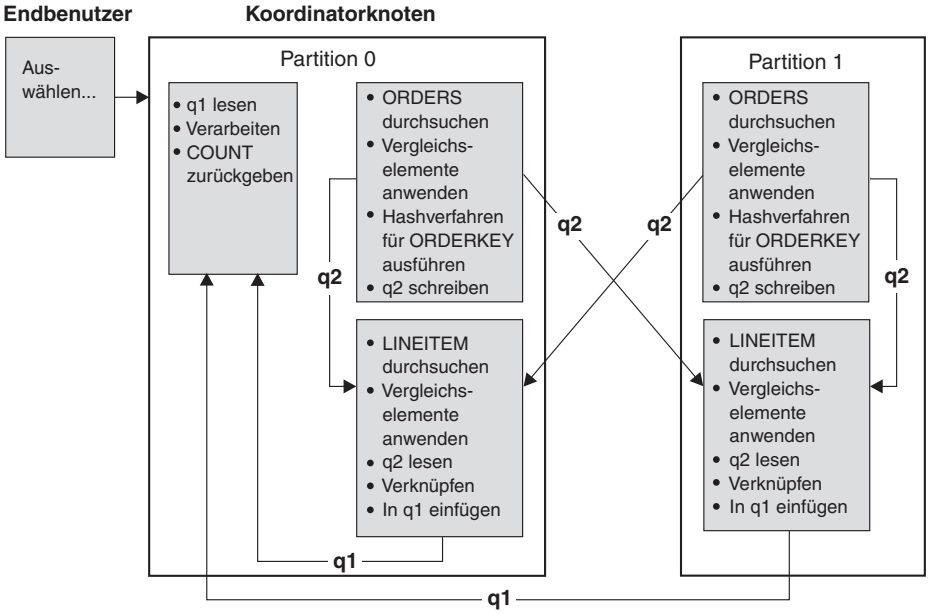


Die Tabelle ORDERS wird an alle Datenbankpartitionen mit der Tabelle LINEITEM gesendet. Tabellenwarteschlange q2 wird im Rundsendebetrieb an alle Datenbankpartitionen der inneren Tabelle gesendet.

Abbildung 24. Beispiel für einen Broadcast-Outer-Table-Join

Directed-Outer-Table-Joins

Bei der Joinstrategie mit gezielt übertragener äußerer Tabelle (Directed-Outer-Table-Join) wird jede Zeile der äußeren Tabelle an einen Teil der inneren Tabelle entsprechend den Teilungsattributen der inneren Tabelle gesendet. Der Join erfolgt in dieser Datenbankpartition. Die folgende Abbildung zeigt ein Beispiel.

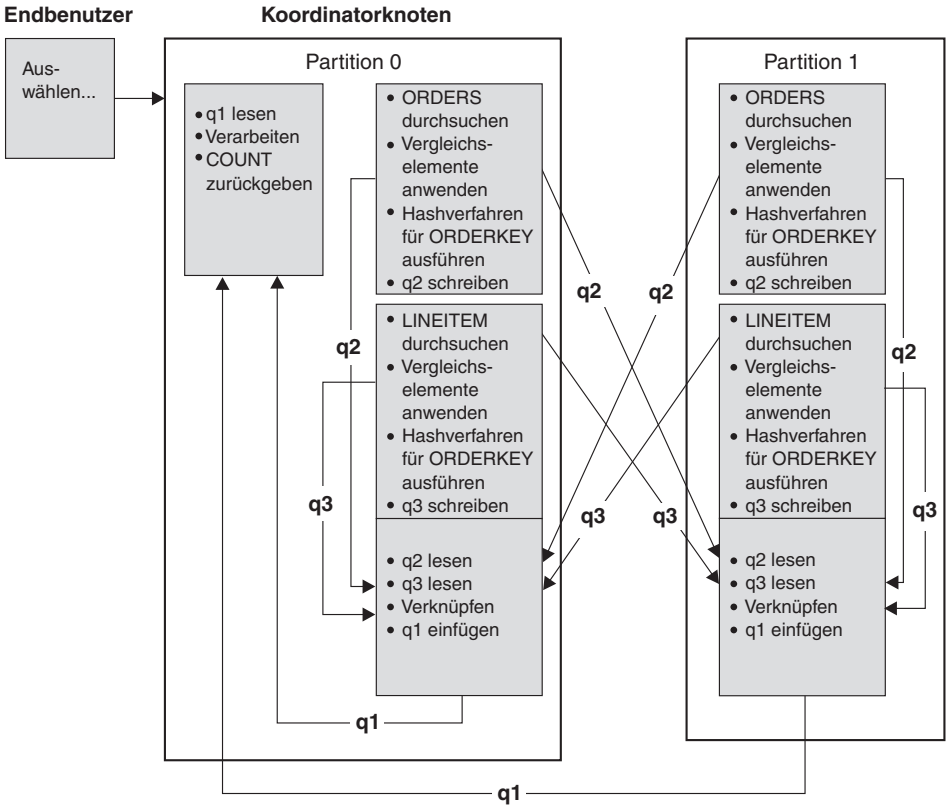


Die Tabelle LINEITEM wird über die Spalte ORDERKEY partitioniert.
Die Tabelle ORDERS wird über eine andere Spalte partitioniert.
Für die Tabelle ORDERS wird das Hashverfahren ausgeführt, und sie
wird an die richtige Datenbankpartition der Tabelle LINEITEM gesendet.
In diesem Beispiel wird folgendes Vergleichselement für den Join angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 25. Beispiel für einen Directed-Outer-Table-Join

Directed-Inner-Table- und Directed-Outer-Table-Joins

Bei der Joinstrategie mit gezielt übertragener innerer und äußerer Tabelle (Directed-Inner-Table- und Directed-Outer-Table-Join) werden Zeilen sowohl der äußeren als auch der inneren Tabelle gezielt an eine Gruppe von Datenbankpartitionen entsprechend den Werten der Joinspalten übertragen. Der Join erfolgt in diesen Datenbankpartitionen. Die folgende Abbildung zeigt ein Beispiel.



Es wird keine Tabelle über die Spalte ORDERKEY partitioniert.
Für beide Tabellen wird das Hashverfahren ausgeführt, und sie
werden an neue Datenbankpartitionen gesendet und dort verknüpft.
Beide Tabellenwarteschlangen q2 und q3 werden übertragen.
In diesem Beispiel wird folgendes Vergleichselement für den Join angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY

Abbildung 26. Beispiel für einen Directed-Inner-Table- und Directed-Outer-Table-Join

Broadcast-Inner-Table-Joins

Bei der Joinstrategie mit rundgesendeter innerer Tabelle (Broadcast-Inner-Table-Join) wird die innere Tabelle per Broadcast an alle Datenbankpartitionen der äußeren Jointabelle gesendet. Die folgende Abbildung zeigt ein Beispiel.

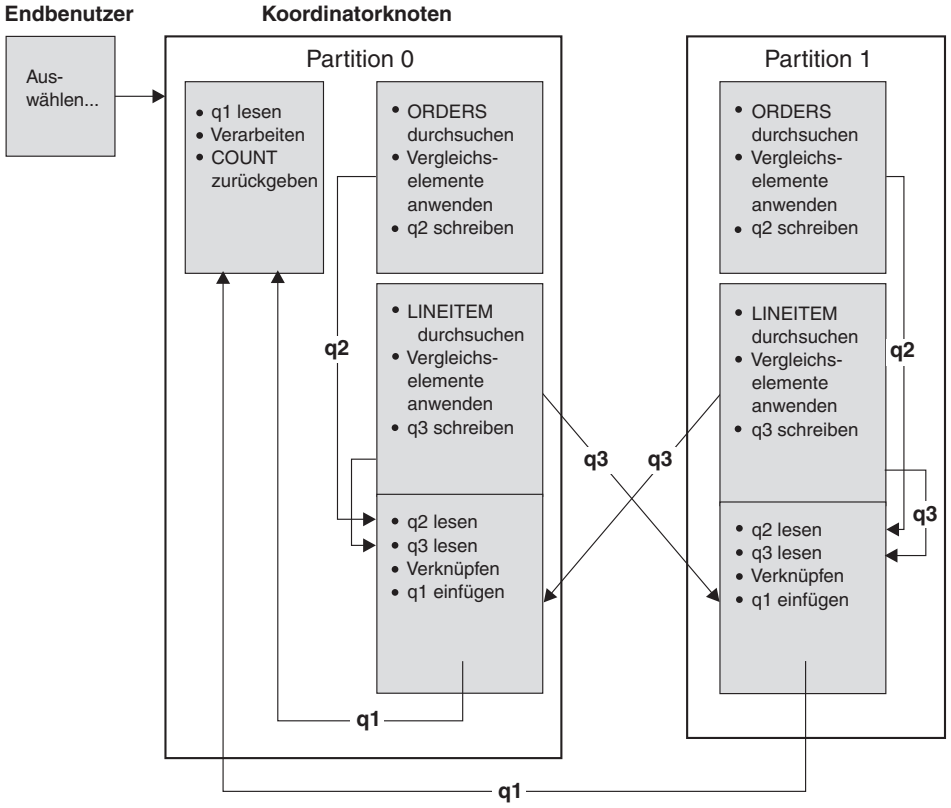
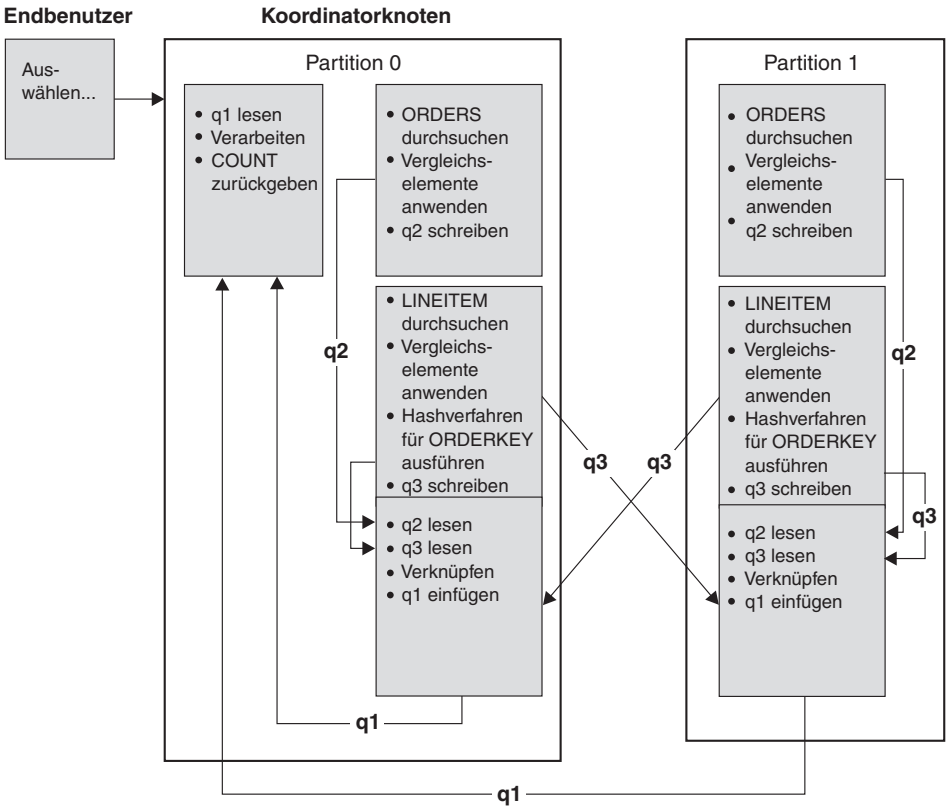


Abbildung 27. Beispiel für einen Broadcast-Inner-Table-Join

Directed-Inner-Table-Joins

Bei der Joinstrategie mit gezielt übertragener innerer Tabelle (Directed-Inner-Table-Join) wird jede Zeile der inneren Tabelle an eine Datenbankpartition der äußeren Tabelle entsprechend den Teilungsattributen der äußeren Tabelle übertragen. Der Join erfolgt in dieser Datenbankpartition. Die folgende Abbildung zeigt ein Beispiel.



Die Tabelle ORDERS wird über die Spalte ORDERKEY partitioniert.
Die Tabelle LINEITEM wird über eine andere Spalte partitioniert.
Für die Tabelle LINEITEM wird das Hashverfahren ausgeführt, und
sie wird an die richtige Datenbankpartition der Tabelle ORDERS gesendet.
In diesem Beispiel wird folgendes Vergleichselement für den Join angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 28. Beispiel für einen Directed-Inner-Table-Join

Auswirkungen des Sortierens und Gruppierens

Wenn das Optimierungsprogramm einen Zugriffsplan auswählt, kalkuliert es die Auswirkungen einer Sortierung von Daten auf die Leistung mit ein. Sortieroperationen werden durchgeführt, wenn kein Index die angeforderte Reihenfolge der abgerufenen Daten herstellen kann. Eine Sortierung kann auch erfolgen, wenn das Optimierungsprogramm feststellt, dass eine Sortierung weniger aufwendig als eine Indexsuche ist. Das Optimierungsprogramm sortiert Daten auf eine der folgenden Arten:

- Weitergeben der Sortierergebnisse über eine Pipe (Piping), wenn die Abfrage ausgeführt wird
- Interne Behandlung der Sortierung innerhalb des Datenbankmanagers

Vergleich von Sortierungen mit und ohne Piping

Wenn die endgültige, sortierte Liste von Daten in einem einzigen sequenziellen Vorgang gelesen werden kann, können die Ergebnisse über eine *Pipe* geleitet werden (Piping). Mit der Piping-Methode lassen sich die Ergebnisse der Sortierung

schneller übertragen als mit Methoden ohne Piping. Das Optimierungsprogramm wählt, wenn möglich, die Pipe zur Übergabe der Sortierergebnisse.

Ungeachtet dessen, ob eine Sortierung mit oder ohne Piping erfolgt, hängt die für die Sortierung benötigte Zeit von einer Reihe von Faktoren ab, wie zum Beispiel der Anzahl der zu sortierenden Zeilen, der Größe des Sortierschlüssels und der Zeilenlänge. Wenn die zu sortierenden Zeilen mehr als den im Zwischenspeicher für Sortierlisten 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 Tabelle im Pufferpool gespeichert. Falls im Pufferpool nicht genügend Platz vorhanden ist, können Seiten aus dieser temporären Tabelle auf die Platte geschrieben werden. Wenn alle Arbeitsgänge des Sortiervorgangs abgeschlossen sind, müssen die sortierten Untermengen zu einer einzigen sortierten Menge von Zeilen zusammengefügt werden. Wenn die Sortierung über eine Pipe geleitet wird, werden die Zeilen beim Zusammenfügen direkt an die Services für relationale Daten (Relational Data Services) weitergegeben.

Pushdown von Gruppier- und Sortieroperatoren

In einigen Fällen kann sich das Optimierungsprogramm entscheiden, einen Sortiervorgang oder eine Spaltenberechnung (Aggregation) von der Komponente der Services für relationale Daten an die Komponente der Datenverwaltungsservices zu verschieben („Pushdown"). Eine solche Verschiebung dieser Operationen verbessert die Leistung, da nun die Datenverwaltungsservices Daten direkt an eine Sortier- oder Spaltenberechnungsroutine übergeben können. Ohne diese Verschiebung übergeben die Datenverwaltungsservices diese Daten zunächst an die Services für relationale Daten, die anschließend wiederum mit den Sortier- bzw. Spaltenberechnungsroutinen kommunizieren. Die folgende Abfrage beispielsweise kann von dieser Art der Optimierung profitieren:

```
SELECT WORKDEPT, AVG(SALARY) AS AVG_DEPT_SALARY
  FROM EMPLOYEE
  GROUP BY WORKDEPT
```

Gruppierungsoperationen in Sortierungen

Wenn das Sortieren dazu dient, die erforderliche Reihenfolge für eine Operation GROUP BY herzustellen, kann das Optimierungsprogramm einige oder alle Spaltenberechnungen (Aggregation) für GROUP BY während des Sortierens durchführen. Dies ist vorteilhaft, wenn die Anzahl der Zeilen in jeder Gruppe sehr groß ist. Es wird sogar noch vorteilhafter, wenn die Durchführung eines Teils der Gruppierung während des Sortierens die Notwendigkeit, dass die Sortierung einen Überlauf auf die Festplatte verursacht, verringert oder ausschließt.

Eine Spaltenberechnung in einer Sortierung erfordert unter Umständen alle der drei folgenden Phasen der Spaltenberechnung, um sicherzustellen, dass die richtigen Ergebnisse zurückgegeben werden.

1. Die erste Phase der Spaltenberechnung, die partielle Spaltenberechnung, errechnet die Ergebniswerte, bis der Sortierspeicher voll ist. Bei der partiellen Spaltenberechnung werden nicht berechnete Daten entgegengenommen und partielle Ergebniswerte erstellt. Wenn der Sortierspeicher voll ist, läuft der Rest der Daten auf die Festplatte über, einschließlich aller partiellen Spaltenberechnungsergebnisse, die im aktuellen Sortierspeicher berechnet wurden. Nach dem Zurücksetzen des Sortierspeichers werden neue Spaltenberechnungen gestartet.
2. Die zweite Phase der Spaltenberechnung, die Zwischenberechnung, nimmt alle übergelaufenen Sortierdurchläufe und setzt die Spaltenberechnung für die

Gruppierschlüssel fort. Die Spaltenberechnung kann nicht ausgeführt werden, weil die Gruppierschlüsselspalten eine Untergruppe der Verteilungsschlüsselspalten sind. Die Zwischenberechnung stellt aus vorhandenen partiellen Spaltenberechnungen neue partielle Spaltenberechnungen. Diese Phase findet nicht immer statt. Sie wird sowohl für partitionsinterne als auch für partitionsübergreifende Parallelität verwendet. Bei der partitionsinternen Parallelität ist die Gruppierung beendet, wenn ein globaler Gruppierschlüssel verfügbar ist. Bei partitionsübergreifender Parallelität findet sie statt, wenn der Gruppierschlüssel eine Untergruppe des Verteilungsschlüssels ist, der Gruppen auf Datenbankpartitionen verteilt, und so eine Neuverteilung zur Beendigung der Spaltenberechnung erforderlich macht. Ein ähnlicher Fall liegt vor, wenn bei partitionsinterner Parallelität jeder Agent seine übergelaufenen Sortierdurchgänge zusammengefügt hat, bevor auf einen einzigen Agenten reduziert wird, um die Spaltenberechnung zu beenden.

3. Die letzte Phase der Spaltenberechnung, die Endberechnung, verwendet alle partiellen Berechnungsergebnisse und erstellt die endgültige Spaltenberechnung. Dieser Schritt findet immer in einem Operator GROUP BY statt. Das Sortieren kann die Spaltenberechnung nicht bis zu Ende durchführen, weil es keine Garantie gibt, dass die Sortierung nicht geteilt wird. Die Komplettberechnung nimmt nicht berechnete Daten auf und produziert Endergebnisse. Wenn die Verteilung die Verwendung dieser Methode nicht verhindert, wird die Methode der Spaltenberechnung in der Regel zur Gruppierung von Daten verwendet, die bereits in der richtigen Reihenfolge vorliegen.

Optimierungsstrategien

Optimierungsstrategien für partitionsinterne Parallelität

Das Optimierungsprogramm kann einen Zugriffsplan wählen, der eine Abfrage parallel innerhalb einer Datenbankpartition ausführt, wenn ein Grad von Parallelität bei der Kompilierung der SQL-Anweisung angegeben wird.

Während der Ausführung werden mehrere Datenbankagenten, so genannte Subagenten, zur Ausführung der Abfrage erstellt. Die Anzahl von Subagenten ist kleiner oder gleich dem Grad von Parallelität, der bei der Kompilierung der SQL-Anweisung angegeben wurde.

Zur Parallelisierung unterteilt das Optimierungsprogramm den Zugriffsplan in Teile, die jeweils von einem Subagenten, sowie in einen Teil, der vom koordinierenden Agenten ausgeführt wird. Die Subagenten übergeben Daten über Tabellenwarteschlangen an den koordinierenden Agenten oder andere Subagenten. In einer Umgebung mit partitionierten Datenbanken können Subagenten Daten an Subagenten in anderen Datenbankpartitionen senden oder von ihnen empfangen.

Strategien zur partitionsinternen Parallelsuche

Tabellensuchen und Indexsuchen können parallel in derselben Tabelle oder demselben Index ausgeführt werden. Für parallele Tabellensuchen wird die Tabelle in Seiten- oder Zeilenbereiche unterteilt. Je ein Bereich von Seiten oder Zeilen wird einem Subagenten zugewiesen. Ein Subagent durchsucht den ihm zugewiesenen Bereich und erhält einen anderen Bereich zugewiesen, wenn er mit dem Durchsuchen des aktuellen Bereichs fertig ist.

Für parallele Indexsuchen wird der Index in Bereiche von Datensätzen entsprechend den Indexschlüsselwerten und der Anzahl von Indexeinträgen für einen Schlüsselwert unterteilt. Die parallele Indexsuche wird wie die parallele Tabellen-

suche mit Subagenten durchgeführt, denen jeweils ein Bereich von Datensätzen zugewiesen wird. Einem Subagenten wird ein neuer Bereich zugewiesen, wenn er die Suche im aktuellen Bereich beendet hat.

Das Optimierungsprogramm legt die Sucheinheit (entweder Seite oder Zeile) sowie die Suchgranularität fest.

Bei Parallelsuchen wird die Arbeit gleichmäßig unter den Subagenten verteilt. Der Zweck einer Parallelsuche besteht darin, die Belastung unter den Subagenten ausgewogen zu verteilen und sie gleichmäßig auszulasten. Wenn die Anzahl aktiver Subagenten gleich der Anzahl verfügbarer Prozessoren ist und die Platten nicht mit E/A-Anforderungen überlastet sind, werden die Ressourcen der Maschine effektiv genutzt.

Andere Zugriffsplanstrategien können eine unausgewogene Datenverteilung bei der Ausführung der Abfrage verursachen. Das Optimierungsprogramm wählt Parallelstrategien aus, die für eine ausgewogene Datenverteilung unter den Subagenten sorgen.

Strategien zur partitionsinternen parallelen Sortierung

Das Optimierungsprogramm kann eine der folgenden parallelen Sortierstrategien auswählen:

- **Reihumsortierung**

  Dies kann auch als *Umverteilungssortieren* bezeichnet werden. Diese Methode nutzt den gemeinsamen Speicher effizient, um die Daten so gleichmäßig wie möglich auf alle Subagenten zu verteilen. Zur Realisierung der gleichmäßigen Verteilung wird eine Art Reihumverteilungsalgorithmus verwendet. Zunächst wird ein Sortiervorgang für jeden Subagenten erstellt. Während der Einfügephase fügen Subagenten Zeilen reihum in jeden der einzelnen Sortiervorgänge ein, um eine gleichmäßigere Datenverteilung zu erzielen.
- **Partitionierte Sortierung**

  Dies ist dem reihumverteilten Sortieren insofern ähnlich, als dass für jeden Subagenten ein Sortiervorgang erstellt wird. Die Subagenten wenden eine Hashfunktion auf die Sortierspalten an, um festzulegen, in welchen Sortiervorgang eine Zeile einzufügen ist. Wenn beispielsweise die innere und die äußere Tabelle eines Mischjoins an einem partitionierten Sortiervorgang beteiligt sind, kann ein Subagent mithilfe eines Mischjoins die entsprechenden Teile verknüpfen und parallel ausgeführt werden.
- **Replizierte Sortierung**

  Diese Art der Sortierung wird verwendet, wenn jeder Subagent die gesamte Ausgabe der Sortierung benötigt. Es wird nur ein Sortiervorgang erstellt. Beim Einfügen von Zeilen in den Sortiervorgang werden die Subagenten synchronisiert. Nach Beendigung der Sortierung liest jeder Subagent das gesamte Sortierergebnis. Wenn die Anzahl von Zeilen gering ist, kann diese Art der Sortierung zum erneuten Ausgleich des Datenstroms verwendet werden.
- **Gemeinsame Sortierung**

  Diese Art der Sortierung entspricht der replizierten Sortierung, abgesehen davon, dass die Subagenten eine parallele Suche über das Sortierergebnis öffnen, um die Daten unter den Subagenten ähnlich wie bei einer Reihumsortierung zu verteilen.

Partitionsinterne temporäre Paralleltabellen

Subagenten können kooperieren, um eine temporäre Tabelle durch Einfügen von Zeilen in dieselbe Tabelle zu erstellen. Eine solche Tabelle ist eine `gemeinsame temporäre Tabelle`. Die Subagenten können private oder parallele Suchoperationen über die gemeinsam genutzte temporäre Tabelle öffnen; dies ist davon abhängig, ob der Datenstrom repliziert oder geteilt werden muss.

Strategien zur partitionsinternen parallelen Spaltenberechnung

Spaltenberechnungen (Aggregation) können von Subagenten parallel durchgeführt werden. Eine Spaltenberechnung setzt voraus, dass die Daten nach den Gruppierungsspalten geordnet sind. Wenn ein Subagent sicher sein kann, dass er alle Zeilen für eine Reihe von Gruppierungsspaltenwerten erhält, kann er eine vollständige Spaltenberechnung (Aggregation) durchführen. Dies ist möglich, wenn der Strom bereits aufgrund einer früheren partitionierten Sortierung über die Gruppierungsspalten geteilt ist.

Andernfalls kann der Subagent eine partielle Spaltenberechnung durchführen und eine andere Strategie zur Vervollständigung der Spaltenberechnung anwenden. Einige dieser Strategien sind:

- Senden der teilweise berechneten Daten an den Koordinatoragenten über eine Tabellenwarteschlange für Mischjoins. Der Koordinatoragent vervollständigt die Spaltenberechnung.
- Einfügen der teilweise berechneten Daten in eine partitionierte Sortierung. Die Sortierung wird über die Gruppierungsspalten geteilt und stellt sicher, dass alle Zeilen für eine Reihe von Gruppierungsspalten in einer Sortierpartition enthalten sind.
- Wenn der Strom zum Ausgleichen der Verarbeitung repliziert werden muss, können die teilweise berechneten Daten in eine replizierte Sortierung eingefügt werden. Die einzelnen Subagenten vervollständigen die Spaltenberechnung über die replizierte Sortierung und erhalten eine identische Kopie des Ergebnisses der Spaltenberechnung.

Strategien zu partitionsinternen parallelen Joins

Joinoperationen können von Subagenten parallel durchgeführt werden. Parallele Joinsrategien werden durch die Merkmale des Datenstroms festgelegt.

Ein Join kann parallelisiert werden, indem der Datenstrom nach der inneren und äußeren Tabelle des Joins partitioniert, repliziert oder beides wird. Zum Beispiel kann ein Join mit Verschachtelungsschleife parallelisiert werden, wenn der äußere Datenstrom für eine Parallelsuche partitioniert ist und der innere Datenstrom von jedem Subagenten unabhängig neu ausgewertet wird. Ein Mischjoin kann parallelisiert werden, wenn der innere und der äußere Datenstrom für partitionierte Sortierungen nach ihren Werten partitioniert sind.

Optimierungsstrategien für MDC-Tabellen

Wenn Sie Tabellen mit mehrdimensionalem Clustering (MDC - Multidimensional Clustering) erstellen, kann sich die Leistung vieler Abfragen verbessern, weil das Optimierungsprogramm zusätzliche Optimierungsstrategien anwenden kann. Diese Strategien basieren in erster Linie auf der verbesserten Effizienz von Blockindizes, jedoch ermöglicht der Vorteil des mehrdimensionalen Clustering auch einen schnelleren Datenabruf.

Anmerkung: Die Optimierungsstrategien für MDC-Tabellen können auch die Leistungsvorteile der partitionsinternen und partitionsübergreifenden Parallelität implementieren.

MDC-Tabellen bieten die folgenden spezifischen Vorteile:

- Dimensionsblockindexsuchen können die erforderlichen Teile der Tabelle ermitteln und schnell nur die angeforderten Blöcke durchsuchen.
- Da Blockindizes kleiner als Satz-ID-Indizes sind, arbeiten Blockindexsuchen schneller.
- AND- und OR-Joins von Indizes (Index ANDing und Index ORing) können auf Blockebene durchgeführt und mit Satz-IDs kombiniert werden.
- Daten werden garantiert in EXTENTSIZE-Speicherbereichen in Clustern gruppiert, was ein schnelleres Abrufen ermöglicht.
- Zeilen können schneller gelöscht werden, wenn ein Rollout (Auslagerung) ausgeführt werden kann.

Betrachten Sie das folgende einfache Beispiel für eine MDC-Tabelle mit dem Namen SALES, in der Dimensionen auf den Spalten **region** und **month** definiert sind:

```
SELECT * FROM SALES
    WHERE MONTH='March' AND REGION='SE'
```

Für diese Abfrage kann das Optimierungsprogramm eine Dimensionsblockindexsuche durchführen, um die Blöcke zu finden, in denen der Monat März (March) und die Region SE vorkommen. Anschließend kann es nur die resultierenden Blöcke der Tabelle durchsuchen, um die Ergebnismenge abzurufen.

Rolloutlöschung

Wenn die Bedingungen für das Löschen durch ein Rollout gegeben sind, wird ein effizienteres Verfahren zum Löschen von Zeilen aus MDC-Tabellen verwendet. Die folgenden Bedingungen müssen erfüllt sein:

- Die Anweisung DELETE wird gesucht, nicht positioniert (d. h. verwendet nicht die Klausel „WHERE CURRENT OF").
- Es gibt keine WHERE-Klauseln (alle Zeilen müssen gelöscht werden) oder die einzigen Bedingungen in der Klausel WHERE beziehen sich auf Dimensionen.
- Die Tabelle wurde nicht mit der Klausel DATA CAPTURE CHANGES definiert.
- Die Tabelle ist nicht die übergeordnete Tabelle in einer referenziellen Integritätsbeziehung.
- Für die Tabelle dürfen keine Trigger beim Löschen (ON DELETE) definiert sein.
- Die Tabelle wird in keinen MQTs (Materialized Query Tables) verwendet, die sofort aktualisiert werden.
- Eine kaskadierende Löschoperation kommt für ein Rollout in Frage, wenn der Fremdschlüssel eine Untermenge der Dimensionsspalten der Tabelle ist.
- Die Anweisung DELETE kann nicht in einer Anweisung SELECT enthalten sein, die an der temporären Tabelle ausgeführt wird, die die Menge der betroffenen Zeilen vor einer auslösenden SQL-Operation (durch die Klausel OLD TABLE AS in der Anweisung CREATE TRIGGER angegeben) angibt.

Bei einer Rolloutlöschung werden die gelöschten Datensätze nicht protokolliert. Stattdessen werden die Seiten, die die Datensätze enthalten, durch eine Neu-

formatierung von Teilen der Seiten optisch geleert. Die Änderungen an den neu formatierten Teilen werden protokolliert, die Datensätze selbst werden jedoch nicht protokolliert.

Das Standardverhalten *Rollout mit sofortiger Bereinigung* sieht vor, dass Satz-IDs (RIDs) beim Löschen bereinigt werden. Dieser Modus kann auch durch Setzen der Registrierdatenbankvariablen **DB2_MDC_ROLLOUT** auf den Wert IMMEDIATE oder durch Angeben von `IMMEDIATE` in der Anweisung SET CURRENT MDC ROLLOUT MODE angefordert werden. Es gibt keine Änderung bei der Protokollierung von Indexaktualisierungen im Vergleich zu einer normalen Löschoperation. Die Leistungsverbesserung hängt also davon ab, wie viele Satz-ID-Indizes vorhanden sind. Je weniger Satz-ID-Indizes vorhanden sind, desto stärker ist die Verbesserung (als Prozentanteil der Gesamtzeit und des gesamten Protokollspeichers).

Ein Schätzwert für den eingesparten Platz im Protokoll kann anhand der folgenden Formel ermittelt werden. Dabei ist `N` die Anzahl der gelöschten Datensätze, `S` die Gesamtgröße der gelöschten Datensätze einschließlich Systemaufwand (z. B. Nullanzeiger und VARCHAR-Längen) und `P` die Anzahl der Seiten in den Blöcken, die die gelöschten Datensätze enthalten:

```
S + 38*N - 50*P
```

Dieser Wert stellt die Verkleinerung in den tatsächlichen Protokolldaten dar. Die Einsparung beim Platzbedarf für das aktive Protokoll ist aufgrund der Einsparung des Speicherplatzes, der für Rollbackzwecke reserviert wird, doppelt so hoch.

Alternativ können Sie die Satz-ID-Indizes auch aktualisieren lassen, nachdem die Transaktion festgeschrieben wurde, indem Sie ein *Rollout mit verzögerter Bereinigung* verwenden. Dieser Modus kann auch durch Setzen der Registrierdatenbankvariablen **DB2_MDC_ROLL_OUT** auf den Wert DEFER oder durch Angeben von `DEFERRED` in der Anweisung SET CURRENT MDC ROLLOUT MODE angefordert werden. Bei einem Rollout mit verzögerter Bereinigung werden Satz-ID-Indizes asynchron im Hintergrund bereinigt, nachdem die Löschoperation festgeschrieben wurde. Diese Rolloutmethode kann zu erheblich schnelleren Löschzeiten bei sehr umfangreichen Löschungen oder Tabellen mit mehreren Satz-ID-Indizes führen. Die Geschwindigkeit der allgemeinen Bereinigungsoperation erhöht sich, weil die Indizes bei einer verzögerten Indexbereinigung parallel bereinigt werden, während bei einer sofortigen Indexbereinigung jede Zeile des Index einzeln bereinigt wird. Darüber hinaus verringert sich der Platzbedarf des Transaktionsprotokolls für die Anweisung DELETE beträchtlich, weil die asynchrone Indexbereinigung die Indexaktualisierungen pro Indexseite und nicht pro Indexschlüssel protokolliert.

Anmerkung: Ein Rollout mit verzögerter Bereinigung erfordert zusätzliche Speicherressourcen, die aus dem Datenbankzwischenspeicher zugeordnet werden. Wenn DB2 die erforderlichen Speicherstrukturen nicht zuordnen kann, schlägt der Rollout mit verzögerter Bereinigung fehl und eine Nachricht wird in das Administratorprotokoll geschrieben.

Empfehlungen für die Verwendung von Rollouts mit verzögerter Bereinigung

Wenn für Sie die Löschleistung der wichtigste Faktor ist und Satz-ID-Indizes für die Tabelle definiert sind, sollte ein Rollout mit verzögerter Bereinigung verwendet werden. Beachten Sie, dass vor der Indexbereinigung indexbasierte Suchoperationen in den durch Rollout gelöschten Blöcken je nach Umfang der gelöschten Daten

eine leichte Leistungseinbuße erfahren. Darüber hinaus sollten auch folgenden Aspekte bei der Entscheidung zwischen sofortiger und verzögerter Indexbereinigung berücksichtigt werden:

- Der Umfang der Löschoperation: Wählen Sie ein Rollout mit verzögerter Bereinigung für sehr umfangreiche Löschoperationen aus. In Fällen, in denen Dimensionslöschanweisungen für viele kleine MDC-Tabellen ausgeführt werden, kann der Aufwand für die asynchrone Bereinigung von Indexobjekten den Vorteil der Zeiteinsparung während der Löschoperationen übersteigen.
- Die Anzahl und der Typ von Indizes: Wenn die Tabelle eine Reihe von Satz-ID-Indizes hat, die eine Verarbeitung auf Zeilenebene erfordern, sollten ein Rollout mit verzögerter Bereinigung verwendet werden.
- Die Blockverfügbarkeit: Wenn Sie wünschen, dass der Blockspeicherplatz durch die Anweisung DELETE freigegeben wird, sodass er sofort nach dem Festschreiben der Löschoperation verfügbar ist, verwenden Sie einen Rollout mit sofortiger Bereinigung.
- Der Protokollspeicherbereich: Wenn der Protokollspeicherbereich begrenzt ist, sollte bei umfangreichen Löschoperationen ein Rollout mit verzögerter Bereinigung verwendet werden.
- Speicherbeschränkungen: Ein Rollout mit verzögerter Bereinigung benötigt zusätzlichen Speicherplatz im Datenbankzwischenspeicher für alle Tabellen, für die eine verzögerte Bereinigung ansteht.

Wenn Sie die Rolloutfunktionalität für Löschoperationen inaktivieren wollen, können Sie die Registrierdatenbankvariable **DB2_MDC_ROLLOUT** auf den Wert OFF setzen oder in der Anweisung SET CURRENT MDC ROLLOUT MODE den Wert `NONE` angeben.

Optimierungsstrategien für partitionierte Tabellen

Die Bezeichnung *Ausschluss von Datenpartitionen* bezieht sich auf die Fähigkeit des Datenbankservers, auf der Grundlage der Abfragevergleichselemente festzustellen, dass nur auf eine Untergruppe der Datenpartitionen einer Tabelle zugegriffen werden muss, um eine Abfrage zu erfüllen. Der Ausschluss von Datenpartitionen ist insbesondere vorteilhaft, wenn eine Entscheidungshilfeabfrage auf eine partitionierte Tabelle ausgeführt wird.

Eine partitionierte Tabelle arbeitet mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als Datenpartitionen oder Datenbereiche (RANGE) bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden. Daten aus einer gegebenen Tabelle werden in mehrere Speicherobjekte auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen partitioniert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Das folgende Beispiel veranschaulicht die Leistungsvorteile des Ausschlusses von Datenpartitionen. Dazu wird die folgende Anweisung zur Erstellung einer partitionierten Tabelle ausführt:

```
CREATE TABLE custlist(subsdate DATE, Province CHAR(2), AccountID INT)
PARTITION BY RANGE(subsdate)
(STARTING FROM '1/1/1990' IN ts1,
STARTING FROM '1/1/1991' IN ts1,
STARTING FROM '1/1/1992' IN ts1,
STARTING FROM '1/1/1993' IN ts2,
STARTING FROM '1/1/1994' IN ts2,
```

```
STARTING FROM '1/1/1995' IN ts2,
STARTING FROM '1/1/1996' IN ts3,
STARTING FROM '1/1/1997' IN ts3,
STARTING FROM '1/1/1998' IN ts3,
STARTING FROM '1/1/1999' IN ts4,
STARTING FROM '1/1/2000' IN ts4,
STARTING FROM '1/1/2001' ENDING '12/31/2001' IN ts4);
```

Nehmen Sie an, Sie interessieren sich für Kundeninformationen des Jahres 2000. Sie führen die folgende Abfrage aus:

```
SELECT * FROM custlist WHERE subsdate BETWEEN '1/1/2000' AND '12/31/2000';
```

Wie in Abb. 29 gezeigt, stellt der Datenbankserver fest, dass nur auf eine Datenpartition in Tabellenbereich 4 (’ts4’) zugegriffen werden muss, um diese Abfrage zu erfüllen.

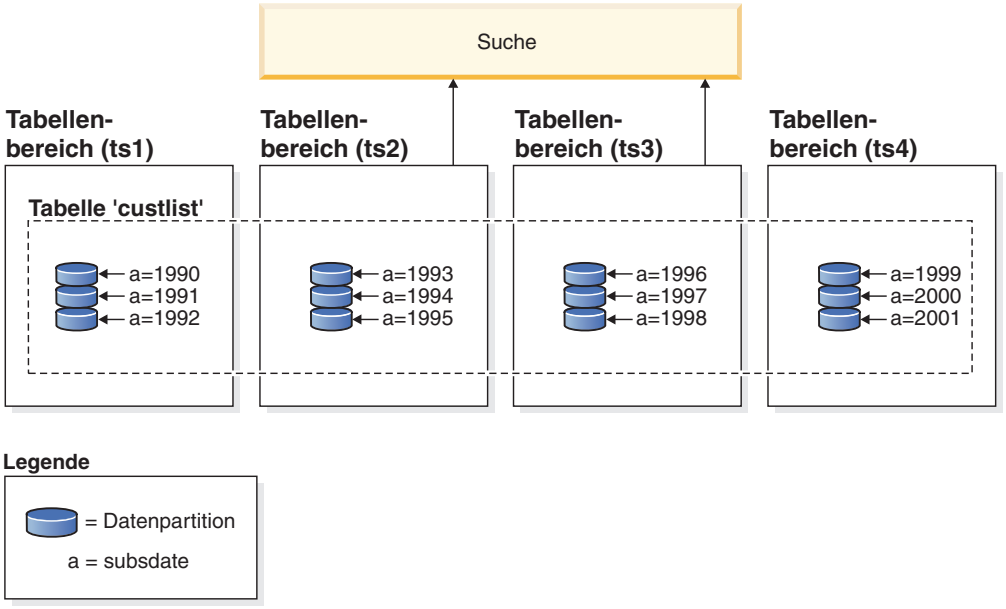


Abbildung 29. Die Leistungsvorteile des Ausschlusses von Datenpartitionen für eine partitionierte Tabelle

Ein weiteres Beispiel für den Ausschluss von Datenpartitionen (siehe Abb. 30 auf Seite 361) ist eine Indexsuche über zwei Indizes, die auf folgendem Schema basieren:

```
CREATE TABLE multi (sale_date date, region char(2))
PARTITION BY (sale_date)
(STARTING '01/01/2005' ENDING '12/31/2005' EVERY 1 MONTH);
CREATE INDEX sx ON multi(sale_date);
CREATE INDEX rx ON multi(region);
```

Sie führen die folgende Abfrage aus:

```
SELECT * FROM multi WHERE
sale_date BETWEEN '6/1/2005' AND '7/31/2005' AND REGION = 'NW';
```

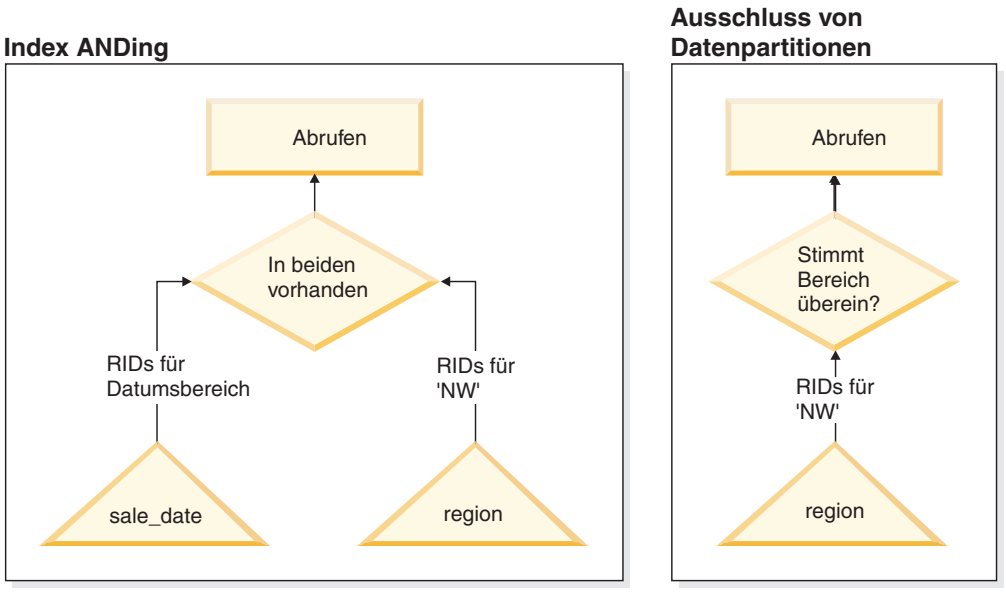


Abbildung 30. Der Entscheidungspfad des Optimierungsprogramms für die Tabellenpartitionierung und das logische Verknüpfen von Indizes über AND (Index ANDing)

Ohne Tabellenpartitionierung besteht ein wahrscheinlicher Plan in der logischen Verknüpfung von der Indizes über AND. Das Index ANDing führt die folgenden Aktionen aus:

- Lesen aller relevanten Indexeinträge aus jedem Index
- Speichern beider Gruppen von Zeilenkennungen (Satz-IDs, RIDs)
- Abgleichen der RIDs, um zu ermitteln, welche in beiden Indizes vorkommen
- Verwenden der RIDs zum Abrufen der Zeilen

Wie in Abb. 30 zu erkennen ist, wird mit der Tabellenpartitionierung der Index gelesen, um Übereinstimmungen für beide Spalten, d. h. 'region' und 'sale_date', zu finden, sodass entsprechende Zeilen schnell abgerufen werden können.

DB2 Explain

Sie können auch mithilfe von DB2 Explain den Ausschluss von Datenpartitionen ermitteln, der vom DB2-Optimierungsprogramm ausgewählt wurde. Die **DP Elim Predicates**-Informationen zeigen, welche Datenpartitionen durchsucht werden, um die folgende Abfrage zu erfüllen:

```
SELECT * FROM custlist WHERE subsdate
BETWEEN '12/31/1999' AND '1/1/2001'

Arguments:
---------
DPESTFLG: (Number of data partitions accessed are Estimated)
    FALSE
DPLSTPRT: (List of data partitions accessed)
    9-11
DPNUMPRT: (Number of data partitions accessed)
   3

DP Elim Predicates:
------------------
Range 1)
   Stop  Predicate: (Q1.A <= '01/01/2001')
   Start Predicate: ('12/31/1999' <= Q1.A)
```

```
Objects Used in Access Plan:
---------------------------

Schema: MRSRINI
Name:   CUSTLIST
Type:   Data Partitioned Table
Time of creation:              2005-11-30-14.21.33.857039
Last statistics update:        2005-11-30-14.21.34.339392
Number of columns:             3
Number of rows:                100000
Width of rows:                 19
Number of buffer pool pages:   1200
Number of data partitions:     12
Distinct row values:           No
Tablespace name:               <VARIOUS>
```

Unterstützung mehrerer Spalten

Der Ausschluss von Datenpartitionen funktioniert auch in Fällen, in denen mehrere Spalten als Tabellenpartitionierungsschlüssel verwendet werden.

Es wird zum Beispiel eine Tabelle mit der folgenden Anweisung erstellt:

```
CREATE TABLE sales(year INT, month INT)
 PARTITION BY RANGE(year, month)
 (STARTING FROM (2001, 1) ENDING AT(2001,3) IN ts1,
 ENDING AT(2001,6) IN ts2,
 ENDING AT(2001,9) IN ts3,
 ENDING AT(2001,12) IN ts4,
 ENDING AT(2002,3) IN ts5,
 ENDING AT(2002,6) IN ts6,
 ENDING AT(2002,9) IN ts7,
 ENDING AT(2002,12) IN ts8)
```

Anschließend wird die folgende Abfrage ausgeführt:

```
 SELECT * FROM sales WHERE year = 2001 AND month < 8
```

Das Abfrageoptimierungsprogramm folgert, dass zur Erfüllung dieser Abfrage nur auf die Datenpartitionen in 'ts1', 'ts2' und 'ts3' zugegriffen werden muss.

Anmerkung: Wenn der Tabellenpartitionierungsschlüssel aus mehreren Spalten gebildet wird, ist der Ausschluss von Datenpartitionen nur möglich, wenn Vergleichselemente für die führenden Spalten des zusammengesetzten Schlüssels verwendet werden, da nicht führende Spalten, die im Tabellenpartitionierungsschlüssel verwendet werden, nicht unabhängig sind.

Unterstützung mehrerer Bereiche

Es ist möglich, einen Ausschluss von Datenpartitionen für mehrere Bereiche zu erzielen (d. h. durch logisches Verknüpfen von Bereichen über OR). An der im vorigen Beispiel erstellten Tabelle wird zum Beispiel die folgende Abfrage ausgeführt:

```
 SELECT * FROM sales
 WHERE (year = 2001 AND month <= 3) OR (year = 2002 and month >= 10)
```

Der Datenbankserver greift nur auf Daten für das erste Quartal von 2001 und das letzte Quartal von 2002 zu.

Generierte Spalten

Sie können generierte Spalten als Tabellenpartitionierungsschlüssel verwenden.

Sie können zum Beispiel die folgende Anweisung ausführen:

```
CREATE TABLE sales(a INT, b INT GENERATED ALWAYS AS (a / 5))
 IN ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10
 PARTITION BY RANGE(b)
 (STARTING FROM (0) ENDING AT(1000) EVERY (50))
```

In diesem Fall werden Vergleichselemente für die generierte Spalte zum Ausschluss von Datenpartitionen verwendet. Wenn der Ausdruck zur Generierung der Spalten außerdem monoton ist, übersetzt der Datenbankserver Vergleichselemente für die Quellenspalten in Vergleichselemente für die generierten Spalten, sodass der Ausschluss von Datenpartitionen über die generierten Spalten erfolgen kann.

Betrachten Sie zum Beispiel die folgende Abfrage:

```
 SELECT * FROM sales WHERE a > 35
```

In diesem Fall generiert der Datenbankserver aus dem Vergleichselement für a (a > 35) ein zusätzliches Vergleichselement für b (b > 7), um den Ausschluss von Datenpartitionen zu ermöglichen.

Joinvergleichselemente

Joinvergleichselemente können ebenfalls beim Ausschluss von Datenpartitionen verwendet werden, wenn das Joinvergleichselement auf die Ebene des Tabellenzugriffs verschoben wird (Pushdown). Das Joinvergleichselement wird nur für die innere Tabelle eines Joins mit Verschachtelungsschleife (NLJN, Nested Loop Join) auf die Tabellenzugriffsebene verschoben.

Betrachten Sie zum Beispiel die folgenden Tabellen:

```
CREATE TABLE T1(A INT, B INT)
 PARTITION BY RANGE(A, B)
 (STARTING FROM (1, 1)
 ENDING (1,10) IN ts1, ENDING (1,20) IN ts2,
 ENDING (2,10) IN ts3, ENDING (2,20) IN ts4,
 ENDING (3,10) IN ts5, ENDING (3,20) IN ts6,
 ENDING (4,10) IN ts7, ENDING (4,20) IN ts8)

CREATE TABLE T2 (A INT, B INT)
```

Verwendete Vergleichselemente:

```
P1: T1.A = T2.A
P2: T1.B > 15
```

In diesem Beispiel lassen sich die genauen Datenpartitionen, auf die zugegriffen wird, wegen unbekannter Werte der äußeren Tabelle des Joins beim Kompilieren nicht bestimmen. In diesem Fall und ebenso in Fällen, in denen Hostvariablen oder Parametermarken verwendet werden, erfolgt der Ausschluss von Datenpartitionen bei der Ausführung, wenn die erforderlichen Werte gebunden werden.

Bei der Ausführung erfolgt, wenn T1 die innere Tabelle eines Joins mit Verschachtelungsschleife (NLJN) ist, der Ausschluss von Datenpartitionen auf der Basis der Vergleichselemente für jeden äußeren Wert von T2.A dynamisch. Bei der Ausführung werden die Vergleichselemente T1.A = 3 und T1.B > 15 für den Wert T2.A = 3

der äußeren Tabelle angewendet. Dadurch werden die Datenpartitionen in den Tabellenbereichen 'ts6' und 'ts7' für den Zugriff ermittelt.

Nehmen Sie an, dass die Spalten A in den Tabellen T1 und T2 folgende Werte enthalten:

Äußere Tabelle T2: Spalte A	Innere Tabelle T1: Spalte A	Innere Tabelle T1: Spalte B	Innere Tabelle T1: Position der Datenpartition
2	3	20	ts6
3	2	10	ts3
3	2	18	ts4
	3	15	ts6
	1	40	ts3

Für den Join mit Verschachtelungsschleife (unter Annahme einer Tabellensuche für die innere Tabelle) führt der Datenbankmanager die folgenden Schritte aus:

1. Er liest die erste Zeile aus T2. Der Wert für A ist 2.
2. Er bindet den Wert T2.A (d. h. 2) an die Spalte T2.A im Joinvergleichselement T1.A = T2.A. Aus dem Vergleichselement wird T1.A = 2.
3. Er wendet den Ausschluss von Datenpartitionen unter Verwendung der Vergleichselemente T1.A = 2 und T1.B > 15 an. Dies qualifiziert die Datenpartitionen in den Tabellenbereichen 'ts4' und 'ts5'.
4. Er durchsucht die Datenpartitionen in den Tabellenbereichen 'ts4' und 'ts5' von Tabelle T1, bis eine Zeile nach Anwendung von T1.A = 2 und T1.B > 15 gefunden wird. Die erste qualifizierte Zeile, die gefunden wird, ist die dritte Zeile von T1.
5. Er verknüpft die übereinstimmenden Zeilen.
6. Er durchsucht die Datenpartitionen in den Tabellenbereichen 'ts4' und 'ts5', bis die nächste Übereinstimmung (mit T1.A = 2 und T1.B > 15) gefunden wird. In diesem Fall werden keine weiteren Zeilen gefunden.
7. Er wiederholt die Schritte 1 bis 6 für die nächste Zeile von T2 (d. h. er nimmt den Wert 3 aus Spalte A). Dieses Verfahren wird fortgesetzt, bis alle Zeilen von T2 durchlaufen sind.

Materialized Query Tables (MQTs)

MQTs bieten eine leistungsstarke Möglichkeit, die Antwortzeit für komplexe Abfragen zu verbessern, insbesondere für Abfragen, für die einige der folgenden Operationen erforderlich sind:

- Erstellen von Ergebnisdaten für eine oder mehrere Dimensionen
- Joins und Ergebnisberechnung von Daten einer Gruppe von Tabellen
- Bereitstellen von Daten aus einer häufig genutzten Untergruppe von Daten, d. h. einer sofort für Verarbeitungsoperationen bereiten horizontalen oder vertikalen Partition
- Erneutes Partitionieren von Daten aus einer Tabelle bzw. einem Teil einer Tabelle in einer partitionierten Datenbankumgebung

In den SQL- und XQuery-Compiler sind Kenntnisse über MQTs integriert. Im Compiler werden in der Phase des Umschreibens von Abfragen und durch das Optimierungsprogramm Abfragen mit MQTs verglichen, um zu ermitteln, ob anstelle einer Abfrage, die auf die Basistabellen zugreift, eine MQT verwendet

werden kann. Wenn eine MQT verwendet wird, kann die EXPLAIN-Einrichtung Informationen darüber bereitstellen, welche MQT ausgewählt wurde.

Da sich MQTs in vielerlei Hinsicht wie reguläre Tabellen verhalten, treffen die Richtlinien zum Optimieren des Datenzugriffs unter Verwendung von Tabellenbereichsdefinitionen, durch Erstellen von Indizes und Ausführen des Dienstprogramms RUNSTATS auch auf MQTs zu.

Zur Veranschaulichung der Leistungsfähigkeit von MQTs wird im folgenden Beispiel eine mehrdimensionale Analyseabfrage dargestellt und erläutert, wie sie von MQTs profitiert.

Nehmen Sie für dieses Beispiel ein Datenbank-Data-Warehouse an, das eine Gruppe von Kunden und eine Gruppe von Kreditkartenkonten enthält. Das Data-Warehouse zeichnet die Gruppe der Transaktionen auf, die mit den Kreditkarten durchgeführt wurden. Alle Transaktionen enthalten eine Anzahl von Artikeln, die gemeinsam gekauft wurden. Dieses Schema wird als Mehrfachsternschema (Multi-Star) klassifiziert, weil es mit zwei großen Tabellen arbeitet, von denen eine die Artikel der Transaktionen und die andere die Kauftransaktionen enthält.

Eine Transaktion wird durch drei hierarchische Dimensionen beschrieben: Produkt, Standort und Zeit. Die Produkthierarchie wird in zwei normalisierten Tabellen gespeichert, die die Produktgruppe und die Produktlinie darstellen. Die Standorthierarchie enthält Informationen zu Ort, Bundesland, sowie Land oder Region, die in einer einzigen denormalisierten Tabelle dargestellt werden. Die Zeithierarchie enthält Informationen zu Tag, Monat und Jahr und ist in einem einzigen Datumsfeld codiert. Die Datumsdimensionen werden aus dem Datumsfeld der Transaktion unter Verwendung integrierter Funktionen extrahiert. Andere Tabellen in diesem Schema stellen die Kontoinformationen für Kunden und Kundeninformationen dar.

Eine MQT wird mit der Summe und der Anzahl der Verkäufe für jede Stufe der folgenden Hierarchien erstellt:

- Produkt
- Standort
- Zeit bestehend aus Jahr, Monat, Tag

Viele Abfragen können aus diesen zusammengefassten Ergebnisdaten erfüllt werden. Das folgende Beispiel zeigt, wie eine gespeicherte Abfrage erstellt wird, die die Summe und die Anzahl der Verkäufe für die Dimensionen 'Produktgruppe' (Product Group) und 'Produktlinie (Product Line), für die Dimensionen 'Ort' (City), 'Bundesland' (State) und 'Land' (Country) und für die Dimension 'Zeit' (Time) berechnet. Das Beispiel enthält auch einige weitere Spalten in der Klausel GROUP BY.

```
CREATE TABLE dba.PG_SALESSUM
  AS (
      SELECT l.id AS prodline, pg.id AS pgroup,
             loc.country, loc.state, loc.city,
             l.name AS linename, pg.name AS pgname,
             YEAR(pdate) AS year, MONTH(pdate) AS month,
             t.status,
             SUM(ti.amount) AS amount,
             COUNT(*) AS count
      FROM   cube.transitem AS ti, cube.trans AS t,
             cube.loc AS loc, cube.pgroup AS pg,
             cube.prodline AS l
      WHERE  ti.transid = t.id
         AND ti.pgid = pg.id
```

```
         AND pg.lineid = l.id
         AND t.locid = loc.id
         AND YEAR(pdate) > 1990
      GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
               year(pdate), month(pdate), t.status, l.name, pg.name
     )
DATA INITIALLY DEFERRED REFRESH DEFERRED;

REFRESH TABLE dba.SALESCUBE;
```

Abfragen, die solche vorberechneten Summen nutzen können, sind unter anderem:

- Verkaufsdaten nach Monat und Produktgruppe
- Gesamtvertrieb für Jahre nach 1990
- Verkauf für 1995 oder 1996
- Summe des Verkaufs für eine Produktgruppe oder Produktlinie
- Summe des Verkaufs für eine bestimmte Produktgruppe oder Produktlinie UND für 1995, 1996
- Summe des Verkaufs für ein bestimmtes Land

Obwohl die präzise Anwort für keine dieser Abfragen in der MQT enthalten ist, könnte der Aufwand zur Berechnung der Antwort mithilfe der MQT erheblich geringer ausfallen als bei Verwendung der umfangreichen Basistabelle, da ein Teil der für die Antwort benötigten Berechnungen bereits erfolgt ist. MQTs können die Notwendigkeit von aufwendigen Joins, Sortierungen und Spaltenberechnungen von Basisdaten verringern.

Die folgenden Beispielabfragen könnten beträchtliche Leistungsverbesserungen erfahren, da sie die bereits berechneten Ergebnisse der Beispiel-MQT nutzen könnten.

Das erste Beispiel liefert die Gesamtverkäufe für 1995 und 1996:

```
SET CURRENT REFRESH AGE=ANY

SELECT YEAR(pdate) AS year, SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY year(pdate);
```

Das zweite Beispiel liefert die Gesamtverkäufe nach Produktgruppe für 1995 und 1996:

```
SET CURRENT REFRESH AGE=ANY

SELECT pg.id AS "PRODUCT GROUP",
       SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY pg.id;
```

Je größer die Basistabellen sind, desto höher können die Leistungsverbesserungen in Antwortzeiten ausfallen, weil die MQT langsamer anwächst als die Basistabelle. MQTs können sich überlappende Arbeitsschritte von Abfragen effektiv vermeiden helfen, indem sie die Berechnungen einmal bei ihrer Erstellung und Aktualisierung (REFRESH) durchführen und ihren Inhalt anschließend vielen Abfragen zur Verfügung stellen.

EXPLAIN-Einrichtung

Der SQL- oder XQuery-Compiler kann Informationen über den Zugriffsplan und die Umgebung statischer oder dynamischer SQL- oder XQuery-Anweisungen erfassen. Die erfassten Informationen helfen Ihnen zu verstehen, wie einzelne SQL- oder XQuery-Anweisungen ausgeführt werden, sodass Sie die Anweisungen und die Konfiguration des Datenbankmanagers zur Leistungsverbesserung optimieren können.

Sie können EXPLAIN-Daten zu folgenden Zwecken erfassen und verwenden:

- Um zu verstehen, wie der Datenbankmanager auf Tabellen und Indizes zur Erfüllung Ihrer Abfrage zugreift.
- Um Ihre Maßnahmen zur Leistungsverbesserung zu beurteilen.

  Wenn Sie einen Aspekt des Datenbankmanagers, der SQL- oder XQuery-Anweisungen oder der Datenbank ändern, sollten Sie die EXPLAIN-Daten untersuchen, um herauszufinden, wie sich Ihre Maßnahme auf die Leistung ausgewirkt hat.

Zu den erfassten Informationen gehören folgende:

- Informationen über die Reihenfolge der Operationen zur Verarbeitung der Abfrage
- Informationen über den Aufwand
- Vergleichselemente und Selektivitätsschätzwerte für jedes Vergleichselement
- Statistikdaten für alle Objekte, auf die in der SQL- oder XQuery-Anweisung zum Zeitpunkt der EXPLAIN-Datenerfassung verwiesen wird
- Werte für die Hostvariablen, Parametermarken oder Sonderregister, die zur Reoptimierung der SQL- oder XQuery-Anweisung verwendet werden

Bevor Sie EXPLAIN-Informationen erfassen können, erstellen Sie die relationalen Tabellen, in denen das Optimierungsprogramm die EXPLAIN-Informationen speichert, und definieren die Sonderregister, die die Art der zu erfassenden EXPLAIN-Informationen festlegen.

Zum Anzeigen von EXPLAIN-Informationen können Sie entweder ein Befehlszeilentool oder Visual Explain verwenden. Mit dem verwendeten Tool wird festgelegt, wie Sonderregister definiert werden, die festlegen, welche EXPLAIN-Daten erfasst werden. Wenn Sie zum Beispiel davon ausgehen, dass Sie nur Visual Explain verwenden, brauchen Sie nur Momentaufnahmeninformationen zu erfassen. Wenn Sie beabsichtigen, eine detaillierte Analyse mit einem der Befehlszeilendienstprogramme oder mit angepassten SQL- oder XQuery-Anweisungen für die EXPLAIN-Tabellen durchzuführen, sollten Sie alle EXPLAIN-Informationen erfassen.

Richtlinien zur Verwendung von EXPLAIN-Informationen

EXPLAIN-Informationen dienen in erster Linie den beiden folgenden Zwecken:

- Analysieren der Gründe, aus denen sich die Anwendungsleistung geändert hat
- Beurteilen von Maßnahmen zur Leistungsoptimierung

Analysieren von Leistungsänderungen

Um sich ein besseres Verständnis für die Ursachen für Änderungen in der Abfrageleistung zu verschaffen, benötigen Sie EXPLAIN-Informationen vor und nach den Änderungen, die Sie durch die folgenden Schritte erhalten können:

- Erfassen Sie EXPLAIN-Informationen für die Abfrage, bevor Sie Änderungen vornehmen, und sichern Sie die resultierenden EXPLAIN-Tabellen. Alternativ könnten Sie auch die Ausgabe aus dem EXPLAIN-Tool `db2exfmt` sichern.
- Sichern oder drucken Sie die aktuellen Katalogstatistiken, wenn Sie zum Anzeigen der entsprechenden Informationen nicht auf Visual Explain zugreifen wollen oder können. Zu diesem Zweck können Sie auch das Produktivitätstool `db2look` verwenden.
- Sichern oder drucken Sie die Anweisungen der Datendefinitionssprache (DDL), einschließlich der Anweisungen für CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE TABLESPACE.

Die Informationen, die Sie auf diese Weise erfassen, stellen einen Referenzpunkt für zukünftige Analysen dar. Bei dynamischen SQL- oder XQuery-Anweisungen können Sie diese Informationen bei der ersten Ausführung Ihrer Anwendung erfassen. Bei statischen SQL- oder XQuery-Anweisungen können Sie diese Informationen auch beim Binden erfassen. Zur Analyse einer Leistungsänderung vergleichen Sie die Informationen, die Sie erfasst haben, mit Informationen, die Sie über die Abfrage und die Umgebung erfassen, wenn Sie Ihre Analyse beginnen.

Ein einfaches Beispiel wäre eine Analyse, die ergäbe, dass ein Index nicht mehr als Bestandteil des Zugriffspfads verwendet wird. Mithilfe der in Visual Explain angezeigten Informationen der Katalogstatistiken könnten Sie feststellen, dass die Anzahl von Indexstufen (Spalte NLEVELS) nun wesentlich höher ist als zu dem Zeitpunkt, als die Abfrage zum ersten Mal an die Datenbank gebunden wurde. Sie könnten in diesem Fall eine der folgenden Maßnahmen durchführen:

- Reorganisieren des Index
- Erfassen neuer Statistikdaten für die Tabelle und Indizes
- Sammeln von EXPLAIN-Informationen beim Rebind der Abfrage

Nach der Durchführung einer dieser Maßnahmen untersuchen Sie den Zugriffsplan erneut. Wenn der Index wieder verwendet wird, ist die Leistung der Abfrage vielleicht kein Problem mehr. Wenn der Index weiterhin nicht verwendet wird oder die Leistung problematisch bleibt, führen Sie eine zweite Maßnahme durch und untersuchen die Ergebnisse. Wiederholen Sie diese Schritte, bis das Problem gelöst ist.

Beurteilen von Maßnahmen zur Leistungsoptimierung

Sie haben die Möglichkeit, die Abfrageleistung durch eine Reihe von Maßnahmen zu verbessern, zu denen das Anpassen von Konfigurationsparametern, das Hinzufügen von Containern, das Erfassen aktueller Katalogstatistiken u. a. gehören.

Nach einer Änderung in einem dieser Bereiche können Sie mit der EXPLAIN-Einrichtung die Auswirkungen ermitteln, die die Änderung auf den ausgewählten Zugriffsplan hat. Wenn sie zum Beispiel einen Index oder eine MQT (Materialized Query Table, gespeicherte Abfragetabelle) gemäß den Richtlinien für Indizes hinzufügen, können Sie mithilfe der EXPLAIN-Daten feststellen, ob der Index oder die MQT tatsächlich in der erwarteten Weise genutzt wird.

Obwohl die EXPLAIN-Ausgabe Informationen liefert, die Ihnen ermöglichen, den ausgewählten Zugriffsplan und seinen relativen Aufwand zu ermitteln, besteht die einzige Möglichkeit, die Verbesserung der Leistung für eine Abfrage genau zu messen, in der Anwendung von Vergleichstesttechniken (Benchmark Tests).

Richtlinien zur Erfassung von EXPLAIN-Informationen

EXPLAIN-Daten werden bei der Kompilierung einer SQL- oder XQuery-Anweisung erfasst, wenn Sie dies anfordern. Bei der Anforderung von EXPLAIN-Daten sollten Sie berücksichtigen, wie die erfassten Informationen später verwendet werden sollen.

Anmerkung:

1. Wenn SQL- oder XQuery-Anweisungen zum inkrementellen Binden während der Ausführung kompiliert werden, werden Daten während der Bearbeitungszeit, nicht während der Bindezeit, in die EXPLAIN-Tabellen eingetragen. Für solche Anweisungen sind das Qualifikationsmerkmal und die Berechtigungs-ID, die in die EXPLAIN-Tabelle eingetragen werden, die des Paketeigners und nicht die des Benutzers, der das Paket ausführt.
2. Die EXPLAIN-Informationen werden nur erfasst, wenn die SQL- oder XQuery-Anweisung kompiliert wird. Nach der Erstkompilierung werden dynamische Abfrageanweisungen erneut kompiliert, wenn eine Änderung an der Umgebung dies erfordert oder wenn die EXPLAIN-Einrichtung aktiv ist. Wenn Sie dieselbe Anweisung PREPARE mehrmals für die gleiche Abfrageanweisung ausführen, erfolgt das Kompilieren der Anweisung und das Erfassen der EXPLAIN-Daten jedes Mal, wenn diese Anweisung vorbereitet oder ausgeführt wird.
3. Wenn ein Paket mit der Bindeoption REOPT ONCE/ALWAYS gebunden wird, werden SQL- oder XQuery-Anweisungen, die Hostvariablen, Parametermarken, globale Variablen oder Sonderregister enthalten, kompiliert. Der Zugriffspfad wird mit den realen Werten dieser Variablen, wenn sie bekannt sind, und mit Standardschätzwerten, wenn die Werte nicht bekannt sind, erstellt.
4. Wenn die Klausel FOR REOPT ONCE verwendet wird, wird ein Versuch unternommen, die angegebene SQL- oder XQuery-Anweisung mit der gleichen Anweisung im Paketcache abzugleichen. Die Werte dieser bereits reoptimierten Abfrageanweisung im Cache werden zur Reoptimierung der angegebenen Abfrageanweisung verwendet. Die EXPLAIN-Tabellen enthalten in diesem Fall den neu generierten, reoptimierten Zugriffsplan und die Werte, die für diese Reoptimierung verwendet wurden, sofern der Benutzer über die erforderlichen Zugriffsrechte verfügt.
5. In einem System mit mehreren Partitionen muss die Anweisung in der gleichen Datenbankpartition mit EXPLAIN bearbeitet werden, in der sie auch ursprünglich kompiliert und mit REOPT ONCE reoptimiert wurde. Ansonsten wird ein Fehler zurückgegeben.

Erfassen von Informationen in den EXPLAIN-Tabellen

- **Statische SQL- und XQuery-Anweisungen oder SQL- und XQuery-Anweisungen zum inkrementellen Binden:**

  Geben Sie für die Befehle BIND oder PREP die Optionen EXPLAIN ALL oder EXPLAIN YES an, oder fügen Sie eine statische EXPLAIN-Anweisung in das Quellenprogramm ein.
- **Dynamische SQL- und XQuery-Anweisungen**:

  Informationen für EXPLAIN-Tabellen werden in folgenden Fällen erfasst:

– Das Sonderregister CURRENT EXPLAIN MODE enthält einen der folgenden Werte:
  - YES: Der SQL- und XQuery-Compiler erfasst EXPLAIN-Daten und führt die Abfrageanweisung aus.
  - EXPLAIN: Der SQL- und XQuery-Compiler erfasst EXPLAIN-Daten, führt die Abfrageanweisung jedoch nicht aus.
  - RECOMMEND INDEXES: Der SQL- und XQuery-Compiler erfasst EXPLAIN-Daten und die Daten zu den empfohlenen Indizes werden in der Tabelle ADVISE_INDEX gespeichert. Die Abfrageanweisung wird jedoch nicht ausgeführt.
  - EVALUATE INDEXES: Der SQL- und XQuery-Compiler verwendet die vom Benutzer in die Tabelle ADVISE_INDEX eingefügten Indizes zur Bewertung. Im Modus EVALUATE INDEXES werden alle dynamischen Anweisungen mit EXPLAIN so bearbeitet, als wären diese virtuellen Indizes verfügbar. Der Abfragecompiler wählt anschließend die virtuellen Indizes aus, wenn sie die Leistung der Anweisungen verbessern. Ansonsten werden die Indizes ignoriert. Durch eine Analyse der EXPLAIN-Ergebnisse können Sie feststellen, ob die vorgeschlagenen Indizes nützlich wären.
  - REOPT: Der Abfragecompiler erfasst während der Ausführung EXPLAIN-Daten für statische oder dynamische SQL- oder XQuery-Anweisungen bei der Reoptimierung der Anweisung, wenn für die Hostvariablen, Sonderregister, globalen Variablen oder Parametermarken tatsächliche Werte zur Verfügung stehen.

- Die Option EXPLAIN ALL wurde im Befehl BIND oder PREP angegeben. Der Abfragecompiler erfasst während der Ausführung EXPLAIN-Daten für dynamisches SQL und XQuery, selbst wenn das Sonderregister CURRENT EXPLAIN MODE auf NO gesetzt ist. Die SQL- oder XQuery-Anweisung wird ebenfalls ausgeführt und gibt die Abfrageergebnisse zurück.

Erfassen von EXPLAIN-Momentaufnahmen

Wenn eine EXPLAIN-Momentaufnahme (Snapshot) angefordert wird, werden EXPLAIN-Informationen in der Spalte SNAPSHOT der Tabelle EXPLAIN_STATEMENT in dem für Visual Explain erforderlichen Format gespeichert. Dieses Format kann von anderen Anwendungen nicht verwendet werden. Weitere Informationen zum Inhalt der Informationen der EXPLAIN-Momentaufnahmen erhalten Sie direkt mithilfe von Visual Explain. Zu diesen Informationen gehören Informationen zu Datenobjekten und Datenoperatoren.

EXPLAIN-Momentaufnahmedaten werden erfasst, wenn eine SQL- oder XQuery-Anweisung kompiliert wird und EXPLAIN-Daten wie folgt angefordert wurden:

- **Statische SQL- und XQuery-Anweisungen oder SQL- und XQuery-Anweisungen zum inkrementellen Binden:**

  Eine EXPLAIN-Momentaufnahme wird erfasst, wenn eine der beiden Klauseln EXPLSNAP ALL oder EXPLSNAP YES für die Befehle BIND oder PREP angegeben wird oder wenn das Quellenprogramm eine statische Anweisung EXPLAIN mit der Klausel FOR SNAPSHOT oder WITH SNAPSHOT enthält.

- **Dynamische SQL- und XQuery-Anweisungen**:

  Eine EXPLAIN-Momentaufnahme wird in folgenden Fällen erfasst:

  – Sie führen eine Anweisung EXPLAIN mit der Klausel FOR SNAPSHOT oder WITH SNAPSHOT aus. Mit der Klausel FOR SNAPSHOT werden nur

Momentaufnahmedaten erfasst. Mit der Klausel WITH SNAPSHOT werden neben den Momentaufnahmedaten auch sämtliche anderen EXPLAIN-Informationen erfasst.

– Das Sonderregister CURRENT EXPLAIN SNAPSHOT enthält einen der folgenden Werte:
  - YES: Der Abfragecompiler erfasst EXPLAIN-Momentaufnahmedaten und führt die SQL- oder XQuery-Anweisung aus.
  - EXPLAIN: Der Abfragecompiler erfasst EXPLAIN-Momentaufnahmedaten, führt die SQL- oder XQuery-Anweisung jedoch nicht aus.
– Sie geben die Option EXPLSNAP ALL im Befehl BIND oder PREP an. Der Abfragecompiler erfasst EXPLAIN-Momentaufnahmedaten bei der Ausführung, selbst wenn das Sonderregister CURRENT EXPLAIN SNAPSHOT auf NO gesetzt ist. Die SQL- oder XQuery-Anweisung wird ebenfalls ausgeführt.

Richtlinien zur Analyse von EXPLAIN-Informationen

Der primäre Zweck der EXPLAIN-Informationen ist die Analyse des Zugriffspfads für Abfrageanweisungen. Die Analyse der EXPLAIN-Daten kann Ihnen auf verschiedene Weise helfen, Ihre Abfragen und Ihre Umgebung zu optimieren. Betrachten Sie die folgende Art einer Analyse:

- **Verwendung von Indizes**

  Die geeigneten Indizes können die Leistung erheblich fördern. Anhand der EXPLAIN-Ausgabe können Sie ermitteln, ob die von Ihnen für eine bestimmte Gruppe von Abfragen erstellten Indizes tatsächlich verwendet werden. In der EXPLAIN-Ausgabe sollten Sie folgende Bereiche auf die Verwendung von Indizes hin überprüfen:
  – Joinvergleichselemente
  – Lokale Vergleichselemente
  – Die Klausel GROUP BY
  – Die Klausel ORDER BY
  – Die Klausel WHERE XMLEXISTS
  – Die SELECT-Liste

  Sie können die EXPLAIN-Einrichtung auch verwenden, um zu ermitteln, ob ein anderer Index anstelle des vorhandenen Index oder kein Index verwendet werden könnte. Führen Sie nach der Erstellung eines neuen Index den Befehl RUNSTATS aus, um Statistikdaten für diesen Index zu erfassen, und kompilieren Sie die Abfrage erneut. Mit der Zeit werden Sie möglicherweise durch die EXPLAIN-Daten feststellen, dass anstelle einer Indexsuche eine Tabellensuche verwendet wird. Dies kann sich aus einer Änderung in der Clusterbildung der Tabellendaten ergeben. Wenn der zuvor verwendete Index nun ein niedriges Clusterverhältnis aufweist, sollten Sie in Erwägung ziehen, die Tabelle zu reorganisieren, um ihre Daten in Clustern entsprechend dem Index anzuordnen, den Befehl RUNSTATS auszuführen, um Statistikdaten für Index und Tabelle zu erfassen, und anschließend die Abfrage erneut zu kompilieren. Prüfen Sie die EXPLAIN-Ausgabe erneut, um herauszufinden, ob das Reorganisieren der Tabelle zu einer Verbesserung des Zugriffsplans geführt hat.

- **Zugriffstyp**

  Analysieren Sie die EXPLAIN-Ausgabe und suchen Sie nach Arten des Zugriffs auf die Daten, die für die Art von Anwendung, die Sie ausführen, normalerweise nicht optimal sind. Zum Beispiel:
  – **OLTP-Abfragen (Online Transaction Processing)**

In OLTP-Anwendungen werden häufig Indexsuchen mit Vergleichselementen durchgeführt, die eine Bereichsbegrenzung vornehmen, weil diese Anwendungen in der Regel nur wenige Zeilen zurückgeben, die den mit einem Gleichheitsvergleichselement für eine Schlüsselspalte angegebenen Bedingungen entsprechen. Wenn Ihre OLTP-Abfragen eine Tabellensuche verwenden, können Sie die EXPLAIN-Daten analysieren, um herauszufinden, warum keine Indexsuche verwendet wurde.

– **Reine Suchabfragen**

  Die Suchbedingungen für eine „reine Suchabfrage" können sehr vage sein, was jedoch bewirkt, dass eine große Menge von Zeilen den Bedingungen entspricht. Wenn sich Benutzer normalerweise nur einige Seiten der Ausgabedaten anzeigen lassen, können Sie dafür sorgen, dass nicht die gesamte Antwortmenge errechnet werden muss, bevor einige Ergebnisse zurückgegeben werden. In diesem Fall unterscheiden sich die Ziele des Benutzers vom grundlegenden Verarbeitungsprinzip des Optimierungsprogramms, das versucht, den Ressourcenbedarf für die gesamte Abfrage und nicht nur für die ersten wenigen Anzeigen mit Daten zu minimieren.

  Wenn zum Beispiel die EXPLAIN-Ausgabe zeigt, dass Operatoren sowohl für Mischjoins als auch für Sortierungen im Zugriffsplan verwendet wurden, wird die gesamte Antwortmenge in einer temporären Tabelle gespeichert, bevor Zeilen an die Anwendung zurückgegeben werden. In diesem Fall können Sie versuchen, den Zugriffsplan durch die Verwendung der Klausel OPTIMIZE FOR in der SELECT-Anweisung zu ändern. Wenn Sie diese Option angeben, kann das Optimierungsprogramm versuchen, einen Zugriffsplan auszuwählen, der nicht die gesamte Antwortmenge in einer temporären Tabelle erstellt, bevor die ersten Zeilen an die Anwendung zurückgegeben werden.

- **Joinmethoden**

  Wenn bei einer Abfrage zwei Tabellen verknüpft werden, sollten Sie die Art der verwendeten Joinverarbeitung überprüfen. Joins mit mehreren Zeilen, wie sie zum Beispiel bei Abfragen von Entscheidungshilfedaten auftreten, werden in der Regel schneller als Hash-Joins oder Mischjoins ausgeführt. Joins, an denen nur einige wenige Zeilen beteiligt sind, wie zum Beispiel für OLTP-Abfragen, sind zumeist als Joins mit Verschachtelungsschleife effizienter. In beiden Fällen kann es jedoch auch Umstände geben, wie beispielsweise die Verwendung lokaler Vergleichselemente oder Indizes, die die normale Verarbeitung dieses Joins möglicherweise ändern.

Verwenden von Zugriffsplänen zur Selbstdiagnose von Leistungsproblemen bei Anweisungen REFRESH TABLE und SET INTEGRITY

Mit den Anweisungen EXPLAIN FOR REFRESH TABLE und EXPLAIN FOR SET INTEGRITY können Sie Zugriffspläne generieren, die zur Selbstdiagnose von Leistungsproblemen mit den Anweisungen REFRESH TABLE und SET INTEGRITY verwendet werden können. Diese Möglichkeit hilft Ihnen, Ihre MQTs (Materialized Query Tables) besser zu verwalten.

Zum Abrufen des Zugriffsplans für eine Anweisung REFRESH TABLE oder SET INTEGRITY können Sie eine der folgenden Methoden verwenden:

- Verwenden Sie die Option EXPLAIN PLAN FOR REFRESH TABLE oder EXPLAIN PLAN FOR SET INTEGRITY in der Anweisung EXPLAIN.

- Setzen Sie das Sonderregister CURRENT EXPLAIN MODE auf EXPLAIN, bevor Sie die Anweisung REFRESH TABLE bzw. SET INTEGRITY ausführen. Setzen Sie das Sonderregister CURRENT EXPLAIN MODE anschließend wieder auf NO.

Einschränkungen:

- Die Anweisungen REFRESH TABLE und SET INTEGRITY kommen für die Reoptimierung nicht in Frage. Daher ist der EXPLAIN-Modus REOPT (oder EXPLAIN SNAPSHOT) auf diese beiden Anweisungen nicht anwendbar.
- Die Klausel WITH REOPT ONCE der Anweisung EXPLAIN, die ebenfalls angibt, dass die angegebene mit EXPLAIN zu bearbeitende Anweisung reoptimiert werden soll, ist auf die Anweisungen REFRESH TABLE und SET INTEGRITY nicht anwendbar.

Szenario

Dieses Szenario veranschaulicht, wie Sie einen Zugriffsplan aus EXPLAIN- und REFRESH TABLE-Anweisungen generieren und zur Selbstdiagnose der Ursache Ihres Leistungsproblems verwenden können.

Schritt 1

Erstellen Sie Ihre Tabellen, und füllen Sie sie mit Daten. Beispiel:

```
CREATE TABLE T
    (i1 INT NOT NULL,
     i2 INT NOT NULL,
     PRIMARY KEY (i1));
INSERT INTO T VALUES (1,1), (2,1), (3,2), (4,2);
CREATE TABLE MQT AS (SELECT i2, COUNT(*) AS CNT FROM T GROUP BY i2)
DATA INITIALLY DEFERRED
REFRESH DEFERRED;
```

Schritt 2

Setzen Sie die Anweisungen EXPLAIN und REFRESH TABLE wie folgt ab:

```
EXPLAIN PLAN FOR REFRESH TABLE MQT;
```

Anmerkung: Alternativ zu diesem Schritt kann der EXPLAIN-Modus im Sonderregister SET CURRENT EXPLAIN MODE wie folgt angegeben werden:

```
SET CURRENT EXPLAIN MODE EXPLAIN;
REFRESH TABLE MQT;
SET CURRENT EXPLAIN MODE NO;
```

Schritt 3

Formatieren Sie den Inhalt der EXPLAIN-Tabellen mit dem Befehl db2exfmt, und rufen Sie den Zugriffsplan ab. Dieses Tool befindet sich im Unterverzeichnis misc des Verzeichnisses sqllib für Ihre Instanz.

```
db2exfmt –d <datenbankname> -o refresh.exp -1
```

Schritt 4

Analysieren Sie den Zugriffsplan, um die Ursache des Leistungsproblems festzustellen. Durch eine Analyse des Plans aus den oben gezeigten Anweisungen ließe sich zum Beispiel ermitteln, dass die Ausführung einer Tabellensuche (TABLE SCAN) für Tabelle T, falls diese sehr groß ist, sehr aufwendig wäre. In diesem Fall könnte die Leistung der Abfrage durch die Erstellung eines Index verbessert werden.

EXPLAIN-Tools

DB2 stellt eine umfassende EXPLAIN-Einrichtung bereit, die ausführliche Informationen über den Zugriffsplan bereitstellt, den das Optimierungsprogramm für eine SQL- oder XQuery-Anweisung auswählt. Der Zugriff auf die Tabellen, in denen EXPLAIN-Daten, d. h. Informationen zu statischen und dynamischen SQL- oder XQuery-Anweisungen, gespeichert werden, ist auf allen unterstützten Plattformen möglich. Verschiedene Tools oder Methoden bieten flexible Möglichkeiten, die EXPLAIN-Informationen zu erfassen, anzuzeigen und zu analysieren.

Detaillierte Informationen des Optimierungsprogramms, die eine eingehende Analyse eines Zugriffsplans ermöglichen, werden getrennt vom eigentlichen Zugriffsplan in EXPLAIN-Tabellen gespeichert. Es stehen mehrere Methoden zum Abrufen von Informationen aus den EXPLAIN-Tabellen zur Verfügung:

- Visual Explain zum Anzeigen von EXPLAIN-Momentaufnahmen

  Rufen Sie Visual Explain über die Steuerzentrale auf, um eine grafische Darstellung eines Abfragezugriffsplans anzuzeigen. Sie können sowohl statische als auch dynamische SQL- oder XQuery-Anweisungen analysieren.

  Visual Explain ermöglicht Ihnen, auf einer anderen Plattform erfasste oder erstellte Momentaufnahmen anzuzeigen. Zum Beispiel kann ein Windows-Client Momentaufnahmen grafisch darstellen, die auf einem Server unter DB2 für HP-UX generiert wurden.
- Das Tool db2exfmt zum Anzeigen von EXPLAIN-Informationen in einer vorformatierten Ausgabe
- Die Tools db2expln und dynexpln

  Zugriffsplaninformationen, die für ein oder mehrere Pakete mit statischen SQL- oder XQuery-Anweisungen verfügbar sind, können mit dem Tool db2expln über die Befehlszeile angezeigt werden. Das Tool db2expln zeigt die tatsächliche Implementierung des gewählten Zugriffsplans. Informationen des Optimierungsprogramms werden nicht angezeigt.

  Das Tool dynexpln, das intern auf db2expln zurückgreift, stellt ein zeiteffizientes Verfahren zur Bearbeitung dynamischer SQL- oder XQuery-Anweisungen mit EXPLAIN dar, die keine Parametermarken enthalten. Die Verwendung von db2expln innerhalb des Tools dynexpln wird durch die Umsetzung der SQL- oder XQuery-Eingabeanweisung in eine statische Anweisung innerhalb eines Pseudopakets ermöglicht. Wenn dieses Verfahren angewandt wird, sind die Informationen nicht immer ganz exakt. Wenn es auf Genauigkeit ankommt, verwenden Sie die EXPLAIN-Einrichtung.

  Das Tool db2expln liefert einen relativ kompakten und englisch aufbereiteten Überblick über die Operationen, die bei der Ausführung stattfinden, indem der tatsächlich generierte Zugriffsplan analysiert wird.
- Schreiben eigener Abfragen für die EXPLAIN-Tabellen

  Durch Schreiben eigener Abfragen können Sie die Ausgabe leicht bearbeiten und verschiedene Abfragen oder auch die gleiche Abfrage über einen Zeitraum hinweg vergleichen.

Anmerkung: Die EXPLAIN-Tools und andere Tools, die über die Befehlszeile ausgeführt werden, wie db2batch, dynexpln und db2_all, befinden sich im Unterverzeichnis misc des Verzeichnisses sqllib. Wenn die Tools aus diesem Pfad an eine andere Position versetzt werden, funktionieren die Befehlszeilenmethoden möglicherweise nicht.

Die folgende Tabelle enthält eine Übersicht über die verschiedenen, in der EXPLAIN-Einrichtung von DB2 verfügbaren Tools und ihre Merkmale. Verwenden Sie die Tabelle zur Auswahl des Tools, das für Ihre Anforderungen und Ihre Umgebung am besten geeignet ist.

Tabelle 65. Tools der EXPLAIN-Einrichtung

Merkmale	Visual Explain	EXPLAIN-Tabellen	db2exfmt	db2expln	dynexpln
Grafische Benutzerschnittstelle	Ja				
Textausgabe			Ja	Ja	Ja
Schnelle Grobanalyse für statisches SQL und XQuery				Ja	
Unterstützung für statisches SQL und XQuery	Ja	Ja	Ja	Ja	
Unterstützung für dynamisches SQL und XQuery	Ja	Ja	Ja	Ja	Ja*
Unterstützung für CLI-Anwendungen	Ja	Ja	Ja		
Verfügbar für DRDA-Anwendungsrequester		Ja			
Detaillierte Informationen des Optimierungsprogramms	Ja	Ja	Ja		
Geeignet zur Analyse mehrerer Anweisungen		Ja	Ja	Ja	Ja
Zugriff auf die Informationen aus einer Anwendung heraus		Ja			

Anmerkung:

* Verwendet indirekt db2expln. Es gelten einige Einschränkungen.

Anzeigen der bei der EXPAIN-Bearbeitung wirksamen Katalogstatistiken

Die EXPLAIN-Einrichtung erfasst die Statistikdaten die zu dem Zeitpunkt wirksam sind, zu dem eine Anweisung mit EXPLAIN bearbeitet wird. Diese Statistiken können sich von denen unterscheiden, die in den Systemkatalogen gespeichert sind, insbesondere wenn die Echtzeitstatistikerfassung aktiviert ist. Wenn die EXPLAIN-Tabellen mit Daten gefüllt wurden, jedoch keine EXPLAIN-Momentaufnahme erstellt wird, sind nur einige Statistikdaten in der Tabelle EXPLAIN_OBJECT aufgezeichnet.

Zur Erfassung aller Katalogstatistiken, die für die Anweisung, die mit EXPLAIN bearbeitet wird, relevant sind, erstellen Sie eine EXPLAIN-Momentaufnahme zur gleichen Zeit, zu der die EXPLAIN-Tabellen mit Daten gefüllt werden. Verwenden Sie anschließend die Funktion SYSPROC.EXPLAIN_FORMAT_STATS, um die Katalogtabellen in der Momentaufnahme zu formatieren.

Wenn das Tool db2exfmt zur Formatierung der EXPLAIN-Informationen verwendet wird, verwendet es automatisch die Funktion SYSPROC.EXPLAIN_FORMAT_STATS, um die Katalogstatistiken anzuzeigen, wenn eine Momentaufnahme erfasst wurde. Visual Explain zeigt automatisch alle Statistiken an, die in der Momentaufnahme enthalten sind.

SQL- und XQuery-EXPLAIN-Tools

Das Tool db2expln beschreibt den Zugriffsplan, der für SQL- und XQuery-Anweisungen ausgewählt wird. Mit diesem Tool kann eine schnelle Erläuterung des ausgewählten Zugriffsplans abgerufen werden, wenn keine EXPLAIN-Daten erfasst wurden. Für statische SQL- und XQuery-Anweisungen überprüft db2expln die Pakete, die in den Systemkatalogtabellen gespeichert sind. Für dynamische SQL- und XQuery-Anweisungen überprüft db2expln die Abschnitte im Abfragecache.

Das Tool dynexpln kann ebenfalls zur Beschreibung des für dynamische Anweisungen ausgewählten Zugriffsplans verwendet werden. Es erstellt ein statisches Paket für die Anweisungen und verwendet dann das Tool db2expln, um sie zu beschreiben. Da dynamische SQL- und XQuery-Anweisungen jedoch auch mit db2expln überprüft werden können, wird dieses Dienstprogramm nur aus Gründen der Abwärtskompatibilität beibehalten.

Die EXPLAIN-Tools (db2expln und dynexpln) befinden sich im Unterverzeichnis bin im Verzeichnis sqllib Ihrer Instanz. Wenn sich db2expln und dynexpln nicht in Ihrem aktuellen Verzeichnis befinden, müssen sie sich in einem Verzeichnis befinden, das in Ihrer Umgebungsvariablen PATH definiert ist.

Das Programm db2expln stellt eine Verbindung zu einer Datenbank her und bindet sich mithilfe der Dateien db2expln.bnd, db2exsrv.bnd und db2exdyn.bnd selbst an diese Datenbank, wenn auf die Datenbank zum ersten Mal zugegriffen wird.

Zum Ausführen von db2expln müssen Sie über das Zugriffsrecht SELECT für die Systemkatalogsichten sowie über das Zugriffsrecht EXECUTE für die Pakete db2expln, db2exsrv und db2exdyn verfügen. Zum Ausführen von dynexpln müssen Sie über die Berechtigung BINDADD für die Datenbank verfügen, und das SQL-Schema, das Sie zur Herstellung einer Verbindung zur Datenbank verwenden, muss vorhanden sein, oder Sie müssen über die Berechtigung IMPLICIT_SCHEMA für die Datenbank verfügen. Zur EXPLAIN-Bearbeitung dynamischer SQL- und XQuery-Anweisungen mit db2expln oder dynexpln müssen Sie außerdem über alle Zugriffsrechte verfügen, die für die mit EXPLAIN bearbeiteten Abfrageanweisungen erforderlich sind. (Wenn Sie über SYSADM- oder DBADM-Berechtigung verfügen, haben Sie automatisch alle erforderlichen Berechtigungsstufen.)

dynexpln

Das Tool dynexpln ist aus Gründen der Abwärtskompatibilität weiterhin verfügbar. Sie können jedoch die **dynamischen-optionen** von db2expln verwenden, um alle Funktionen von dynexpln auszuführen.

Wenn Sie die dynamischen Optionen von db2expln verwenden, wird die Anweisung als wahre dynamische SQL- und XQuery-Anweisung vorbereitet (PREPARE), und der generierte Plan wird aus dem Abfragecache heraus mit EXPLAIN bearbeitet. Diese EXPLAIN-Ausgabemethode stellt genauere Zugriffspläne als das Tool dynexpln bereit, das die Anweisung als statische SQL- und XQuery-Anweisung vorbereitet. Sie ermöglicht darüber hinaus die Verwendung von Funktionen, die nur in dynamischen SQL- und XQuery-Anweisungen verfügbar sind, wie zum Beispiel Parametermarken.

Hinweise zur Verwendung von dynexpln: Zur Bearbeitung dynamischer Anweisungen erstellt dynexpln eine statische Anwendung für die Anweisungen und ruft anschließend db2expln auf. Zur Erstellung der statischen Anweisungen generiert dynexpln ein triviales C-Programm mit den Anweisungen und ruft anschließend den DB2-Precompiler zur Erstellung des Pakets auf. (Das generierte C-Programm

ist nicht vollständig und kann nicht kompiliert werden. Es enthält nur so viele Informationen, dass der Precompiler das Paket erstellen kann.)

Es folgen einige allgemeine Nachrichten, die dynexpln ausgeben kann:

- Alle Fehlernachrichten aus db2expln

  Da dynexpln das Programm db2expln aufruft, können die Mehrzahl der Fehlernachrichten von db2expln in der Ausgabe auftreten.

- `Error connecting to the database.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn ein Fehler bei der Herstellung der Verbindung zur Datenbank aufgetreten ist. In einer weiteren CLI-Fehlernachricht wird die Ursache für das Fehlschlagen der Verbindung angezeigt. Beheben Sie die Fehlerursache, und führen Sie dynexpln erneut aus.

- `The file "<dateiname>" must be removed before dynexpln will run.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn die angegebene Datei zur Zeit der Ausführung von dynexpln existiert. Entfernen Sie die Datei, oder ändern Sie den Wert für die Umgebungsvariable DYNEXPLN_PACKAGE, um den Namen der Datei zu ändern, die erstellt wird, und führen Sie dynexpln erneut aus.

- `The package "<ersteller>.<paket>" must be dropped before dynexpln will run.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn das angegebene Paket zur Zeit der Ausführung von dynexpln existiert. Löschen Sie das Paket, oder ändern Sie den Wert für die Umgebungsvariable **DYNEXPLN_PACKAGE**, um den Namen des Pakets zu ändern, das erstellt wird, und führen Sie dynexpln erneut aus.

- `Error writing file "<dateiname>".`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn in die Datei nicht geschrieben werden kann. Stellen Sie sicher, dass dynexpln in die Dateien des aktuellen Verzeichnisses schreiben kann, und führen Sie dynexpln erneut aus.

- `Error reading input file "<dateiname>".`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn die mit der Option **-f** angegebene Datei nicht gelesen werden kann. Stellen Sie sicher, dass die Datei vorhanden ist und dass dynexpln sie lesen kann. Führen Sie anschließend dynexpln erneut aus.

Umgebungsvariablen: Es gibt zwei verschiedene Umgebungsvariablen, die in Verbindung mit dynexpln verwendet werden können:

- **DYNEXPLN_OPTIONS** definiert die Optionen für den SQL- und XQuery-Precompiler, die Sie bei der Erstellung des Pakets für Ihre Anweisungen verwenden. Verwenden Sie dieselbe Syntaxvariable wie bei der Eingabe eines Befehls PREP über CLP.

  Beispiel: `DYNEXPLN_OPTIONS="OPTLEVEL 5 BLOCKING ALL"`

- **DYNEXPLN_PACKAGE** definiert den Namen des Pakets, das in der Datenbank erstellt wird. Die mit EXPLAIN zu bearbeitenden Anweisungen werden in dieses Paket geschrieben. Falls diese Variable nicht definiert ist, erhält das Paket den Standardnamen **DYNEXPLN**. (Es werden nur die ersten acht Zeichen des Namens in dieser Umgebungsvariablen verwendet.)

  Der Name wird auch zur Generierung von Namen für von dynexpln benötigte temporäre Dateien verwendet.

Beschreibung der Ausgabe von db2expln und dynexpln

In der Ausgabe werden die EXPLAIN-Informationen für jedes Paket in den beiden folgenden Teilen dargestellt:

- Paketinformationen wie das Datum des Bindevorgangs und relevante Bindeoptionen
- Abschnittsinformationen, wie z. B. die Abschnittsnummer, gefolgt von der mit EXPLAIN bearbeiteten SQL- oder XQuery-Anweisung. Unter den Abschnittsinformationen befindet sich die EXPLAIN-Ausgabe des für die SQL- oder XQuery-Anweisung ausgewählten Zugriffsplans.

Die Schritte eines Zugriffsplans oder Abschnitts werden in der Reihenfolge aufgeführt, in der sie vom Datenbankmanager ausgeführt werden. Jeder Hauptschritt wird als linksbündig ausgerichtete Überschrift angezeigt. Informationen zum Schritt werden darunter eingerückt angezeigt. Am linken Rand der EXPLAIN-Ausgabe für den Zugriffsplan befinden sich Einrückungsbalken. Diese Balken markieren gleichzeitig den Geltungsbereich der Operation. Operationen niedrigerer Einrückungsstufe in derselben Operation, das heißt solche, die weiter rechts angezeigt werden, werden verarbeitet, bevor zur vorherigen Einrückungsstufe zurückgekehrt wird.

Beachten Sie, dass der ausgewählte Zugriffsplan auf einer erweiterten Version der ursprünglichen SQL- oder XQuery-Anweisung basiert, die in der Ausgabe gezeigt wird. Beispielsweise kann die ursprüngliche Anweisung die Aktivierung von Triggern und Integritätsbedingungen bewirken. Darüber hinaus kann die Komponente des Abfragecompilers zum Umschreiben von Abfragen die SQL- oder XQuery-Anweisung in ein äquivalentes, jedoch effizienteres Format umschreiben. Alle diese Faktoren sind in den Informationen enthalten, die das Optimierungsprogramm verwendet, wenn es den effizientesten Plan zum Ausführen der Anweisung ermittelt. Daher kann sich der in der EXPLAIN-Ausgabe angezeigte Zugriffsplan erheblich von dem unterscheiden, den Sie möglicherweise für die ursprüngliche SQL- oder XQuery-Anweisung erwarten. Die EXPLAIN-Einrichtung, zu der die EXPLAIN-Tabellen, der Modus SET CURRENT EXPLAIN und Visual Explain gehören, zeigt die tatsächlich für die Optimierung verwendete SQL- oder XQuery-Anweisung in Form einer SQL- oder XQuery-ähnlichen Anweisung, die durch Rückübersetzung der internen Darstellung der Abfrage erstellt wird.

Beim Vergleichen der Ausgabe von `db2expln` oder `dynexpln` mit der Ausgabe der EXPLAIN-Einrichtung kann die Option für die Operator-ID (**-opids**) sehr nützlich sein. Jedesmal, wenn `db2expln` bzw. `dynexpln` die Verarbeitung eines neuen Operators aus der EXPLAIN-Einrichtung beginnt, wird die Operator-ID links neben dem von EXPLAIN gezeigten Plan ausgegeben. Die Operator-ID kann zum Abgleichen der Schritte in den verschiedenen Darstellungen des Zugriffsplans verwendet werden. Beachten Sie, dass es nicht immer eine Eins-zu-eins-Entsprechung gibt zwischen den Operatoren in der Ausgabe der EXPLAIN-Einrichtung und den Operationen, die in der Ausgabe von `db2expln` und `dynexpln` angezeigt werden.

Tabellenzugriffsinformationen: Diese Angabe zeigt den Namen und den Typ der Tabelle an, auf die zugegriffen wird. Es werden zwei Formate verwendet:

1. Drei Typen von regulären Tabellen:
   - Zugriff auf Tabellenname (Access Table Name):

     ```
     Access Table Name = schema.name  ID = ts,n
     ```

     Dabei gilt Folgendes:
     - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
     - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.

- Zugriff auf Hierarchietabellenname (Access Hierarchy Table Name):

  ```
  Access Hierarchy Table Name = schema.name  ID = ts,n
  ```

  Dabei gilt Folgendes:
  - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
  - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.
- Zugriff auf den Namen einer MQT (Materialized Query Table, gespeicherten Abfragetabelle) (Access Materialized Query Table Name):

  ```
  Access Materialized Query Table Name = schema.name  ID = ts,n
  ```

  Dabei gilt Folgendes:
  - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
  - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.

2. Zwei Typen von temporären Tabellen:
   - Zugriff auf ID für temporäre Tabellen (Access Temporary Table ID):

     ```
     Access Temp Table  ID = tn
     ```

     Dabei gilt Folgendes:
     - *ID* ist die entsprechende Kennung, die von `db2expln` zugeordnet wurde.
   - Zugriff auf ID für deklarierte temporäre Tabellen (Access Declared Global Temporary Table ID):

     ```
     Access Global Temp Table  ID = ts,tn
     ```

     Dabei gilt Folgendes:
     - *ID* ist die entsprechende TABLESPACEID aus dem Katalog SYSCAT.TABLES für die Tabelle (`ts`) und die entsprechende von `db2expln` zugeordnete Kennung (`tn`)

Nach der Angabe über den Zugriff auf die Tabelle folgen weitere Angaben, die den Zugriff weiter beschreiben. Diese Angaben werden unter der Angabe über den Zugriff auf die Tabelle eingerückt. Folgende Angaben sind möglich:

- Anzahl von Spalten
- Blockzugriff
- Parallelsuche
- Suchrichtung
- Zeilenzugriffsmethode
- Sperrabsichten
- Vergleichselemente
- Verschiedene Angaben

Anzahl von Spalten

Die folgende Angabe zeigt die Anzahl von Spalten an, die aus jeder Zeile der Tabelle verwendet werden:

```
#Columns = n
```

Blockzugriff

Die folgende Angabe zeigt an, dass für die Tabelle ein oder mehrere Dimensionsblockindizes definiert sind:

```
Clustered by Dimension for Block Index Access
```

Wenn dieser Text nicht angezeigt wird, wurde die Tabelle ohne die Klausel DIMENSION erstellt.

Parallelsuche

Die folgende Angabe zeigt an, dass der Datenbankmanager mehrere Subagenten zum parallelen Lesen aus der Tabelle verwendet:

```
Parallel Scan
```

Wenn diese Angabe fehlt, wird die Tabelle nur von einem Agenten (bzw. Subagenten) gelesen.

Suchrichtung

Die folgende Angabe zeigt an, dass der Datenbankmanager Zeilen in umgekehrter Reihenfolge liest:

```
Scan Direction = Reverse
```

Wenn dieser Text nicht angezeigt wird, wird in Vorwärtsrichtung gesucht (dies ist der Standardwert).

Zeilenzugriffsmethode

Eine der folgenden Angaben zur Art und Weise des Zugriffs auf die Zeilen in der Tabelle, die den angegebenen Bedingungen entsprechen, wird angezeigt:

- Die Angabe `Relation Scan` bedeutet, dass die Tabelle sequenziell durchsucht wird, um die den angegebenen Bedingungen entsprechenden Zeilen zu ermitteln.
  - Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf Daten erfolgt:

    ```
    Relation Scan
    |  Prefetch: None
    ```

  - Die folgende Angabe bedeutet, dass das Optimierungsprogramm die Anzahl der Seiten, die vorabgelesen werden, vorbestimmt hat:

    ```
    Relation Scan
    |  Prefetch: n Pages
    ```

  - Die folgende Angabe weist darauf hin, dass die Daten wahrscheinlich vorabgelesen werden:

    ```
    Relation Scan
    |  Prefetch: Eligible
    ```

  - Die folgende Angabe bedeutet, dass die Identifizierung der Zeilen, die den angegebenen Kriterien entsprechen, und der Zugriff auf sie über einen Index erfolgen:

    ```
    Index Scan:  Name = schema.name  ID = xx
    | Index type
    | Index Columns:
    ```

    Dabei gilt Folgendes:

- *schema.name* ist der vollständig qualifizierte Name des Index, der durchsucht wird.
- *ID* ist die entsprechende Spalte IID in der Katalogsicht SYSCAT.INDEXES.
- Der Indextyp ist einer der folgenden:

  ```
  Regular Index (Not Clustered)
  Regular Index (Clustered)
  Dimension Block Index
  Composite Dimension Block Index
  Index over XML data
  ```

  Diesen Angaben folgen jeweils eine Zeile für jede Spalte im Index. Jede Spalte im Index wird in einer der folgenden Formen aufgeführt:

  ```
  n: spaltenname (Ascending)
  n: spaltenname (Descending)
  n: spaltenname (Include Column)
  ```

Die folgenden Angaben präzisieren die Art der Indexsuche:

- Die bereichsbegrenzenden Vergleichselemente für den Index werden folgendermaßen angegeben:

  ```
  #Key Columns = n
  |  Start Key: xxxxx
  |  Stop Key: xxxxx
  ```

  Dabei ist xxxxx eine der folgenden Angaben:

  - `Start of Index`
  - `End of Index`
  - `Inclusive Value:` oder `Exclusive  Value:`

    Ein inklusiver Schlüsselwert wird in die Indexsuche mit einbezogen. Ein exklusiver Schlüsselwert wird in der Suchoperation nicht berücksichtigt. Der Wert für den Schlüssel wird durch eine der folgenden Zeilen für jeden Teil des Schlüssels angegeben:

    ```
    n: 'zeichenfolge'
    n: nnn
    n: jjjj-mm-tt
    n: ss:mm:ss
    n: jjjj-mm-tt ss:mm:ss.uuuuuu
    n: NULL
    n: ?
    ```

    Wenn eine Literalzeichenfolge angezeigt wird, werden nur die ersten 20 Zeichen angegeben. Ist die Zeichenfolge länger als 20 Zeichen, wird dieses durch `...` am Ende der Zeichenfolge angezeigt. Einige Schlüssel können erst bestimmt werden, wenn der Abschnitt (section) ausgeführt wird. Dies wird durch den Wert `?` angezeigt.

- `Reiner Indexzugriff`

  Wenn alle benötigten Spalten aus dem Indexschlüssel abgerufen werden können, wird diese Angabe für reinen Indexzugriff angezeigt, und es wird nicht auf Tabellendaten zugegriffen.

- Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf die Indexseiten erfolgt:

  ```
  Index Prefetch: None
  ```

- Die folgende Angabe bedeutet, dass Indexseiten wahrscheinlich vorabgelesen werden:

  ```
  Index Prefetch: Eligible
  ```

- Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf Datenseiten erfolgt:

```
Data Prefetch: None
```

- Die folgende Angabe weist darauf hin, dass Datenseiten wahrscheinlich vorabgelesen werden:

```
Data Prefetch: Eligible
```

- Wenn Vergleichselemente existieren, die an den Indexmanager übermittelt werden können, um bei der Qualifizierung von Indexeinträgen zu helfen, zeigt die folgende Angabe die Anzahl der Vergleichselemente:

```
Sargable Index Predicate(s)
|  #Predicates = n
```

– Wenn auf die den Bedingung entsprechenden Zeilen über die Zeilen-IDs (RIDs) zugegriffen wird, die zuvor im Zugriffsplan vorbereitet wurden, wird dies durch die folgende Angabe gezeigt:

```
Fetch Direct Using Row IDs
```

Wenn für die Tabelle ein oder mehrere Blockindizes definiert sind, kann der Zugriff auf die Zeilen entweder über Block-IDs oder über Zeilen-IDs erfolgen. Dies wird wie folgt angegeben:

```
Fetch Direct Using Block or Row IOs
```

Sperrabsicht

Für jeden Tabellenzugriff wird die Art der Sperre, die auf Tabellen- und Zeilenebene aktiviert werden soll, mit der folgenden Angabe angezeigt:

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

Folgende Werte sind für eine Tabellensperre möglich:
- Exclusive
- Intent Exclusive
- Intent None
- Intent Share
- Share
- Share Intent Exclusive
- Super Exclusive
- Update

Folgende Werte sind für eine Zeilensperre möglich:
- Exclusive
- Next Key Exclusive (wird nicht in der Ausgabe von `db2expln` angezeigt)
- None
- Share
- Next Key Share
- Update
- Next Key Weak Exclusive
- Weak Exclusive

Vergleichselemente

Es gibt zwei Angaben, die Informationen über die in einem Zugriffsplan verwendeten Vergleichselemente enthalten:

1. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die für jeden Datenblock ausgewertet werden, der aus einem Blockindex abgerufen wird.

```
Block Predicates(s)
|  #Predicates = n
```

2. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die ausgewertet werden, während auf die Daten zugegriffen wird. Die Anzahl der Vergleichselemente lässt verschobene (Pushdown) Operationen wie Spaltenberechnung oder Sortierung unberücksichtigt.

```
Sargable Predicate(s)
|  #Predicates = n
```

3. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die ausgewertet werden, nachdem die Daten zurückgegeben wurden (d. h. Restvergleichselemente):

```
Residual Predicate(s)
|  #Predicates = n
```

Die Anzahl der in den oben stehenden Anweisungen angezeigten Vergleichselemente spiegelt möglicherweise die Anzahl der Vergleichselemente der Abfrageanweisung aus folgenden Gründen nicht genau wider:

- Vergleichselemente können mehrmals in derselben Abfrage verwendet werden.
- Vergleichselemente können durch Hinzufügung impliziter Vergleichselemente während der Optimierung der Abfrage umgewandelt und erweitert worden sein.
- Vergleichselemente können während der Optimierung der Abfrage in weniger Vergleichselemente umgewandelt und komprimiert worden sein.

Verschiedene Tabellenangaben

- Die folgende Angabe weist darauf hin, dass nur auf eine Zeile zugegriffen wird:

  ```
  Single Record
  ```

- Die folgende Angabe wird angezeigt, wenn die für diesen Tabellenzugriff verwendete Isolationsstufe nicht der Isolationsstufe der Anweisung entspricht:

  ```
  Isolation Level: xxxx
  ```

  Eine andere Isolationsstufe kann aus einer Reihe von Gründen verwendet werden, zum Beispiel:

  – Ein Paket wurde mit wiederholtem Lesen (RR) gebunden und hat Auswirkungen auf referenzielle Integritätsbedingungen. Der Zugriff der übergeordneten Tabelle zum Prüfen referenzieller Integritätsbedingungen wird auf die Isolationsstufe Cursorstabilität (CS) heruntergestuft, um unnötige Sperren für diese Tabelle zu vermeiden.

  – Ein mit der Isolationsstufe für nicht festgeschriebenen Lesevorgang (UR) gebundenes Paket gibt eine Anweisung DELETE oder UPDATE aus. Der Tabellenzugriff für den tatsächlichen Löschvorgang wird auf Cursorstabilität (CS) hochgestuft.

- Die folgende Angabe weist darauf hin, dass einige oder alle Zeilen, die aus der temporären Tabelle gelesen wurden, außerhalb des Pufferpools zwischengespeichert werden, wenn genügend Sortierspeicher verfügbar ist:

  ```
  Keep Rows In Private Memory
  ```

- Wenn für die Tabelle das Attribut für flüchtige Kardinalität definiert wurde, wird dies folgendermaßen angezeigt:

  ```
  Volatile Cardinality
  ```

Informationen zu temporären Tabellen: Eine temporäre Tabelle wird von einem Zugriffsplan während dessen Ausführung zum Speichern von Daten in einer temporären Arbeitstabelle oder Übergangstabelle verwendet. Die Tabelle existiert nur, solange der Zugriffsplan ausgeführt wird. Allgemein werden temporäre Tabellen

verwendet, wenn Unterabfragen frühzeitig im Zugriffsplan ausgewertet werden müssen oder wenn Zwischenergebnisse nicht in den vorhandenen Speicher passen.

Wenn eine temporäre Tabelle erstellt werden muss, kann eine von zwei möglichen Angaben angezeigt werden. Diese Angaben weisen darauf hin, dass eine temporäre Tabelle erstellt und Zeilen in sie eingefügt werden. Die ID ist eine Kennung, die aus praktischen Gründen von `db2expln` zugeordnet wird, wenn auf die temporäre Tabelle Bezug genommen wird. Dieser ID wird als Präfix der Buchstabe 't' vorangestellt, um anzuzeigen, dass es sich um eine temporäre Tabelle handelt.

- Die folgende Angabe bedeutet, dass eine normale temporäre Tabelle erstellt wird:
  ```
  Insert Into Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine normale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Shared Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte temporäre Tabelle erstellt wird:
  ```
  Insert Into Sorted Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Sorted Shared Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine deklarierte globale temporäre Tabelle erstellt wird:
  ```
  Insert Into Global Temp Table  ID = ts,tn
  ```
- Die folgende Angabe bedeutet, dass eine deklarierte globale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Shared Global Temp Table  ID = ts,tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte deklarierte globale temporäre Tabelle erstellt wird:
  ```
  Insert Into Sorted Global Temp Table  ID = ts,tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte deklarierte globale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Sorted Shared Global Temp Table  ID = ts,tn
  ```

Nach jeder der oben gezeigten Angaben folgt die Angabe:

```
#Columns = n
```

Diese Angabe zeigt die Anzahl der Spalten für jede Zeile, die in die temporäre Tabelle eingefügt wird.

Sortierte temporäre Tabellen

Sortierte temporäre Tabellen treten z. B. als Ergebnis folgender Operationen auf:

- ORDER BY
- DISTINCT
- GROUP BY
- Mischjoin (Merge Join)
- Unterabfrage '= ANY'
- Unterabfrage '<> ALL'
- INTERSECT oder EXCEPT
- UNION (ohne das Schlüsselwort ALL)

Eine Reihe zusätzlicher Angaben kann auf die ursprüngliche Angabe über die Erstellung für eine sortierte temporäre Tabelle folgen:

- Die folgende Angabe zeigt die Anzahl der bei der Sortierung verwendeten Schlüsselspalten:

  ```
  #Sort Key Columns = n
  ```

  Für jede Spalte im Sortierschlüssel wird eine der folgenden Zeilen angezeigt:

  ```
  Key n: spaltenname (Ascending)
  Key n: spaltenname (Descending)
  Key n: (Ascending)
  Key n: (Descending)
  ```

- Die folgenden Angaben enthalten Schätzwerte für die Anzahl und die Größe der Zeilen, sodass die optimale Größe für den Sortierspeicher zur Laufzeit zugeordnet werden kann.

  ```
  Sortheap Allocation Parameters:
  #Rows     = n
  Row Width = n
  ```

- Wenn lediglich die ersten Zeilen des sortierten Ergebnisses benötigt werden, wird Folgendes angezeigt:

  ```
  Sort Limited To Estimated Row Count
  ```

- Für Sortiervorgänge in einer symmetrischen Mehrprozessorumgebung (SMP-Umgebung) wird die Art der durchzuführenden Sortierung durch eine der folgenden Angaben angezeigt:

  ```
  Use Partitioned Sort
  Use Shared Sort
  Use Replicated Sort
  Use Round-Robin Sort
  ```

- Die folgenden Angaben zeigen an, ob das Ergebnis der Sortierung im Sortierspeicher verbleibt:

  ```
  Piped
  ```

  und

  ```
  Not Piped
  ```

  Wenn eine über Pipe geleitete Sortierung angegeben wird, behält der Datenbankmanager die sortierte Ausgabe im Speicher, anstatt das sortierte Ergebnis in eine andere temporäre Tabelle zu schreiben.

- Die folgende Angabe bedeutet, dass doppelte Werte während der Sortierung entfernt werden:

  ```
  Duplicate Elimination
  ```

- Wenn Spaltenberechnungen in der Sortierung durchgeführt werden, wird dies durch eine der folgenden Angaben angezeigt:

  ```
  Partial Aggregation
  Intermediate Aggregation
  Buffered Partial Aggregation
  Buffered Intermediate Aggregation
  ```

Abschließen der temporären Tabelle

Nach einem Tabellenzugriff, der eine an eine Datenquelle verschobene (Pushdown) Operation zum Erstellen einer temporären Tabelle enthält (d. h. eine Operation zum Erstellen einer temporären Tabelle, die im Rahmen eines Tabellenzugriffs auftritt), folgt eine Abschlussangabe (Completion), die das Dateiende verarbeitet,

indem sie die temporäre Tabelle darauf vorbereitet, Zeilen für den nachfolgenden Zugriff auf die temporäre Tabelle zur Verfügung zu stellen. Es wird eine der folgenden Zeilen angezeigt:

```
Temp Table Completion  ID = tn
Shared Temp Table Completion  ID = tn
Sorted Temp Table Completion  ID = tn
Sorted Shared Temp Table Completion  ID = tn
```

Tabellenfunktionen

Tabellenfunktionen sind benutzerdefinierte Funktionen (UDFs), die Daten in Form einer Tabelle an die Anweisung zurückgeben. Tabellenfunktionen werden durch folgende Angaben angezeigt:

```
Access User Defined Table Function
  Name = schema.funktionsname
  Specific Name = spezifischer_name
  SQL Access Level = zugriffsebene
  Language = sprache
  Parameter Style = parmstil
  Fenced                      Not Deterministic
  Called on NULL Input        Disallow Parallel
  Not Federated               Not Threadsafe
```

Der spezifische Name identifiziert die aufgerufene Tabellenfunktion eindeutig. Die übrigen Zeilen geben Details über die Attribute der Funktion an.

Joininformationen: Es gibt drei Arten von Joins:

- Hash-Join
- Mischjoin (Merge Join)
- Join mit Verschachtelungsschleife (Nested Loop Join)

Wenn bei der Ausführung eines Abschnitts der Zeitpunkt kommt, an dem ein Join ausgeführt wird, wird eine der folgenden Angaben angezeigt:

```
Hash Join
Merge Join
Nested Loop Join
```

Es ist möglich, dass ein linker Outer Join durchgeführt wird. Ein linker Outer Join wird durch eine der folgenden Angaben angezeigt:

```
Left Outer Hash Join
Left Outer Merge Join
Left Outer Nested Loop Join
```

Bei Mischjoins und Joins mit Verschachtelungsschleife ist die äußere Tabelle des Joins die Tabelle, auf die in der vorherigen Zugriffsangabe, die in der Ausgabe angezeigt wird, verwiesen wird. Die innere Tabelle bei diesem Join ist die Tabelle, auf die in der Zugriffsangabe verwiesen wird, die sich im Bereich der Joinangabe befindet. Bei Hash-Joins sind die Zugriffsangaben umgekehrt, d. h. die äußere Tabelle ist im Joinbereich enthalten, und die innere Tabelle wird vor dem Join angezeigt.

Bei einem Hash- oder Mischjoin können folgende weitere Angaben auftreten:

- In einigen Fällen muss bei einem Join lediglich festgestellt werden, ob eine Zeile der inneren Tabelle mit der aktuellen Zeile in der äußeren Tabelle übereinstimmt. Dies wird durch folgende Angabe angezeigt:

  ```
  Early Out: Single Match Per Outer Row
  ```

- Es ist möglich, nach Abschluss des Joins Vergleichselemente anzuwenden. Die Anzahl der Vergleichselemente, die angewendet werden, wird folgendermaßen angezeigt:

```
Residual Predicate(s)
|  #Predicates = n
```

Bei einem Hash-Join können folgende weitere Angaben auftreten:

- Die Hashtabelle wird aus der inneren Tabelle erstellt. Wenn die Erstellung der Hashtabelle in ein Vergleichselement im Zugriff auf die innere Tabelle verschoben wurde, wird darauf durch die folgende Angabe im Zugriff der inneren Tabelle hingewiesen:

```
Process Hash Table For Join
```

- Beim Zugriff der äußeren Tabelle kann eine Prüftabelle erstellt werden, um die Leistung des Joins zu steigern. Auf die Erstellung der Prüftabelle wird durch die folgende Angabe im Zugriff der äußeren Tabelle hingewiesen:

```
Process Probe Table For Hash Join
```

- Die geschätzte erforderliche Anzahl Byte zum Erstellen der Hashtabelle wird durch folgende Angabe dargestellt:

```
Estimated Build Size: n
```

- Die geschätzte erforderliche Anzahl Byte für die Prüftabelle wird durch folgende Angabe dargestellt:

```
Estimated Probe Size: n
```

Bei einem Join mit Verschachtelungsschleife kann die folgende zusätzliche Angabe direkt nach der Joinangabe angezeigt werden:

```
Piped Inner
```

Diese Angabe bedeutet, dass die innere Tabelle des Joins das Ergebnis einer anderen Reihe von Operationen ist. Dies wird auch als eine *zusammengesetzte innere (composite inner)* Tabelle bezeichnet.

Wenn ein Join mehr als zwei Tabellen umfasst, müssen die EXPLAIN-Schritte von oben nach unten gelesen werden. Angenommen, die EXPLAIN-Ausgabe hat folgende Struktur:

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

Die Ausführung würde in diesem Fall in folgenden Schritten ablaufen:

1. Abrufen einer den Bedingungen entsprechenden Zeile aus `W`
2. Join der Zeile aus `W` mit der (nächsten) Zeile aus `X` und Benennung des Ergebnisses als `P1` (für Jointeilergebnis Nr. 1)
3. Join von `P1` mit der (nächsten) Zeile aus `Y` zum Erstellen von `P2`
4. Join von `P2` mit der (nächsten) Zeile aus `Z` zum Erstellen einer vollständigen Ergebniszeile
5. Wenn weitere Zeilen in `Z` sind, weiter mit Schritt 4
6. Wenn weitere Zeilen in `Y` sind, weiter mit Schritt 3
7. Wenn weitere Zeilen in `X` sind, weiter mit Schritt 2
8. Wenn weitere Zeilen in `W` sind, weiter mit Schritt 1

Datenstrominformationen: In einem Zugriffsplan ist es oft erforderlich, die Erstellung und den Fluss von Daten von einer Reihe von Operationen zur anderen zu steuern. Das Konzept des Datenstroms erlaubt es, eine Gruppe von Operationen innerhalb eines Zugriffsplans als Einheit zu steuern. Der Beginn eines Datenstroms wird durch folgende Anweisung gekennzeichnet:

```
Data Stream n
```

Hierbei ist `n` eine eindeutige Kennung, die zur leichteren Bezugnahme von `db2expln` zugeordnet wird. Das Ende des Datenstroms wird durch folgende Angabe gekennzeichnet:

```
End of Data Stream n
```

Alle Operationen zwischen diesen Angaben werden als Teil desselben Datenstroms angesehen.

Ein Datenstrom hat eine Anzahl von Merkmalen, und auf die einleitende Datenstromangabe können eine oder mehrere Angaben folgen, um diese Merkmale zu beschreiben:

- Wenn die Verarbeitung des Datenstroms von einem Wert abhängt, der früher im Zugriffsplan generiert wurde, wird der Datenstrom mit folgender Angabe markiert:

  ```
  Correlated
  ```

- Ähnlich wie bei einer sortierten temporären Tabelle zeigen die folgenden Angaben, ob die Ergebnisse des Datenstroms im Speicher behalten werden:

  ```
  Piped
  ```

  und

  ```
  Not Piped
  ```

  Wie bei temporären Tabellen kann ein über eine Pipe geleiteter Datenstrom auf die Platte geschrieben werden, wenn zur Ausführungszeit zu wenig Speicher vorhanden ist. Der Zugriffsplan sieht beide Möglichkeiten vor.

- Die folgende Angabe bedeutet, dass nur ein einziger Satz aus diesem Datenstrom benötigt wird:

  ```
  Single Record
  ```

Wenn auf einen Datenstrom zugegriffen wird, wird die folgende Angabe in der Ausgabe angezeigt:

```
Access Data Stream n
```

Informationen zu INSERT, UPDATE und DELETE: Der EXPLAIN-Text für diese SQL-Anweisungen ist selbsterklärend. Nachfolgend sind mögliche Texte für diese SQL-Operationen dargestellt:

```
Insert: Table Name = schema.name ID = ts,n
Update: Table Name = schema.name ID = ts,n
Delete: Table Name = schema.name ID = ts,n
Insert: Hierarchy Table Name = schema.name ID = ts,n
Update: Hierarchy Table Name = schema.name ID = ts,n
Delete: Hierarchy Table Name = schema.name ID = ts,n
Insert: Materialized Query Table = schema.name ID = ts,n
Update: Materialized Query Table = schema.name ID = ts,n
Delete: Materialized Query Table = schema.name ID = ts,n
Insert: Global Temporary Table ID = ts, tn
Update: Global Temporary Table ID = ts, tn
Delete: Global Temporary Table ID = ts, tn
```

Informationen zur Vorbereitung von Block- und Zeilen-IDs: Für einige Zugriffspläne ist es effizienter, wenn die den Bedingungen entsprechenden Satz-IDs (RIDs) und Block-IDs (BIDs) sortiert und mehrfach auftretende Werte entfernt werden (bei OR-Joins von Indizes) bzw. wenn eine Technik verwendet wird, mit der IDs, die in allen verwendeten Indizes auftreten (bei AND-Joins von Indizes), identifiziert werden, bevor der tatsächliche Zugriff auf die Tabelle erfolgt. Es gibt drei Hauptgründe für die Vorbereitung von IDs, auf die in den EXPLAIN-Angaben hingewiesen wird:

- Die beiden folgenden Angaben bedeuten, dass ein OR-Join von Indizes (Index ORing) zur Vorbereitung der Liste der qualifizierten IDs verwendet wird:

  ```
  Index ORing Preparation
  Block Index ORing Preparation
  ```

  Index ORing (OR-Join von Indizes) bezeichnet ein Verfahren, bei dem mehrere Indexzugriffe vorgenommen und die Ergebnisse kombiniert werden, sodass die unterschiedlichen IDs, die mindestens in einem der Indizes auftreten, auf die zugegriffen wird, zusammengefasst werden. Das Optimierungsprogramm erwägt die Verwendung dieses Verfahrens, wenn Vergleichselemente durch Schlüsselwörter OR verknüpft werden oder ein Vergleichselement IN existiert. Die Zugriffe können auf den gleichen Index oder auf verschiedene Indizes erfolgen.
- Ein weiterer Grund für die Vorbereitung von IDs ist die Vorbereitung der Eingabedaten, die während des Vorablesezugriffs über Listen verwendet werden sollen, wie dies durch folgende Angaben angezeigt wird:

  ```
  List Prefetch Preparation
  Block List Prefetch RID Preparation
  ```
- *Index ANDing* (AND-Joins von Indizes) bezeichnet ein Verfahren, beim dem mehrere Indexzugriffe vorgenommen und die Ergebnisse kombiniert werden, sodass nur die IDs zusammengefasst werden, die in allen Indizes auftreten, auf die zugegriffen wird. Die folgenden Angaben zeigen an, dass die Verarbeitung eines AND-Joins von Indizes gestartet wird:

  ```
  Index ANDing
  Block Index ANDing
  ```

  Wenn das Optimierungsprogramm die Größe der Ergebnismenge abgeschätzt hat, wird der Schätzwert mit der folgenden Angabe angezeigt:

  ```
  Optimizer Estimate of Set Size: n
  ```

  Die Filteroperationen bei AND-Joins von Indizes verarbeiten die IDs und verwenden Bit-Filtermethoden, um die IDs zu bestimmen, die in allen Indizes vorkommen, auf die zugegriffen wird. Die folgenden Angaben weisen darauf hin, dass IDs für AND-Joins von Indizes verarbeitet werden:

  ```
  Index ANDing Bitmap Build Using Row IDs
  Index ANDing Bitmap Probe Using Row IDs
  Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Build Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs and Build Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs
  Block Index ANDing Bitmap Probe Using Row IDs
  ```

  Wenn das Optimierungsprogramm die Größe der Ergebnismenge für eine Bitmaske (Bitmap) abgeschätzt hat, wird der Schätzwert mit der folgenden Angabe angezeigt:

  ```
  Optimizer Estimate of Set Size: n
  ```

Bei jeder Art von ID-Vorbereitung wird die Möglichkeit, dass ein Vorablesezugriff über Listen erfolgen kann, mit folgender Angabe angezeigt:

```
Prefetch: Enabled
```

Informationen zu Spaltenberechnungen (Aggregation): Spaltenberechnungen (Aggregation) werden für die Zeilen vorgenommen, die den etwaigen, in den Vergleichselementen der SQL-Anweisungen festgelegten Bedingungen entsprechen. Wenn eine Art von Spaltenfunktion auszuführen ist, wird eine der folgenden Angaben angezeigt:

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

Die Angabe *Predicate aggregation* besagt, dass die Spaltenberechnungsoperation zur Verarbeitung als Vergleichselement auf den Zeitpunkt, zu dem auf die Daten wirklich zugegriffen wird, verschoben wurde.

Unter jeder der oben aufgeführten Angaben über Spaltenberechnungen befindet sich eine Angabe, die die Art der durchgeführten Spaltenfunktion anzeigt:

```
Group By
Column Function(s)
Single Record
```

Die spezifische Spaltenfunktion kann von der ursprünglichen SQL-Anweisung abgeleitet werden. Ein einzelner Satz (Single Record) wird aus einem Index abgerufen, um eine Operation MIN oder MAX auszuführen.

Wenn eine Spaltenberechnung über Vergleichselemente verwendet wird, folgt auf die Angabe über den Tabellenzugriff, in der die Spaltenberechnung auftrat, eine Angabe über den Abschluss der Spaltenberechnung, die alle erforderlichen Verarbeitungsschritte bei Gruppenende bzw. bei Dateiende vornimmt. Dazu wird eine der folgenden Zeilen in der Ausgabe angezeigt:

```
Aggregation Completion
Partial Aggregation Completion
Intermediate Aggregation Completion
Final Aggregation Completion
```

Informationen zur Parallelverarbeitung: Für die parallele Ausführung einer SQL-Anweisung (entweder mit partitionsinterner oder partitionsübergreifender Parallelität) sind einige besondere Operationen erforderlich. Die Operationen für Parallelpläne werden im Folgenden beschrieben.

- Bei der Ausführung eines partitionsinternen Parallelplans werden Abschnitte des Plans gleichzeitig mithilfe verschiedener Subagenten ausgeführt. Die Erstellung der Subagenten wird durch folgende Angabe angezeigt:

  ```
  Process Using n Subagents
  ```

- Bei der Ausführung eines partitionsübergreifenden parallelen Zugriffsplans wird der Abschnitt (Section) in mehrere Teilbereiche (Subsections) geteilt. Jeder Teilbereich wird zur Ausführung an einen oder mehrere Knoten gesendet. Ein wichtiger Teilbereich ist der *Koordinatorteilbereich*. Der Koordinatorteilbereich ist der erste Teilbereich in jedem Plan. Er erhält zunächst die Steuerung und ist dann für die Verteilung der anderen Teilbereiche und die Rückgabe von Ergebnissen an die aufrufende Anwendung zuständig.

Die Verteilung von Teilbereichen wird durch folgende Angabe angezeigt:

```
Distribute Subsection #n
```

Die Knoten, die einen Teilbereich empfangen, können durch eine von acht Methoden bestimmt werden:

- Die folgende Angabe bedeutet, dass der Teilbereich abhängig vom Wert der Spalten zu einem Knoten innerhalb der Datenbankpartitionsgruppe gesendet wird.

  ```
  Directed by Hash
  |  #Columns = n
  |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an einen vorbestimmten Knoten gesendet wird. (Dies tritt häufig auf, wenn die Anweisung mit der Funktion NODENUMBER() arbeitet.)

  ```
  Directed by Node Number
  ```

- Die folgende Angabe weist darauf hin, dass der Teilbereich an den Knoten gesendet wird, der einer vorbestimmten Datenbankpartitionsnummer in der angegebenen Datenbankpartitionsgruppe entspricht. (Dies tritt häufig auf, wenn die Anweisung mit der Funktion PARTITION() arbeitet.)

  ```
  Directed by Partition Number
  |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an den Knoten gesendet wird, der die aktuelle Zeile für den Cursor der Anwendung verfügbar gemacht hat.

  ```
  Directed by Position
  ```

- Die folgende Angabe bedeutet, dass nur ein einziger Knoten, der bei der Kompilierung der Anweisung festgelegt wird, den Teilbereich empfängt.

  ```
  Directed to Single Node
  |  Node Number = n
  ```

- Die folgenden Angaben bedeuten, dass der Teilbereich auf dem Koordinatorknoten ausgeführt wird.

  ```
  Directed to Application Coordinator Node
  Directed to Local Coordinator Node
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an alle aufgeführten Knoten gesendet wird.

  ```
  Broadcast to Node List
  |  Nodes = n1, n2, n3, ...
  ```

- Die folgende Angabe bedeutet, dass nur ein einziger Knoten, der bei der Ausführung der Anweisung festgelegt wird, den Teilbereich empfängt.

  ```
  Directed to Any Node
  ```

- Tabellenwarteschlangen werden zum Versetzen von Daten zwischen Teilbereichen in einer Umgebung mit partitionierten Datenbanken oder zwischen Subagenten in einer symmetrischen Mehrprozessorumgebung (SMP) verwendet. Tabellenwarteschlangen werden folgendermaßen beschrieben:

  - Die folgenden Angaben zeigen an, dass Daten in eine Tabellenwarteschlange eingefügt werden:

    ```
    Insert Into Synchronous Table Queue  ID = qn
    Insert Into Asynchronous Table Queue  ID = qn
    Insert Into Synchronous Local Table Queue  ID = qn
    Insert Into Asynchronous Local Table Queue  ID = qn
    ```

  - Bei Tabellenwarteschlangen einer Datenbankpartition wird das Ziel für Zeilen, die in die Tabellenwarteschlange eingefügt werden, durch eine der folgenden Angaben angezeigt.

Alle Zeilen werden an den Koordinatorknoten gesendet:

```
Broadcast to Coordinator Node
```

Alle Zeilen werden an jede Datenbankpartition gesendet, auf der der angegebene Teilbereich ausgeführt wird:

```
Broadcast to All Nodes of Subsection n
```

Jede Zeile wird abhängig von den Werten in der Zeile an eine Datenbankpartition gesendet:

```
Hash to Specific Node
```

Jede Zeile wird an eine Datenbankpartition gesendet, die während der Ausführung der Anweisung bestimmt wird:

```
Send to Specific Node
```

Jede Zeile wird an einen zufällig bestimmten Knoten gesendet:

```
Send to Random Node
```

– In einigen Fällen ist es möglich, dass die Tabellenwarteschlange der Datenbankpartition einige Zeilen wegen Kapazitätsüberschreitung in eine temporäre Tabelle versetzen muss. Diese Möglichkeit wird durch folgende Angabe angezeigt:

```
Rows Can Overflow to Temporary Table
```

– Auf einen Tabellenzugriff, der eine verschobene Operation (Pushdown) zum Einfügen von Zeilen in eine Tabellenwarteschlange enthält, folgt eine Abschlussangabe (Completion), die die Zeilen handhabt, die nicht unmittelbar gesendet werden konnten. Dazu wird eine der folgenden Zeilen in der Ausgabe angezeigt:

```
Insert Into Synchronous Table Queue Completion  ID = qn
Insert Into Asynchronous Table Queue Completion  ID = qn
Insert Into Synchronous Local Table Queue Completion  ID = qn
Insert Into Asynchronous Local Table Queue Completion  ID = qn
```

– Die folgenden Angaben zeigen an, dass Daten aus einer Tabellenwarteschlange abgerufen werden:

```
Access Table Queue  ID = qn
Access Local Table Queue  ID = qn
```

Diesen Nachrichten folgt stets eine Angabe der Anzahl der Zeilen, die abgerufen werden.

```
#Columns = n
```

– Wenn die Tabellenwarteschlange die Zeilen auf der Empfängerseite sortiert, wird der Angabe über den Tabellenwarteschlangenzugriff außerdem eine der folgenden Nachrichten beigefügt:

```
Output Sorted
Output Sorted and Unique
```

Diesen Nachrichten folgt eine Angabe der Anzahl der Schlüsselspalten, die für die Sortierung verwendet wurden.

```
#Key Columns = n
```

Für jede Spalte im Sortierschlüssel wird eine der folgenden Angaben angezeigt:

```
Key n: (Ascending)
Key n: (Descending)
```

– Wenn Vergleichselemente auf der Empfängerseite der Tabellenwarteschlange auf die Zeilen angewendet werden, wird folgende Nachricht in der Ausgabe angezeigt:

```
Residual Predicate(s)
|  #Predicates = n
```

- Einige Teilbereiche in Umgebungen mit partitionierten Datenbanken springen explizit zurück an den Start des Teilbereichs. Dies wird durch folgende Angabe angezeigt:

```
Jump Back to Start of Subsection
```

Informationen zu Abfragen auf föderierte Datenbanken: Für die Ausführung einer SQL-Anweisung in einer föderierten Datenbank ist die Fähigkeit erforderlich, Teile der betreffenden Anweisung an anderen Datenquellen auszuführen.

Die folgende Angabe weist darauf hin, dass eine Datenquelle gelesen wird:

```
Ship Distributed Subquery #n
|   #Columns = n
```

Es ist möglich, dass Vergleichselemente auf die Daten angewandt werden, die von der verteilten Unterabfrage (Distributed Subquery) zurückgegeben werden. Die Anzahl der Vergleichselemente, die angewendet werden, wird folgendermaßen angezeigt:

```
Residual Predicate(s)
|   #Predicates = n
```

Eine INSERT-, UPDATE- oder DELETE-Operation, die an einer Datenquelle stattfindet, wird durch die entsprechende Angabe gezeigt:

```
Ship Distributed Insert #n
Ship Distributed Update #n
Ship Distributed Delete #n
```

Wenn eine Tabelle an einer Datenquelle explizit gesperrt wird, wird dies durch folgende Angabe gezeigt:

```
Ship Distributed Lock Table #n
```

DDL-Anweisungen für eine Datenquelle werden in zwei Teile aufgeteilt. Der Teil, der an der Datenquelle aufgerufen wird, wird wie folgt gezeigt:

```
Ship Distributed DDL Statement #n
```

Wenn der Server der föderierten Datenbanken eine partitionierte Datenbank ist, muss ein Teil der DDL-Anweisung auf dem Katalogknoten ausgeführt werden. Dies wird wie folgt angegeben:

```
Distributed DDL Statement #n Completion
```

Die Einzelheiten für die verteilten Unteranweisungen werden getrennt bereitgestellt. Die Optionen für verteilte Unteranweisungen werden nachfolgend beschrieben:

- Die Datenquelle für die Unterabfrage wird durch eine der folgenden Angaben gezeigt:

```
Server: servername (typ, version)
Server: servername (typ)
Server: servername
```

- Wenn die Datenquelle relational ist, wird das SQL für die Unteranweisung wie folgt dargestellt:

```
SQL Statement:
     anweisung
```

Nicht relationale Datenquellen werden wie folgt angegeben:

```
Non-Relational Data Source
```

- Die Kurznamen (Nicknames), auf die in der Unteranweisung verwiesen wird, werden folgendermaßen aufgelistet:

  ```
  Nicknames Referenced:
     schema.kurzname ID = n
  ```

  Wenn die Datenquelle relational ist, wird die Basistabelle für den Kurznamen wie folgt angegeben:

  ```
  Base = basisschema.basistabelle
  ```

  Wenn die Datenquelle nicht relational ist, wird die Quellendatei für den Kurznamen wie folgt angezeigt:

  ```
  Source File = dateiname
  ```

- Wenn die Werte vom Server mit föderierten Datenbanken an die Datenquelle übergeben werden, bevor die Unteranweisung ausgeführt wird, wird die Zahl der Werte folgendermaßen gezeigt:

  ```
  #Input Columns: n
  ```

- Wenn die Werte von der Datenquelle an den Server mit föderierten Datenbanken übergeben werden, nachdem die Unteranweisung ausgeführt wird, wird die Zahl der Werte folgendermaßen gezeigt:

  ```
  #Output Columns: n
  ```

Verschiedene EXPLAIN-Informationen:

- Abschnitte für Anweisungen der Datendefinitionssprache (DDL - Data Definition Language) werden in der Ausgabe folgendermaßen gekennzeichnet:

  ```
  DDL Statement
  ```

  Für DDL-Anweisungen wird keine weitere EXPLAIN-Ausgabe bereitgestellt.

- Abschnitte für SET-Anweisungen für die aktualisierbaren Sonderregister wie **CURRENT EXPLAIN SNAPSHOT** werden in der Ausgabe folgendermaßen gekennzeichnet:

  ```
  SET Statement
  ```

  Für SET-Anweisungen wird keine weitere EXPLAIN-Ausgabe bereitgestellt.

- Wenn die SQL-Anweisung die Klausel DISTINCT enthält, kann der folgende Text in der Ausgabe angezeigt werden:

  ```
  Distinct Filter  #Columns = n
  ```

  Dabei ist `n` die Anzahl von Spalten, die beim Abrufen eindeutiger Zeilen verwendet wird. Zum Abrufen eindeutiger Zeilenwerte müssen die Zeilen geordnet sein, sodass gleiche Werte übersprungen werden können. Diese Angabe wird nicht angezeigt, wenn der Datenbankmanager gleiche Werte nicht explizit entfernen muss wie in folgenden Fällen:

  – Es existiert ein eindeutiger Index, und alle Spalten im Indexschlüssel sind Teil der Operation für die Klausel DISTINCT.
  – Die mehrfach auftretenden Werte können beim Sortieren entfernt werden.

- Die folgende Angabe wird angezeigt, wenn die nächste Operation von einer bestimmten Satz-ID (RID) abhängig ist:

  ```
  Positioned Operation
  ```

Wenn die Positionierungsoperation an einer Datenquelle in einer föderierten Datenbankumgebung stattfindet, sieht die Angabe wie folgt aus:

```
Distributed Positioned Operation
```

Diese Angabe wird für jede SQL-Anweisung angezeigt, die die Syntax WHERE CURRENT OF verwendet.

- Die folgende Angabe wird angezeigt, wenn Vergleichselemente vorhanden sind, die auf das Ergebnis angewendet werden müssen, aber die nicht als Teil einer anderen Operation angewendet werden konnten (Restvergleichselemente):

  ```
  Residual Predicate Application
  |  #Predicates = n
  ```

- Die folgende Angabe wird angezeigt, wenn ein Operator UNION in der SQL-Anweisung enthalten ist:

  ```
  UNION
  ```

- Die folgende Angabe wird angezeigt, wenn sich im Zugriffsplan eine Operation befindet, deren einziger Zweck das Erstellen von Zeilenwerten zur Verwendung durch nachfolgende Operationen ist:

  ```
  Table Constructor
  |  n-Row(s)
  ```

  Table Constructors können verwendet werden, um Werte in einer Menge in eine Reihe von Zeilen umzuwandeln, die anschließend an nachfolgende Operationen übergeben werden. Wenn die nächste Zeile von einem Table Constructor angefordert wird, wird die folgende Angabe angezeigt:

  ```
  Access Table Constructor
  ```

- Die folgende Angabe wird angezeigt, wenn eine Operation existiert, die nur unter bestimmten Bedingungen verarbeitet wird:

  ```
  Conditional Evaluation
  |  Condition #n:
  |  #Predicates = n
  |  Action #n:
  ```

  Durch bedingte Auswertung werden Aktivitäten wie die SQL-Anweisung CASE oder interne Mechanismen wie referenzielle Integritätsbedingungen oder Trigger implementiert. Wenn keine Aktion angezeigt wird, werden nur Datenbearbeitungsoperationen verarbeitet, wenn die Bedingung wahr ist.

- Eine der folgenden Angaben wird angezeigt, wenn eine Unterabfrage nach ALL, ANY oder EXISTS im Zugriffsplan verarbeitet wird:
  - `ANY/ALL Subquery`
  - `EXISTS Subquery`
  - `EXISTS SINGLE Subquery`
- Vor bestimmten UPDATE- und DELETE-Operationen muss die Position einer speziellen Zeile in der Tabelle bestimmt werden. Dies wird durch folgende Angabe gekennzeichnet:

  ```
  Establish Row Position
  ```

- Die folgenden Angaben werden für Löschungen an MDC-Tabellen angezeigt, für die eine Rollout-Optimierung in Betracht kommt:

  ```
  CELL DELETE with deferred cleanup
  ```

  oder

  ```
  CELL DELETE with immediate cleanup
  ```

- Die folgende Angabe wird angezeigt, wenn Zeilen an die Anwendung zurückgegeben werden:

```
Return Data to Application
|  #Columns = n
```

Wenn die Operation in einen Tabellenzugriff verschoben (Pushdown) wurde, wird eine Abschlussphase (Completion) erforderlich. Diese Phase wird wie folgt angezeigt:

```
Return Data Completion
```

- Die folgenden Informationen werden angezeigt, wenn eine gespeicherte Prozedur aufgerufen wird:

```
Call Stored Procedure
|   Name = schema.funktionsname
|   Specific Name = spezifischer_name
|   SQL Access Level = zugriffsebene
|   Language = sprache
|   Parameter Style = parmstil
|   Expected Result Sets = n
|   Fenced                      Not Deterministic
|   Called on NULL Input        Disallow Parallel
|   Not Federated               Not Threadsafe
```

- Die folgenden Informationen werden angezeigt, wenn mindestens ein LOB-Querverweis freigegeben wird:

```
Free LOB Locators
```

Beispiele für die Ausgabe von db2expln und dynexpln

Zum besseren Verständnis finden Sie im Folgenden Beispiele, die den Aufbau und das Format der Ausgabe von `db2expln` und `dynexpln` verdeutlichen. Diese Beispiele wurden für die Beispieldatenbank SAMPLE ausgeführt, die mit DB2 ausgeliefert wird. Jedes Beispiel wird kurz erläutert. Die signifikanten Unterschiede von einem Beispiel zum nächsten wurden **fett** hervorgehoben.

Beispiel 1: keine Parallelität: In diesem Beispiel wird einfach eine Liste aller Namen von Mitarbeitern (employee), ihrer Aufgaben (job), ihrer Abteilungen (department) und Standorte (location) sowie der Namen der Projekte, an denen sie arbeiten, abgerufen. Das wesentliche Merkmal dieses Zugriffsplans besteht darin, dass die relevanten Daten aus allen angegebenen Tabellen durch Hash-Joins verknüpft werden. Da keine Indizes verfügbar sind, führt der Zugriffsplan beim Join einer Tabelle eine Tabellensuche in der jeweiligen Tabelle aus.

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:05:00

 Bind Timestamp = 2002-01-04-14.05.00.415403

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ----------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
```

```
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 120.518692
Estimated Cardinality = 221.535980

(    6) Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
         |  #Columns = 3
         |  Relation Scan
         |  |  Prefetch: Eligible
         |  Lock Intents
         |  |  Table: Intent Share
         |  |  Row  : Next Key Share
(    6)  |  Process Build Table for Hash Join
(    2) Hash Join
         |  Estimated Build Size: 7111
         |  Estimated Probe Size: 9457
(    5)  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
         |  |  #Columns = 2
         |  |  Relation Scan
         |  |  |  Prefetch: Eligible
         |  |  Lock Intents
         |  |  |  Table: Intent Share
         |  |  |  Row  : Next Key Share
(    5)  |  |  Process Build Table for Hash Join
(    3)  |  Hash Join
         |  |  Estimated Build Size: 5737
         |  |  Estimated Probe Size: 6421
(    4)  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
         |  |  |  #Columns = 3
         |  |  |  Relation Scan
         |  |  |  |  Prefetch: Eligible
         |  |  |  Lock Intents
         |  |  |  |  Table: Intent Share
         |  |  |  |  Row  : Next Key Share
(    4)  |  |  |  Process Probe Table for Hash Join
(    1) Return Data to Application
         |  #Columns = 5

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
              HSJOIN
              (   2)
              /     \
         HSJOIN      TBSCAN
         (   3)      (   6)
         /     \       |
    TBSCAN    TBSCAN   Table:
    (   4)    (   5)   DOOLE
      |         |      EMPLOYEE
  Table:      Table:
  DOOLE       DOOLE
  DEPARTMENT  PROJECT
```

Im ersten Teil des Plans wird auf die Tabellen DEPARTMENT und PROJECT zugegriffen, die über einen Hash-Join verknüpft werden. Das Ergebnis dieses Joins wird mit der Tabelle EMPLOYEE verknüpft. Die Ergebniszeilen werden an die Anwendung zurückgegeben.

Beispiel 2: Zugriffsplan für Einzelpartition mit partitionsinterner Parallelität: In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt, jedoch wurde diese Abfrage für eine 4-Wege-SMP-Maschine kompiliert.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:12:38

 Bind Timestamp = 2002-01-04-14.12.38.732627

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 133.934692
Estimated Cardinality = 221.535980

(    2) Process Using 4 Subagents
(    7) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Parallel Scan
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    7) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 9457
(    6) |  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  |  #Columns = 2
        |  |  |  Parallel Scan
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    6) |  |  |  Process Build Table for Hash Join
(    4) |  |  Hash Join
```

```
        |  |  |  Estimated Build Size: 5737
        |  |  |  Estimated Probe Size: 6421
(    5) |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  |  #Columns = 3
        |  |  |  |  Parallel Scan
        |  |  |  |  Relation Scan
        |  |  |  |  |  Prefetch: Eligible
        |  |  |  |  Lock Intents
        |  |  |  |  |  Table: Intent Share
        |  |  |  |  |  Row  : Next Key Share
(    5) |  |  |  |  Process Probe Table for Hash Join
(    2) |  Insert Into Asynchronous Local Table Queue  ID = q1
(    2) Access Local Table Queue  ID = q1  #Columns = 5
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
               LTQ
              (   2)
                |
              HSJOIN
              (   3)
             /      \
         HSJOIN      TBSCAN
         (   4)      (   7)
        /      \        |
    TBSCAN    TBSCAN   Table:
    (   5)    (   6)   DOOLE
      |         |      EMPLOYEE
  Table:    Table:
  DOOLE     DOOLE
  DEPARTMENT PROJECT
```

Dieser Plan stimmt mit dem Plan des ersten Beispiels weitgehend überein. Der Unterschied besteht in der Erstellung von vier Subagenten beim ersten Starten des Plans und der Tabellenwarteschlange am Ende des Plans, um die Ergebnisse der Arbeit aller Subagenten aufzunehmen, bevor sie an die Anwendung zurückgegeben werden.

Beispiel 3: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender Parallelität: Dieses Beispiel zeigt wiederum dieselbe SQL-Anweisung wie das erste Beispiel, jedoch wurde hier die Abfrage für eine partitionierte Datenbank kompiliert, die aus drei Datenbankpartitionen besteht.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:54:57

 Bind Timestamp = 2002-01-04-14.54.57.033666

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = No
```

```
 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 118.483406
Estimated Cardinality = 474.720032

        Coordinator Subsection:
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    8)    Access Table Queue  ID = q2  #Columns = 2
(    3)    Hash Join
           |  Estimated Build Size: 5737
           |  Estimated Probe Size: 8015
(    6)    |  Access Table Queue  ID = q3  #Columns = 3
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5333
           |  |  Estimated Probe Size: 6421
(    5)    |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
           |  |  |  #Columns = 3
           |  |  |  Relation Scan
           |  |  |  |  Prefetch: Eligible
           |  |  |  Lock Intents
           |  |  |  |  Table: Intent Share
           |  |  |  |  Row  : Next Key Share
(    5)    |  |  |  Process Probe Table for Hash Join
(    2)    Insert Into Asynchronous Table Queue  ID = q1
           |  Broadcast to Coordinator Node
           |  Rows Can Overflow to Temporary Table

        Subsection #2:
(    9)    Access Table Name = DOOLE.PROJECT  ID = 2,7
           |  #Columns = 2
           |  Relation Scan
           |  |  Prefetch: Eligible
           |  Lock Intents
           |  |  Table: Intent Share
           |  |  Row  : Next Key Share
(    9)    |  Insert Into Asynchronous Table Queue  ID = q2
           |  |  Hash to Specific Node
           |  |  Rows Can Overflow to Temporary Tables
(    8)    Insert Into Asynchronous Table Queue Completion  ID = q2

        Subsection #3:
```

```
(    7)    Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
           |  #Columns = 3
           |  Relation Scan
           |  |  Prefetch: Eligible
           |  Lock Intents
           |  |  Table: Intent Share
           |  |  Row  : Next Key Share
(    7)    |  Insert Into Asynchronous Table Queue  ID = q3
           |  |  Hash to Specific Node
           |  |  Rows Can Overflow to Temporary Tables
(    6)    Insert Into Asynchronous Table Queue Completion  ID = q3

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
               BTQ
              (   2)
                |
              HSJOIN
              (   3)
             /      \
       HSJOIN          DTQ
       (   4)         (   8)
      /      \          |
   TBSCAN     DTQ     TBSCAN
   (   5)    (   6)   (   9)
     |         |       |
 Table:      TBSCAN   Table:
 DOOLE       (   7)   DOOLE
 DEPARTMENT    |      PROJECT
            Table:
            DOOLE
            EMPLOYEE
```

Dieser Plan enthält dieselben Bestandteile wie der Plan im ersten Beispiel, aber der Bereich wurde in vier Teilbereiche (Subsection) unterteilt. Die Teilbereiche haben folgende Aufgaben:

- **Coordinator Subsection (Koordinatorteilbereich)**. Dieser Teilbereich koordiniert die anderen Teilbereiche. In diesem Plan sorgt er dafür, dass die anderen Teilbereiche verteilt werden, und er verwendet anschließend eine Tabellenwarteschlange, um die Ergebnisse zu sammeln, die an die Anwendung zurückgegeben werden sollen.
- **Subsection #1**. Dieser Teilbereich durchsucht die Tabellenwarteschlange q2 und verwendet einen Hash-Join, um sie mit den Daten aus Tabellenwarteschlange q3 zu verknüpfen. Ein zweiter Hash-Join fügt anschließend die Daten aus der Tabelle DEPARTMENT hinzu. Die verknüpften Zeilen werden dann über Tabellenwarteschlange q1 an den Koordinatorteilbereich gesendet.
- **Subsection #2**. Dieser Teilbereich durchsucht die Tabelle PROJECT und verteilt sie per Hashverfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.
- **Subsection #3**. Dieser Teilbereich durchsucht die Tabelle EMPLOYEE und verteilt sie per Hashverfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.

Beispiel 4: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender und partitionsinterner Parallelität: In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt, jedoch wurde die Abfrage für eine partitionierte Datenbank mit drei Datenbankpartitionen kompiliert, die sich jeweils auf einer 4-Wege-SMP-Maschine befinden.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:58:35

 Bind Timestamp = 2002-01-04-14.58.35.169555

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION -------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 145.198898
Estimated Cardinality = 474.720032

        Coordinator Subsection:
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    3)    Process Using 4 Subagents
(   10)    |  Access Table Queue  ID = q3  #Columns = 2
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5737
           |  |  Estimated Probe Size: 8015
(    7)    |  |  Access Table Queue  ID = q5  #Columns = 3
(    5)    |  |  Hash Join
           |  |  |  Estimated Build Size: 5333
           |  |  |  Estimated Probe Size: 6421
(    6)    |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
           |  |  |  |  #Columns = 3
           |  |  |  |  Parallel Scan
```

```
            |  |  |  |  Relation Scan
            |  |  |  |  |  Prefetch: Eligible
            |  |  |  |  Lock Intents
            |  |  |  |  |  Table: Intent Share
            |  |  |  |  |  Row  : Next Key Share
(    6)     |  |  |  |  Process Probe Table for Hash Join
(    3)     |  Insert Into Asynchronous Local Table Queue  ID = q2
(    3)    Access Local Table Queue  ID = q2  #Columns = 5
(    2)    Insert Into Asynchronous Table Queue  ID = q1
            |  Broadcast to Coordinator Node
            |  Rows Can Overflow to Temporary Table

        Subsection #2:
(   11)    Process Using 4 Subagents
(   12)    |  Access Table Name = DOOLE.PROJECT  ID = 2,7
           |  |  #Columns = 2
           |  |  Parallel Scan
           |  |  Relation Scan
           |  |  |  Prefetch: Eligible
           |  |  Lock Intents
           |  |  |  Table: Intent Share
           |  |  |  Row  : Next Key Share
(   11)    |  Insert Into Asynchronous Local Table Queue  ID = q4
(   11)    Access Local Table Queue  ID = q4  #Columns = 2
(   10)    Insert Into Asynchronous Table Queue  ID = q3
           |  Hash to Specific Node
           |  Rows Can Overflow to Temporary Tables

        Subsection #3:
(    8)    Process Using 4 Subagents
(    9)    |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
           |  |  #Columns = 3
           |  |  Parallel Scan
           |  |  Relation Scan
           |  |  |  Prefetch: Eligible
           |  |  Lock Intents
           |  |  |  Table: Intent Share
           |  |  |  Row  : Next Key Share
(    8)    |  Insert Into Asynchronous Local Table Queue  ID = q6
(    8)    Access Local Table Queue  ID = q6  #Columns = 3
(    7)    Insert Into Asynchronous Table Queue  ID = q5
           |  Hash to Specific Node
           |  Rows Can Overflow to Temporary Tables

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
               BTQ
              (   2)
                |
               LTQ
              (   3)
                |
              HSJOIN
              (   4)
              /     \
          HSJOIN      DTQ
          (   5)     (  10)
          /     \       |
      TBSCAN    DTQ     LTQ
      (   6)   (   7)  (  11)
        |        |       |
```

```
Table:       LTQ      TBSCAN
DOOLE       (   8)    (  12)
DEPARTMENT    |         |
            TBSCAN    Table:
            (   9)    DOOLE
              |       PROJECT
           Table:
           DOOLE
           EMPLOYEE
```

Dieser Plan ist ähnlich wie der im dritten Beispiel, nur dass mehrere Subagenten jeden Teilbereich ausführen. Außerdem sammelt am Ende eines jeden Teilbereichs eine lokale Tabelle die Ergebnisse von allen Subagenten, bevor die entsprechenden Zeilen in die zweite Tabellenwarteschlange eingefügt werden, die per Hashverfahren an einen bestimmten Knoten gesendet wird.

Beispiel 5: Zugriffsplan für eine föderierte Datenbank: In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt. Die Abfrage wurde jedoch für eine föderierte Datenbank kompiliert, bei der sich die Tabellen DEPARTMENT und PROJECT in einer Datenquelle und die Tabelle EMPLOYEE auf dem Server für die föderierten Datenbanken befinden.

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/11
 Prep Time = 13:52:48

 Bind Timestamp = 2002-01-11-13.52.48.325413

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 1804.625000
Estimated Cardinality = 112000.000000

(    7) Ship Distributed Subquery #2
        |  #Columns = 2
(    2) Hash Join
        |  Estimated Build Size: 48444
        |  Estimated Probe Size: 232571
(    6) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
```

```
        |  |  |  Row  : Next Key Share
(    6) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 64606
(    4) |  |  Ship Distributed Subquery #1
        |  |  |  #Columns = 3
(    1) Return Data to Application
        |  #Columns = 5

Distributed Substatement #1:
(    4) Server: REMOTE  (DB2/UDB 8.1)
        SQL Statement:

             SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
             FROM "DOOLE"."DEPARTMENT" A0

        Nicknames Referenced:
           DOOLE.DEPARTMENT  ID = 32768
              Base = DOOLE.DEPARTMENT
        #Output Columns = 3

Distributed Substatement #2:
(    7) Server: REMOTE  (DB2/UDB 8.1)
        SQL Statement:

             SELECT A0."DEPTNO", A0."PROJNAME"
             FROM "DOOLE"."PROJECT" A0

        Nicknames Referenced:
           DOOLE.PROJECT  ID = 32769
              Base = DOOLE.PROJECT
        #Output Columns = 2

End of section

Optimizer Plan:

               RETURN
               (   1)
                 |
               HSJOIN
               (   2)
              /      \
        HSJOIN          SHIP
        (   3)         (   7)
       /      \          |
    SHIP      TBSCAN   Nickname:
   (   4)     (   6)   DOOLE
     |          |      PROJECT
 Nickname:   Table:
 DOOLE       DOOLE
 DEPARTMENT  EMPLOYEE
```

Dieser Plan enthält dieselben Bestandteile wie der Plan im ersten Beispiel. Abweichend vom ersten Beispiel stammen die Daten für zwei der Tabellen aus Datenquellen. Auf diese beiden Tabellen wird über verteilte Unterabfragen zugegriffen, die in diesem Fall einfach alle Zeilen der betreffenden Tabellen auswählen. Wenn die Daten zum Server mit föderierten Datenbanken zurückgegeben werden, werden sie mit den Daten aus der lokalen Tabelle verknüpft.

Beispiel 6: Operatoren XANDOR und XISCAN: Dieses Beispiel veranschaulicht, wie der Operator XANDOR die XISCAN-Suchoperationen von zwei einzelnen Indizes für XML-Daten (*IDX1* und *IDX2*) kombiniert, die für dieselbe Tabelle *XISCANTABLE* definiert wurden.

```
IBM DB2 Database SQL Explain Tool

******************** DYNAMIC ************************************

=================== STATEMENT ========================================

        Isolation Level          = Cursor Stability
        Blocking                 = Block Unambiguous Cursors
        Query Optimization Class = 5

        Partition Parallel       = No
        Intra-Partition Parallel = No

        SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "ATTALURI"


Query Statement:
  xquery
  for $c in db2-fn:xmlcolumn("XISCANTABLE.XMLCOL ")/a[@x="1" ]/b[@y=
         "2" ] return $c

Section Code Page = 819

Estimated Cost = 192.266113
Estimated Cardinality = 1.800000

(    6) Index ANDing and ORing over XML
         |  Xpath is
         |  |  /child::element(a)[./child::element(b)
         |  |  /attribute::attribute(y)(:Index Search over XML 1:)
         |  |   and ./attribute::attribute(x)(:Index Search over XML 2:)
         |  |  ]
         |  Index Search over XML 1
         |  |  Access Table Name = ATTALURI.XISCANTABLE
         |  |  |  Index Scan over XML:  Name = ATTALURI.IDX1  ID = 6
         |  |  |  |  Physical Index over XML
         |  |  |  |  Index Columns:
         |  |  |  |  |  1: XMLCOL (Ascending)
         |  |  |  #Key Columns = 4
         |  |  |  |  Start Key: Inclusive Value
         |  |  |  |  |  |  1: ?
         |  |  |  |  |  |  2: ?
         |  |  |  |  |  |  3: ?
         |  |  |  |  |  |  4: ?
         |  |  |  |  Stop Key: Inclusive Value
         |  |  |  |  |  |  1: ?
         |  |  |  |  |  |  2: ?
         |  |  |  Index-Only Access
         |  |  |  Index Prefetch: None
         |  |  |  Isolation Level: Uncommitted Read
         |  |  |  Lock Intents
         |  |  |  |  Table: Intent None
         |  |  |  |  Row  : None
         |  |  |  StopKey = StartKey
         |  |  |  Value Start Key = ?
         |  Index Search over XML 2
         |  |  Access Table Name = ATTALURI.XISCANTABLE
         |  |  |  Index Scan over XML:  Name = ATTALURI.IDX2  ID = 4
         |  |  |  |  Physical Index over XML
         |  |  |  |  Index Columns:
         |  |  |  |  |  1: XMLCOL (Ascending)
         |  |  |  #Key Columns = 4
         |  |  |  |  Start Key: Inclusive Value
         |  |  |  |  |  |  1: ?
         |  |  |  |  |  |  2: ?
         |  |  |  |  |  |  3: ?
         |  |  |  |  |  |  4: ?
         |  |  |  |  Stop Key: Inclusive Value
         |  |  |  |  |  |  1: ?
         |  |  |  |  |  |  2: ?
```

```
        |  |  |  Index-Only Access
        |  |  |  Index Prefetch: None
        |  |  |  Isolation Level: Uncommitted Read
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent None
        |  |  |  |  Row  : None
        |  |  |  StopKey = StartKey
        |  |  |  Value Start Key = ?
(    5) Insert Into Sorted Temp Table  ID = t1
        |  #Columns = 1
        |  #Sort Key Columns = 1
        |  |  Key 1: (Ascending)
        |  Sortheap Allocation Parameters:
        |  |  #Rows     = 2
        |  |  Row Width = 16
        |  Piped
        |  Duplicate Elimination
(    4) List Prefetch Preparation
(    4) |  Access Table Name = ATTALURI.XISCANTABLE  ID = 2,16
        |  |  #Columns = 1
        |  |  Fetch Using Prefetched List
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    2) Nested Loop Join
        |  Piped Inner
(    9) |  XML Doc Navigation
        |  |  Navigator is
        |  |  |  /fn:root($CONTEXT_NODE$())/child::element(a)(:#Xpath Predicates = 1:)
        |  |  |  [./child::element(b)(:Output nodeSeqRef :)
        |  |  |  (:#Xpath Predicates = 1:)
        |  |  |  /attribute::attribute(y) and
        |  |  |  ./attribute::attribute(x)]
(    1) Iterate over XML sequence for Xquery bindout
(    1) Return Data to Application
        |  #Columns = 1

End of section


Optimizer Plan:

                                    RETURN
                                    (   1)
                                      |
                                    NLJOIN
                                    (   2)
                                   /      \
                                FETCH    XSCAN
                               (----)    (   9)
                               /           \
                          RIDSCN            Table:
                          (   4)               ATTALURI
                            |                  XISCANTABLE
                           SORT
                          (   5)
                            |
                          XANDOR
                          (   6)
                       /---------/     \--------\
                    XISCAN                      XISCAN
                    (   6)                      (   6)
                   /      \                    /      \
              Index:       Table:          Index:       Table:
                 ATTALURI     ATTALURI        ATTALURI     ATTALURI
                 IDX1         XISCANTABLE     IDX2         XISCANTABLE
```

Beispiel 7: Operator XSCAN: In diesen Beispiel wird gezeigt, wie der Operator XSCAN im Zugriffsplan dargestellt werden kann. Dieser Operator verarbeitet Knotenverweise, die von einem Joinoperator (NLJOIN) mit verschachtelter Schleife übergeben wurden. Im Zugriffsplan wird dieses Element nicht mit einer direkten Eingabe dargestellt.

```
IBM DB2 Database SQL Explain Tool

******************** DYNAMIC ***************************************

=================== STATEMENT ==========================================

        Isolation Level          = Cursor Stability
        Blocking                 = Block Unambiguous Cursors
        Query Optimization Class = 5

        Partition Parallel       = No
        Intra-Partition Parallel = No

        SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "ATTALURI"


Query Statement:
  xquery
  for $b in db2-fn:xmlcolumn("XISCANTABLE.XMLCOL" )//book[position()<=
          2] return $b

Section Code Page = 819

Estimated Cost = 779592.625000
Estimated Cardinality = 540000000.000000

(    4) Access Table Name = ATTALURI.XISCANTABLE  ID = 2,16
         |  #Columns = 1
         |  Relation Scan
         |  |  Prefetch: Eligible
         |  Lock Intents
         |  |  Table: Intent Share
         |  |  Row  : Next Key Share
(    3) Nested Loop Join
         |  Piped Inner
(    6)  |  XML Doc Navigation
         |  |  Navigator is
         |  |  |  /fn:root($CONTEXT_NODE$())/descendant-or-self::node()(:Output nodeSeqRef :)
         |  |  |
(    5)  |  Nested Loop Join
         |  |  Piped Inner
(   11)  |  |  XML Doc Navigation
         |  |  |  Navigator is
         |  |  |  |  /fn:root($CONTEXT_NODE$())/child::element(book)(:Output nodeSeqRef :)
         |  |  |  |
(   10)  |  |  Aggregation
         |  |  |  Column Function(s)
(    9)  |  |  Nested Loop Join
         |  |  |  Piped Inner
(   12)  |  |  |  Unnest input XML sequence into stream of items with item number
(    8)  |  |  Nested Loop Join
         |  |  |  Piped Inner
(   14)  |  |  |  Table Constructor
         |  |  |  |  1-Row(s)
(    2) Nested Loop Join
         |  Piped Inner
(   16)  |  Unnest input XML sequence into stream of items
(    1) Iterate over XML sequence for Xquery bindout
(    1) Return Data to Application
         |  #Columns = 1
```

```
End of section

Optimizer Plan:

               RETURN
               (   1)
                 |
               NLJOIN
               (   2)
             /-/      \-----\
         NLJOIN             *
         (   3)             *
       /        \           |
    TBSCAN       NLJOIN     TBSCAN
    (   4)       (   5)     (  16)
     |           /     \       |
   Table:      XSCAN   *      TFunc:
   ATTALURI    (   6)  *      SYSIBM
   XISCANTABLE         |      GENROW
                     NLJOIN
                     (   8)
                    /      \
               NLJOIN     FILTER
              (   9)      (  13)
             /      \       |
          GRPBY   TBSCAN  TBSCAN
          (  10)  (  12)  (  14)
            |       |       |
          XSCAN   TFunc:  TFunc:
          (  11)  SYSIBM  SYSIBM
                  GENROW  GENROW
```

Beispiel 8: Operator XISCAN: Im vorliegenden Beispiel wird gezeigt, wie der Operator XISCAN den Index für XML-Daten *IDX1* durchsucht, der für die Tabelle *XISCANTABLE* definiert wurde.

```
IBM DB2 Database SQL Explain Tool

******************** DYNAMIC ************************************

==================== STATEMENT ============================================

        Isolation Level          = Cursor Stability
        Blocking                 = Block Unambiguous Cursors
        Query Optimization Class = 5

        Partition Parallel       = No
        Intra-Partition Parallel = No

        SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "ATTALURI"


Query Statement:
  xquery
  for $c in db2-fn:xmlcolumn("XISCANTABLE.XMLCOL ")/a[@x="1" ] return
          $c


Section Code Page = 819

Estimated Cost = 1666.833862
Estimated Cardinality = 18.000000

(    6) Access Table Name = ATTALURI.XISCANTABLE
         |  Index Scan over XML:  Name = ATTALURI.IDX1  ID = 4
         |  |  Physical Index over XML
         |  |  Index Columns:
```

```
        |  |  |  1: XMLCOL (Ascending)
        |  #Key Columns = 2
        |  |  Start Key: Inclusive Value
        |  |  |  1: ?
        |  |  |  2: ?
        |  |  Stop Key: Inclusive Value
        |  |  |  1: ?
        |  |  |  2: ?
        |  Index-Only Access
        |  Index Prefetch: None
        |  Isolation Level: Uncommitted Read
        |  Lock Intents
        |  |  Table: Intent None
        |  |  Row  : None
        |  StopKey = StartKey
        |  Value Start Key = ?
        |  Xpath is
        |  |  /child::element(a)/attribute::attribute(x)
(    5) Insert Into Sorted Temp Table  ID = t1
        |  #Columns = 1
        |  #Sort Key Columns = 1
        |  |  Key 1: (Ascending)
        |  Sortheap Allocation Parameters:
        |  |  #Rows     = 18
        |  |  Row Width = 16
        |  Piped
        |  Duplicate Elimination
(    4) List Prefetch Preparation
(    4) |  Access Table Name = ATTALURI.XISCANTABLE  ID = 2,16
        |  |  #Columns = 1
        |  |  Fetch Using Prefetched List
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    2) Nested Loop Join
        |  Piped Inner
(    7) |  XML Doc Navigation
        |  |  Navigator is
        |  |  |  /fn:root($CONTEXT_NODE$())/child::element(a)(:Output nodeSeqRef :)
        |  |  |  (:#Xpath Predicates = 1:)
        |  |  |  /attribute::attribute(x)
(    1) Iterate over XML sequence for Xquery bindout
(    1) Return Data to Application
        |  #Columns = 1

End of section


Optimizer Plan:

                    RETURN
                    (   1)
                      |
                    NLJOIN
                    (   2)
                   /      \
                FETCH     XSCAN
                (----)    (   7)
               /      \
            RIDSCN     Table:
            (   4)     ATTALURI
              |        XISCANTABLE
            SORT
            (   5)
              |
            XISCAN
```

```
        (   6)
       /      \
Index:        Table:
ATTALURI      ATTALURI
IDX1          XISCANTABLE
```

EXPLAIN-Tabellen und Organisation von EXPLAIN-Informationen

Sämtliche EXPLAIN-Informationen sind nach dem Konzept einer EXPLAIN-Instanz organisiert. Eine EXPLAIN-Instanz stellt einen Aufruf der EXPLAIN-Einrichtung für eine oder mehrere SQL- oder XQuery-Anweisungen dar. Die EXPLAIN-Informationen, die in einer EXPLAIN-Instanz erfasst werden, umfassen die Umgebung der Kompilierung und den zur Ausführung der SQL- oder XQuery-Anweisung, die kompiliert wird, ausgewählten Zugriffsplan. Zum Beispiel kann eine EXPLAIN-Instanz aus jeder der folgenden Informationsgruppen bestehen:

- Alle in Frage kommenden SQL- oder XQuery-Anweisungen in einem Paket für statische Abfrageanweisungen. Bei SQL-Anweisungen (einschließlich derer, die XML-Daten abfragen) können Sie EXPLAIN-Informationen für folgende Anweisungen erfassen: (dynamische) Compound SQL-Anweisungen, CALL, DELETE, INSERT, MERGE, REFRESH, SELECT, SET INTEGRITY, SELECT INTO, UPDATE, VALUES und VALUES INTO. Bei XQuery-Anweisungen können Sie EXPLAIN-Informationen für Anweisungen mit XQUERY db2-fn:xmlcolumn und XQUERY db2-fn:sqlquery abrufen.

  Anmerkung: Anweisungen REFRESH TABLE und SET INTEGRITY werden nicht statisch, sondern nur dynamisch kompiliert.
- Eine bestimmte SQL-Anweisung für SQL-Anweisungen zum inkrementellen Binden (Incremental Bind).
- Eine bestimmte SQL-Anweisung für dynamische SQL-Anweisungen.
- Jede SQL-Anweisung EXPLAIN (dynamisch oder statisch).

Die Informationen der EXPLAIN-Tabellen spiegeln die Beziehungen zwischen Operatoren und Datenobjekten im Zugriffsplan wieder. Die folgende Abbildung zeigt die Beziehungen zwischen diesen Tabellen.

EXPLAIN-Informationen werden in den folgenden Tabellen gespeichert:

Tabelle 66. Relationale Tabellen zur Speicherung von EXPLAIN-Daten

Tabellenname	Beschreibung
EXPLAIN_ARGUMENT	Enthält die spezifischen Merkmale für jeden einzelnen Operator (sofern vorhanden).
EXPLAIN_INSTANCE	Die Hauptsteuertabelle für alle EXPLAIN-Informationen. Jede Datenzeile in den EXPLAIN-Tabellen ist explizit mit einer eindeutigen Zeile in dieser Tabelle verbunden. In dieser Tabelle werden grundlegende Informationen über die Quelle der SQL- oder XQuery-Anweisungen, die mit EXPLAIN bearbeitet werden, und Umgebungsinformationen gespeichert.
EXPLAIN_OBJECT	Identifiziert die Datenobjekte, die für den Zugriffsplan erforderlich sind, der zur Erfüllung der SQL- oder XQuery-Anweisung generiert wurde.
EXPLAIN_OPERATOR	Enthält alle Operatoren, die zur Erfüllung der SQL- oder XQuery-Anweisung durch den Abfragecompiler erforderlich sind.
EXPLAIN_PREDICATE	Enthält Informationen über die Vergleichselemente, die von einem bestimmten Operator angewandt werden.

Tabelle 66. Relationale Tabellen zur Speicherung von EXPLAIN-Daten (Forts.)

Tabellenname	Beschreibung
EXPLAIN_STATEMENT	Enthält den Text der SQL- oder XQuery-Anweisung, wie er für die verschiedenen Stufen der EXPLAIN-Informationen vorhanden ist. Die ursprüngliche, vom Benutzer eingegebene SQL- oder XQuery-Anweisung wird in dieser Tabelle mit der vom Optimierungsprogramm zum Auswählen eines Zugriffsplans verwendeten Version gespeichert. Wenn eine EXPLAIN-Momentaufnahme angefordert wird, werden zusätzliche EXPLAIN-Informationen aufgezeichnet, um den Zugriffsplan zu beschreiben, der vom Abfrageoptimierungsprogramm ausgewählt wurde. Diese Informationen werden in der Spalte SHAPSHOT der Tabelle EXPLAIN_STATEMENT in dem für Visual Explain erforderlichen Format gespeichert. Dieses Format kann von anderen Anwendungen nicht verwendet werden.
EXPLAIN_STREAM	Stellt die Eingabe- und Ausgabedatenströme zwischen einzelnen Operatoren und Datenobjekten dar. Die Datenobjekte selbst werden in der Tabelle EXPLAIN_OBJECT dargestellt. Die an einem Datenstrom beteiligten Operatoren werden in der Tabelle EXPLAIN_OPERATOR dargestellt.
EXPLAIN_DIAGNOSTIC	Enthält einen Eintrag für jede einzelne Diagnosenachricht, die für eine bestimmte Instanz einer EXPLAIN-Anweisung in der Tabelle EXPLAIN_STATEMENT erstellt wird.
EXPLAIN_DIAGNOSTIC_DATA	Enthält Nachrichtentokens für bestimmte Diagnosenachrichten, die in der Tabelle EXPLAIN_DIAGNOSTIC aufgezeichnet werden. Die Nachrichtentokens enthalten weitere Informationen, die sich auf die Ausführung der durch die Nachricht generierten SQL-Anweisung beziehen.
ADVISE_WORKLOAD	Ermöglicht Benutzern, eine Auslastung für die Datenbank zu beschreiben. Jede Zeile in der Tabelle stellt eine SQL- oder XQuery-Anweisung in der Auslastung dar und wird durch eine zugeordnete Häufigkeit beschrieben. Das Tool `db2advis` verwendet diese Tabelle, um Auslastungsdaten zu erfassen und zu speichern.
ADVISE_INSTANCE	Enthält Informationen zur Ausführung von `db2advis`. Diese Informationen beinhalten auch die Startzeit. Für jede Ausführung von `db2advis` wird eine Zeile eingefügt.
ADVISE_INDEX	Speichert Informationen über empfohlene Indizes. Die Tabelle kann durch den Abfragecompiler, das Dienstprogramm `db2advis` oder einen Benutzer mit Daten gefüllt werden. Diese Tabelle wird zu zwei Zwecken verwendet: • Zum Abrufen empfohlener Indizes • Zum Bewerten von Indizes auf der Grundlage von Eingaben zu vorgeschlagenen Indizes
ADVISE_MQT	Enthält die CREATE-DDL-Anweisungen, die Abfrage, die die einzelnen MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle) definiert, die Statistikdaten für die MQTs wie beispielsweise COLSTATS (pro Spalte) im XML-Format, NUMROWS usw. sowie die Stichprobenabfrage zum Abrufen von Stichprobenstatistiken für die MQT.
ADVISE_TABLE	Speichert die DDL zur Tabellenerstellung mit den endgültigen Empfehlungen des Designadvisors für empfohlene MQTs, MDC-Tabellen und Datenbankpartitionen in Abhängigkeit von den angegebenen Optionen und den generierten Empfehlungen.
ADVISE_PARTITION	Speichert von `db2advis` generierte und bewertete virtuelle Datenbankpartitionen.

Anmerkung: Nicht alle der oben genannten Tabellen werden standardmäßig erstellt. Zur Erstellung der Tabellen führen Sie das Script EXPLAIN.DDL aus, das sich im Unterverzeichnis *misc* des Unterverzeichnisses *sqllib* befindet.

In DB2 Version 9.5 können Sie jetzt EXPLAIN-Tabellen mithilfe der Prozedur SYSPROC.SYSINSTALLOBJECTS erstellen, löschen und prüfen. Diese Prozedur ermöglicht eine Erstellung der EXPLAIN-Tabellen unter einem bestimmten Schema und in einem bestimmten Tabellenbereich. Ein Beispiel finden Sie in der Datei EXPLAIN.DLL.

EXPLAIN-Tabellen können für mehrere Benutzer gemeinsame Daten enthalten. Jedoch können die EXPLAIN-Tabellen auch für nur einen Benutzer definiert werden. Anschließend können Aliasnamen für jeden weiteren Benutzer definiert werden, der den gleichen Namen verwendet, um auf die definierten Tabellen zu verweisen. Die EXPLAIN-Tabellen können auch unter dem Schema SYSTOOLS definiert werden. Die EXPLAIN-Funktion verwendet standardmäßig das Schema SYSTOOLS, wenn keine anderen EXPLAIN-Tabellen oder -Aliasnamen unter der Sitzungs-ID des Benutzers für dynamische SQL- oder XQuery-Anweisungen oder unter der Berechtigungs-ID der Anweisung für statische SQL- oder XQuery-Anweisungen gefunden werden. Jeder Benutzer, der auf die gemeinsamen EXPLAIN-Tabellen zugreift, muss das Zugriffsrecht INSERT zum Einfügen für diese Tabellen aufweisen. Der Lesezugriff für die allgemeinen EXPLAIN-Tabellen muss ebenfalls eingeschränkt werden, speziell für Benutzer, die die EXPLAIN-Informationen analysieren.

EXPLAIN-Informationen für Datenobjekte

Ein einziger Zugriffsplan kann zur Erfüllung der SQL- oder XQuery-Anweisung ein oder mehrere Datenobjekte verwenden.

Objektstatistiken: Die EXPLAIN-Einrichtung zeichnet Informationen über das Objekt auf, wie z. B. folgende Angaben:

- Die Erstellungszeit
- Den Zeitpunkt, an dem zum letzten Mal Statistiken für das Objekt gesammelt wurden
- Eine Angabe, ob die Daten im Objekt sortiert sind (nur Tabellen- oder Indexobjekte)
- Die Anzahl von Spalten im Objekt (nur Tabellen- oder Indexobjekte)
- Die geschätzte Anzahl von Zeilen im Objekt (nur Tabellen- oder Indexobjekte)
- Die Anzahl von Seiten, die das Objekt im Pufferpool einnimmt
- Den geschätzten Gesamtaufwand in Millisekunden für eine einzelne wahlfreie Ein-/Ausgabeoperation für den angegebenen Tabellenbereich, in dem dieses Objekt gespeichert ist
- Die geschätzte Übertragungsrate in Millisekunden zum Lesen einer 4-KB-Seite aus dem angegebenen Tabellenbereich
- Die Größen für PREFETCHSIZE und EXTENTSIZE (in 4-KB-Seiten)
- Den Grad der Datenclusterbildung in Bezug auf den Index
- Die Anzahl von Blattseiten (Leaf Pages), die vom Index für dieses Objekt verwendet werden, und die Anzahl der Indexstufen in der Indexbaumstruktur
- Die Anzahl unterschiedlicher vollständiger Schlüsselwerte im Index für dieses Objekt
- Die Gesamtzahl der Überlaufsätze in der Tabelle

EXPLAIN-Informationen für Datenoperatoren

Ein einzelner Zugriffsplan kann mehrere Operationen an den Daten ausführen, um eine SQL- bzw. XQuery-Anweisung zu erfüllen und die Ergebnisse an Sie zurückzugeben. Der Abfragecompiler bestimmt die erforderlichen Operationen, wie zum Beispiel eine Tabellensuche, eine Indexsuche, einen Join mit Verschachtelungsschleife (Nested Loop Join) oder einen Gruppierungsoperator (GROUP BY).

Neben den Operatoren, die in einem Zugriffsplan verwendet werden, sowie den Informationen zu jedem Operator enthalten EXPLAIN-Informationen auch Angaben zu den kumulativen Effekten des Zugriffsplans.

Informationen zum geschätzten Aufwand: Die folgenden Angaben zum geschätzten kumulativen Aufwand können für die Operatoren angezeigt werden. Diese Angaben beziehen sich auf den ausgewählten Zugriffsplan bis einschließlich zu dem Operator, für den die Informationen erfasst werden.

- Der Gesamtaufwand (in Timerons)
- Die Anzahl der Seiten-E/A-Operationen
- Die Anzahl der CPU-Instruktionen
- Der Aufwand (in Timerons) für den Abruf der ersten Zeile, einschließlich des erforderlichen Anfangsaufwands
- Der Übertragungsaufwand (in Rahmen (Frames))

Die Einheit *Timeron* ist eine relative Maßeinheit. Timerons werden durch das Optimierungsprogramm auf der Basis interner Werte ermittelt, wie zum Beispiel Statistiken, die sich im Lauf der Datenbankverwendung ändern. Infolgedessen ist der Messwert in Timerons für eine SQL- bzw. XQuery-Anweisung nicht unbedingt jedes Mal gleich, wenn der geschätzte Aufwand in Timerons ermittelt wird.

Operatormerkmale: Die folgenden Informationen werden durch die EXPLAIN-Einrichtung erfasst, um die Merkmale der einzelnen Operatoren zu beschreiben:

- Die Gruppe von Tabellen, auf die zugegriffen wurde
- Die Gruppen von Spalten, auf die zugegriffen wurde
- Die Spalten, nach denen die Daten angeordnet werden, wenn das Optimierungsprogramm feststellt, dass diese Reihenfolge von nachfolgenden Operatoren genutzt werden kann
- Die Gruppe von Vergleichselementen, die angewendet wurden
- Die geschätzte Anzahl von Zeilen, die zurückgegeben werden (Kardinalität)

EXPLAIN-Informationen für Instanzen

Informationen über EXPLAIN-Instanzen werden in der Tabelle EXPLAIN_INSTANCE gespeichert. Zusätzliche bestimmte Informationen zu jeder Abfrageanweisung in einer Instanz werden in der Tabelle EXPLAIN_STATEMENT gespeichert.

Kennzeichnung von EXPLAIN-Instanzen: Anhand der in den folgenden Punkten bereitgestellten Informationen können Sie jede EXPLAIN-Instanz eindeutig identifizieren und die Informationen für Abfrageanweisungen einem bestimmten Aufruf der Einrichtung zuordnen:

- Der Benutzer, der die EXPLAIN-Informationen anforderte
- Der Zeitpunkt, zu dem die EXPLAIN-Anforderung begann
- Der Name des Pakets, das die mit EXPLAIN bearbeitete Abfrageanweisung enthält

- Das SQL-Schema des Pakets, das die mit EXPLAIN bearbeitete Abfrageanweisung enthält
- Die Version des Pakets, das die Anweisung enthält
- Eine Angabe, ob Momentaufnahmendaten (Snapshot) erfasst wurden

Einstellungen der Umgebung: Es werden Informationen zur Umgebung des Datenbankmanagers erfasst, in der der Abfragecompiler die Abfragen optimiert hat. Zu den Umgebungsinformationen gehören die folgenden:

- Die Versions- und Releasenummer für die DB2-Stufe
- Der Grad der Parallelität, für den die Abfrage kompiliert wurde

  Das Sonderregister CURRENT DEGREE, die Bindeoption DEGREE, die API SET RUNTIME DEGREE und der Konfigurationsparameter *dft_degree* bestimmen den Grad der Parallelität, für den eine bestimmte Abfrage kompiliert wird.
- Ob die Abfrageanweisung dynamisch oder statisch ist
- Die zum Kompilieren der Abfrage verwendete Optimierungsklasse
- Der Typ der Zeilenblockung für Cursor, der beim Kompilieren der Abfrage angegeben wurde
- Die Isolationsstufe, in der die Abfrage ausgeführt wird
- Die Werte verschiedener Konfigurationsparameter zum Zeitpunkt der Kompilierung der Abfrage. Die folgenden Parameter werden bei der Erfassung einer EXPLAIN-Momentaufnahme (Snapshot) aufgezeichnet:
  - Zwischenspeicher für Sortierlisten (*sortheap*)
  - Durchschnittliche Anzahl aktiver Anwendungen (*avg_appls*)
  - Datenbankzwischenspeicher (*dbheap*)
  - Maximaler Speicher für Sperrenliste (*locklist*)
  - Maximale Anzahl Sperren pro Anwendung (*maxlocks*)
  - CPU-Geschwindigkeit (*cpuspeed*)
  - Kommunikationsbandbreite (*comm_bandwidth*)

Kennzeichnung von Anweisungen: Für jede EXPLAIN-Instanz können mehrere Abfrageanweisungen mit EXPLAIN bearbeitet werden. Zusätzlich zu den Informationen, die die EXPLAIN-Instanz eindeutig kennzeichnen, kann jede einzelne Abfrageanweisung anhand der folgenden Informationen identifiziert werden:

- Der Typ der Anweisung: SELECT, DELETE, INSERT, UPDATE, positioniertes DELETE, positioniertes UPDATE, SET INTEGRITY
- Die Anweisungs- und Abschnittsnummer des Pakets, das die Abfrageanweisung absetzt, die in der Katalogsicht SYSCAT.STATEMENTS aufgezeichnet wurde

Die Felder QUERYTAG und QUERYNO in der Tabelle EXPLAIN_STATEMENT enthalten Kennungen, die im Rahmen des EXPLAIN-Prozesses mit Werten gefüllt werden. Für dynamische EXPLAIN-Abfrageanweisungen, die in einer CLP- oder CLI-Sitzung übergeben wurden, wird QUERYTAG auf „CLP" bzw. „CLI" gesetzt, wenn EXPLAIN MODE oder EXPLAIN SNAPSHOT aktiv ist. In diesem Fall wird für QUERYNO standardmäßig eine Nummer angegeben, die mindestens um den Wert 1 für jede Anweisung erhöht wird. Für alle anderen dynamischen EXPLAIN-Abfrageanweisungen, die nicht über CLP bzw. CLI angefordert werden oder die EXPLAIN-Abfrageanweisung nicht verwenden, wird das Feld QUERYTAG mit Leerzeichen gefüllt, und QUERYNO wird immer auf „1" gesetzt.

Schätzung des Aufwands: Für jede mit EXPLAIN bearbeitete Anweisung zeichnet das Optimierungsprogramm einen Schätzwert für den relativen Aufwand zur Aus-

führung des ausgewählten Zugriffsplans auf. Dieser Aufwand wird in einer künstlich geschaffenen relativen Maßeinheit mit der Bezeichnung *timeron* angegeben. Schätzwerte für die benötigten Ausführungszeiten werden aus folgenden Gründen nicht bereitgestellt:

- Das Abfrageoptimierungsprogramm schätzt nicht die benötigte Zeit, sondern nur den Ressourcenbedarf ab.
- Das Optimierungsprogramm modelliert nicht alle Faktoren nach, die die benötigte Zeit beeinflussen können. Es ignoriert Faktoren, die keine Auswirkung auf die Effizienz des Zugriffsplans haben. Die benötigte Zeit wird von einer Reihe Laufzeitfaktoren beeinflusst, zu denen die folgenden gehören: die Systemauslastung, die Ressourcenverfügbarkeit, der Umfang der Parallelverarbeitung und der Ein-/Ausgabeoperationen, der Aufwand für die Rückgabe von Zeilen an den Benutzer sowie die Übertragungszeit zwischen Client und Server.

Anweisungstext: Für jede mit EXPLAIN bearbeitete Anweisung werden zwei Versionen des Texts der Abfrageanweisung aufgezeichnet. Eine Version ist der Code, den der Abfragecompiler von der Anwendung empfängt. Die andere Version ist die rückübersetzte Version aus der compilerinternen Darstellung der Abfrage. Obwohl diese Übersetzung anderen Abfrageanweisungen sehr ähnlich ist, entspricht sie nicht unbedingt der richtigen Syntax der Abfragesprache und spiegelt nicht in jedem Fall den tatsächlichen Inhalt der internen Darstellung als Ganzes wider. Diese Übersetzung wird nur zur Verfügung gestellt, um Ihnen einen Einblick in den Kontext zu geben, in dem das SQL- oder XQuery-Optimierungsprogramm den Zugriffsplan ausgewählt hat. Um zu verstehen, wie der SQL- und XQuery-Compiler Ihre Abfrage zur Optimierung umgeschrieben hat, vergleichen Sie den vom Benutzer geschriebenen Anweisungstext mit der internen Darstellung der Abfrageanweisung. Die umgeschriebene Anweisungen zeigt Ihnen außerdem noch weitere Elemente in der Umgebung, die sich auf Ihre Anweisung auswirken, wie zum Beispiel Trigger und Integritätsbedingungen. Einige Schlüsselwörter, die in diesem „optimierten" Text verwendet werden, sind:

$C*n* Der Name einer abgeleiteten Spalte, wobei *n* ein ganzzahliger Wert ist.

$CONSTRAINT$
Diese Kennung weist auf den Namen einer Integritätsbedingung hin, die der ursprünglichen Abfrageanweisung bei der Kompilierung hinzugefügt wurde. Es tritt in Kombination mit dem Präfix $WITH_CONTEXT$ auf.

$DERIVED.T*n*
Der Name einer abgeleiteten Tabelle, wobei *n* ein ganzzahliger Wert ist.

$INTERNAL_FUNC$
Diese Kennung weist auf das Vorhandensein einer Funktion hin, die vom SQL- und XQuery-Compiler für die mit EXPLAIN bearbeitete Abfrage verwendet wurde, jedoch nicht zur allgemeinen Verwendung verfügbar ist.

$INTERNAL_PRED$
Diese Kennung weist auf das Vorhandensein eines Vergleichselements hin, das vom SQL- und XQuery-Compiler bei der Kompilierung der mit EXPLAIN bearbeiteten Abfrage hinzugefügt wurde, jedoch nicht zur allgemeinen Verwendung verfügbar ist. Vom Compiler wird ein internes Vergleichselement verwendet, um den infolge von Triggern und Integritätsbedingungen der ursprünglichen Abfrageanweisung hinzugefügten Zusatzkontext zu erfüllen.

$INTERNAL_XPATH$
Zeigt eine interne Tabellenfunktion an, die als Parameter ein einziges

eingegebenes, mit Anmerkungen versehenes XPath-Muster enthält und eine Tabelle mit mindestens einer Spalte zurückgibt, die mit dem Muster abgeglichen wird.

RID Dieses Kennzeichen dient zur Identifizierung der Spalte der Satz-ID (RID) für eine bestimmte Zeile.

$TRIGGER$
Diese Kennung weist auf den Namen eines Triggers hin, der der ursprünglichen Abfrageanweisung bei der Kompilierung hinzugefügt wurde. Es tritt in Kombination mit dem Präfix $WITH_CONTEXT$ auf.

$WITH_CONTEXT$(...)
Dieses Präfix tritt am Anfang des Texts auf, wenn zusätzliche Trigger oder Integritätsbedingungen in die ursprüngliche Abfrageanweisung eingefügt wurden. Diesem Präfix folgt eine Liste der Namen aller Trigger oder Integritätsbedingungen, die sich auf die Kompilierung und Auflösung der Abfrageanweisung auswirken.

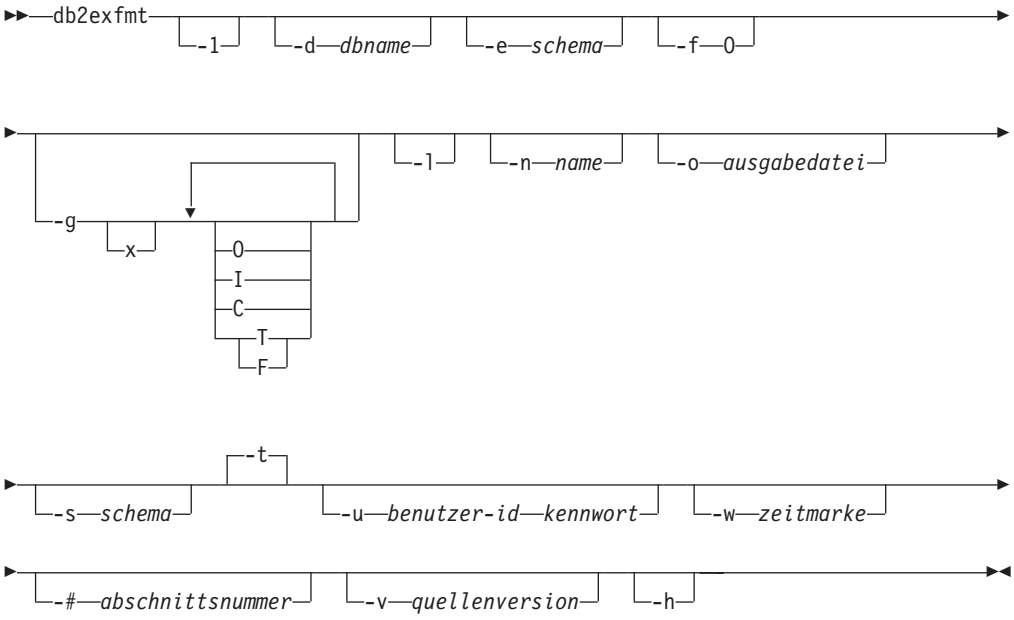
db2exfmt - EXPLAIN-Tabellen formatieren

Mithilfe des Tools db2exfmt können Sie den Inhalt der EXPLAIN-Tabellen formatieren. Dieses Tool befindet sich im Unterverzeichnis `misc` des Verzeichnisses `sqllib` für Ihre Instanz. Dieses Tool arbeitet mit den Statistiken aus der EXPLAIN-Momentaufnahme, wenn die Momentaufnahme verfügbar ist.

Berechtigung

Zur Verwendung des Tools müssen Sie über Lesezugriff auf die EXPLAIN-Tabellen verfügen, die formatiert werden sollen.

Befehlssyntax



Befehlsparameter

db2exfmt
Wenn keine Optionen angegeben werden, wechselt der Befehl in den interaktiven Modus und fordert Sie zu Eingaben auf.

-1 Gibt die Standardwerte `-e % -n % -s % -v % -w -1 -# 0` an.

Wenn das EXPLAIN-Schema nicht angegeben wird, wird der Inhalt der Umgebungsvariablen $USER oder $USERNAME als Standardwert verwendet. Falls diese Variable nicht gefunden wird, wird der Benutzer zur Eingabe eines EXPLAIN-Schemas aufgefordert.

-d *dbname*
Gibt den Namen der Datenbank an, die Pakete enthält.

-e *schema*
Gibt das SQL-Schema für die EXPLAIN-Tabellen an.

-f Gibt Formatmarkierungen an. In diesem Release wird nur der Wert O (Operatorzusammenfassung) unterstützt.

-g Gibt den Diagrammplan an.

x Inaktiviert die Optionen (OFF; standardmäßig werden die Optionen aktiviert).

Wenn nur -g angegeben wird, wird ein Diagramm gefolgt von formatierten Informationen für alle Tabellen generiert. Ansonsten kann eine beliebige Kombination der folgenden gültigen Werte angegeben werden:

O Generiert nur das Diagramm. Tabelleninhalte werden nicht formatiert.

T Fügt den Gesamtaufwand unter jedem Operator im Diagramm ein.

F Fügt den Aufwand für das erste Tupel im Diagramm ein.

I Fügt den Ein-/Ausgabeaufwand unter jedem Operator im Diagramm ein.

C Fügt die erwartete Ausgabekardinalität (Anzahl von Tupeln) jedes Operators im Diagramm ein.

Jede beliebige Kombination aus diesen Optionen ist zulässig. Einzige Ausnahmen sind F und T, die sich gegenseitig ausschließen.

-l Beachtet die Groß-/Kleinschreibung bei der Verarbeitung von Paketnamen.

-n *name*
Gibt den Namen der Quelle der EXPLAIN-Anforderung (SOURCE_NAME) an.

-s *schema*
Gibt das SQL-Schema oder -Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung (SOURCE_SCHEMA) an.

-o *ausgabedatei*
Gibt den Namen der Ausgabedatei an.

-t Leitet die Ausgabe an das Terminal.

-u *benutzer-id kennwort*
Gibt die Benutzer-ID und das Kennwort an, die bei der Herstellung einer Verbindung zu einer Datenbank zu verwenden sind.

Sowohl die Benutzer-ID als auch das Kennwort müssen entsprechend den Namenskonventionen gültig sein und von der Datenbank erkannt werden.

-w *zeitmarke*
Gibt die EXPLAIN-Zeitmarke an. Geben Sie den Wert -1 an, um die aktuellste EXPLAIN-Anforderung abzurufen.

-# *abschnittsnummer*
Gibt die Abschnittsnummer in der Quelle an. Geben Sie 0 an, um alle Abschnitte anzufordern.

-v *quellenversion*
Gibt die Quellenversion der Quelle der EXPLAIN-Anforderung (Standardwert: %) an.

-h Zeigt den Hilfetext an. Bei Angabe dieser Option werden alle anderen Optionen ignoriert und nur der Hilfetext angezeigt.

Hinweise

Sie werden zur Eingabe jedes Parameters aufgefordert, der nicht angegeben bzw. unvollständig angegeben wurde. Davon ausgenommen sind nur die Optionen -h und -l.

Wenn kein SQL-Schema für eine EXPLAIN-Tabelle angegeben wird, wird der Wert der Umgebungsvariablen USER als Standardwert verwendet. Falls diese Variable nicht gefunden wird, wird der Benutzer zur Eingabe eines SQL-Schemas für EXPLAIN-Tabellen aufgefordert.

Quellenname, SQL-Quellenschema und EXPLAIN-Zeitmarke können in Form eines Vergleichselements LIKE angegeben werden, bei dem das Prozentzeichen (%) und das Unterstreichungszeichen (_) als Platzhalterzeichen zur Mustererkennung verwendet werden können, um mehrere Quellen in einem Aufruf auszuwählen. Für die zuletzt ausgeführte Anweisung EXPLAIN kann die EXPLAIN-Zeit durch den Wert -1 angegeben werden.

Wenn -o ohne Dateinamen angegeben wird und -t nicht angegeben wird, wird der Benutzer zur Eingabe eines Dateinamens aufgefordert (Standardname: `db2exfmt.out`). Wenn weder -o noch -t angegeben wird, wird der Benutzer zur Eingabe eines Dateinamens aufgefordert. (Die Standardoption ist die Terminalausgabe). Wenn sowohl -o als auch -t angegeben werden, wird die Ausgabe an das Terminal geleitet.

Der Befehl db2exfmt zeigt die Statistiken aus der EXPLAIN-Momentaufnahme an, wenn die Momentaufnahme verfügbar ist. Ansonsten zeigt der Befehl db2exfmt Statistiken, die in der Tabelle EXPLAIN_OBJECT gespeichert sind, sowie einige Statistiken an, die direkt aus dem Systemkatalog abgerufen werden.

Die folgenden Beispiele zeigen Befehle mit EXPLAIN-Momentaufnahmen.

```
db2 explain plan with snapshot for abfrage
db2exfmt
```

oder

```
db2 set current explain mode yes
db2 set current explain snapshot yes
Ausführen der Abfrage
db2exfmt
```

Optimieren von Abfragezugriffsplänen

Optimierungsklassen

Wenn Sie eine SQL- oder XQuery-Abfrage kompilieren, können Sie eine Optimierungsklasse angeben, die bestimmt, wie das Optimierungsprogramm den effizientesten Zugriffsplan für diese Abfrage auswählt. Die Optimierungsklassen werden nach Anzahl und Typ der Optimierungsstrategien unterschieden, die in der Abfragekompilierung berücksichtigt werden. Obwohl Sie einerseits die Optimierungstechniken einzeln angeben können, um die Laufleistung für die Abfrage zu verbessern, werden andererseits um so mehr Zeit und Systemressourcen für die Abfragekompilierung benötigt, je mehr Optimierungstechniken Sie angeben.

Bei der Kompilierung einer SQL- oder XQuery-Abfrage können Sie eine der folgenden Optimierungsklassen angeben:

0 - Diese Klasse weist das Optimierungsprogramm an, nur eine Minimaloptimierung zur Generierung eines Zugriffsplans durchzuführen. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Statistiken über ungleichmäßige Verteilungen von Datenwerten werden vom Optimierungsprogramm nicht berücksichtigt.
- Nur Grundregeln für das Umschreiben der Abfragen werden angewandt.
- Schnelle Joinaufzählung findet statt.
- Nur die Zugriffsmethoden durch Joins mit Verschachtelungsschleife und Indexsuchen sind möglich.
- Ein Vorablesezugriff wird in den generierten Zugriffsmethoden nicht verwendet.
- Die Strategie des Sternjoins wird nicht berücksichtigt.

Diese Klasse sollte nur unter Umständen verwendet werden, unter denen der Systemaufwand zur Kompilierung der Abfrage so gering wie möglich gehalten werden muss. Die Abfrageoptimierungsklasse 0 eignet sich für eine Anwendung, die insgesamt aus sehr einfachen dynamischen SQL- oder XQuery-Anweisungen besteht, die auf korrekt indexierte Tabellen zugreifen.

1 - Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Statistiken über ungleichmäßige Verteilungen von Datenwerten werden vom Optimierungsprogramm nicht berücksichtigt.
- Nur eine Teilmenge der Regeln für das Umschreiben von Abfragen wird angewandt.
- Schnelle Joinaufzählung findet statt.
- Ein Vorablesezugriff wird in den generierten Zugriffsmethoden nicht verwendet.

Die Optimierungsklasse 1 ist der Klasse 0 ähnlich, abgesehen davon, dass Mischjoins und Tabellensuchen ebenfalls verfügbar sind.

2 - Diese Klasse weist das Optimierungsprogramm an, einen Optimierungsgrad zu verwenden, der den der Klasse 1 deutlich übertrifft, und gleichzeitig den Kompilierungsaufwand für komplexe Abfragen wesentlich geringer als bei den Klassen ab 3 aufwärts zu halten. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Alle verfügbaren Statistiken, einschließlich der Statistiken zur Häufigkeit und zu Quantilen ungleichmäßiger Verteilungen, werden verwendet.
- Alle Regeln für das Umschreiben von Abfragen, einschließlich der Weiterleitung von Abfragen an MQTs (MQT = Materialized Query Table, gespeicherte Abfragetabelle), werden berücksichtigt, außer den Regeln, die sehr rechenintensiv sind und nur in seltenen Fällen zur Anwendung kommen.
- Schnelle Joinaufzählung wird verwendet.
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen und der Weiterleitung an MQTs.
- Die Strategie des Sternjoins wird gegebenenfalls berücksichtigt.

Die Optimierungsklasse 2 ist der Klasse 5 ähnlich, sie verwendet jedoch schnelle Joinaufzählung und nicht dynamische programmierte Joinaufzählung. Diese Klasse hat den höchsten Optimierungsgrad aller Klassen, die mit dem Algorithmus für schnelle Joinaufzählung arbeiten, der für komplexe Abfragen weniger Alternativen berücksichtigt und dadurch einen geringeren Kompilierungsaufwand erfordert als die Klassen ab 3 aufwärts. Klasse 2 empfiehlt sich für sehr komplexe Abfragen in einer Umgebung zur Entscheidungshilfe oder mit analytischer Onlineverarbeitung (OLAP). In solchen Umgebungen werden spezifische Abfragen nur selten exakt wiederholt, sodass ein Zugriffsplan mit hoher Wahrscheinlichkeit nicht bis zur nächsten Ausführung der Abfrage im Cache erhalten bleibt.

3 - Diese Klasse fordert einen moderaten Optimierungsrad an. Diese Klasse kommt den Merkmalen der Abfrageoptimierung von DB2 für MVS/ESA, OS/390 oder z/OS am nächsten. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Statistiken über ungleichmäßige Verteilungen von Datenwerten, die die Häufigkeit von Werten erfassen, werden verwendet, wenn sie verfügbar sind.
- Die meisten Regeln zum Umschreiben von Abfragen, einschließlich der Umsetzungen von Unterabfragen in Joins, werden angewandt.
- Dynamisch programmierte Joinaufzählung (Dynamic Programming Join Enumeration) wie folgt:
  - Eingeschränkte Verwendung zusammengesetzter innerer Tabellen
  - Eingeschränkte Verwendung kartesischer Produkte für Sternschemata, für die Suchtabellen erforderlich sind
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen, logischer Joins von Indizes über AND (Index ANDing) und Sternjoins.

Diese Klasse eignet sich für eine große Bandbreite von Anwendungen. Diese Klasse verbessert Zugriffspläne für Abfragen mit vier und mehr Joins. Es ist jedoch möglich, dass der vom Optimierungsprogramm gewählte Plan nicht so gut ist wie einer, der mit der Standardoptimierungsklasse gewählt würde.

5 - Diese Klasse weist das Optimierungsprogramm an, einen bedeutenden Grad an Optimierung bei der Generierung eines Zugriffsplans durchzuführen. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Alle verfügbaren Statistiken, einschließlich der Verteilungsstatistiken zur Häufigkeit und zu Quantilen, werden verwendet.

- Alle Regeln für das Umschreiben von Abfragen, einschließlich der Weiterleitung von Abfragen an MQTs, werden berücksichtigt, außer den Regeln, die sehr rechenintensiv sind und nur in seltenen Fällen zur Anwendung kommen.
- Dynamisch programmierte Joinaufzählung (Dynamic Programming Join Enumeration) wie folgt:
  - Eingeschränkte Verwendung zusammengesetzter innerer Tabellen
  - Eingeschränkte Verwendung kartesischer Produkte für Sternschemata, für die Suchtabellen erforderlich sind
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen, logischer Joins von Indizes über AND und der Weiterleitung an MQTs.

Stellt das Optimierungsprogramm fest, dass die zusätzlichen Ressourcen und die Verarbeitungszeit für komplexe dynamische SQL- oder XQuery-Abfragen nicht gewährleistet sind, wird die Optimierung reduziert. Das Ausmaß oder der Umfang der Reduzierung hängt von der Maschinengröße und der Anzahl der Vergleichselemente ab.

Reduziert das Abfrageoptimierungsprogramm den Grad an Abfrageoptimierung, wendet es weiterhin alle Regeln für das Umschreiben von Abfragen an, die normalerweise angewandt würden. Es verwendet jedoch die schnelle Joinaufzählung und reduziert die Anzahl der Zugriffsplankombinationen, die in Erwägung gezogen werden.

Die Abfrageoptimierungsklasse 5 ist hervorragend für eine gemischte Umgebung geeignet, in der sowohl Transaktionen als auch komplexe Abfragen ausgeführt werden. Diese Optimierungsklasse wurde zur Verwendung der wertvollsten Abfragetransformationen und anderer Optimierungstechniken für Abfragen in einer effizienten Weise entwickelt.

7 - Diese Klasse weist das Optimierungsprogramm an, einen bedeutenden Grad an Optimierung bei der Generierung eines Zugriffsplans durchzuführen. Sie entspricht der Abfrageoptimierungsklasse 5, jedoch reduziert sie nicht den Umfang der Abfrageoptimierung für komplexe dynamische SQL- oder XQuery-Abfragen.

9 - Diese Klasse weist das Optimierungsprogramm an, alle verfügbaren Optimierungstechniken anzuwenden. Dazu gehören:

- Alle verfügbaren Statistiken
- Alle Regeln für das Umschreiben von Abfragen
- Alle Möglichkeiten für Joinaufzählungen, einschließlich kartesischer Produkte und uneingeschränkter zusammengesetzter innerer Tabellen
- Alle Zugriffsmethoden

Diese Klasse kann die Anzahl der möglichen Zugriffspläne, die vom Optimierungsprogramm ausgewertet werden, erheblich vergrößern. Diese Klasse kann verwendet werden, um festzustellen, ob eine umfassendere Optimierung zur Generierung eines besseren Zugriffsplans für sehr komplexe und zeitintensive Abfragen auf große Tabellen führen würde. Überprüfen Sie anhand von EXPLAIN-Daten und Leistungswerten, ob ein besserer Plan gefunden wurde.

Auswählen von Optimierungsklassen

Die Einstellung der Optimierungsklasse kann, insbesondere zu folgenden Zwecken, einige der Vorteile bringen, die durch die explizite Angabe von Optimierungstechniken erzielt werden:

- Verwalten sehr kleiner Datenbanken oder sehr einfacher dynamischer Abfragen
- Kompilieren mit beschränkten Speicherkapazitäten auf dem Datenbankserver
- Verkürzen der Abfragekompilierzeit, zum Beispiel für PREPARE

Die meisten Anweisungen lassen sich in angemessener Weise bei einem sinnvollen Einsatz von Ressourcen unter Verwendung der Standardoptimierungsklasse 5 optimieren. Bei einer bestimmten Optimierungsklasse werden die Kompilierzeit und der Bedarf an Ressourcen in erster Linie durch die Komplexität der Abfrage, insbesondere durch die Anzahl der Joins und Unterabfragen, beeinflusst. Allerdings werden die Kompilierzeit und der Ressourcenbedarf auch vom Grad der durchgeführten Optimierung beeinflusst.

Die Abfrageoptimierungsklassen 1, 2, 3, 5 und 7 sind alle für allgemeine Zwecke geeignet. Ziehen Sie die Klasse 0 nur in Betracht, wenn Sie die Kompilierzeit von Abfragen weiter verringern müssen und wissen, dass die SQL- und XQuery-Anweisungen extrem einfach sind.

Tipp: Zur Analyse von Abfragen mit langer Laufzeit können Sie die Abfragen mit db2batch ausführen, um herauszufinden, wie viel Zeit auf die Kompilierung und wie viel Zeit auf die Ausführung verwendet wird. Wenn die Kompilierung mehr Zeit erfordert, verringern Sie die Optimierungsklasse. Wenn die Ausführung mehr Zeit erfordert, ziehen Sie eine höhere Optimierungsklasse in Betracht.

Beachten Sie bei der Auswahl einer Optimierungsklasse die folgenden allgemeinen Richtlinien:

- Beginnen Sie mit der Standardoptimierungsklasse für Abfragen: Klasse 5.
- Zur Verwendung einer anderen als die Standardklasse, versuchen Sie zunächst die Klasse 1, 2 oder 3. Die Klassen 0, 1 und 2 arbeiten mit dem Algorithmus der schnellen Joinaufzählung (Greedy Join Enumeration).
- Verwenden Sie die Optimierungsklasse 1 oder 2, wenn Sie viele Tabellen mit vielen Joinvergleichselementen für dieselbe Spalte haben und die Dauer der Kompilierung von Bedeutung ist.
- Verwenden Sie eine niedrige Optimierungsklasse (0 oder 1) für Abfragen, die sehr kurze Laufzeiten von unter einer Sekunde haben. Solche Abfragen sind häufig durch folgende Merkmale gekennzeichnet:
  - Zugriff auf eine oder nur einige wenige Tabellen
  - Abruf nur einer oder einiger weniger Zeilen
  - Verwendung vollständig qualifizierter eindeutiger Indizes

OLTP-Transaktionen (Online-Transaktionsverarbeitung) sind gute Beispiele für diese Art von Abfrage.

- Verwenden Sie eine höhere Optimierungsklasse (3, 5 oder 7) für Abfragen mit längerer Laufzeit, die mehr als 30 Sekunden benötigen.
- Klasse 3 und höhere Klassen arbeiten mit dem Algorithmus zur dynamisch programmierten Joinaufzählung (Dynamic Programming Join Enumeration). Dieser Algorithmus berücksichtigt wesentlich mehr alternative Pläne und kann, insbesondere bei steigender Tabellenzahl, deutlich mehr Kompilierungszeit als die Klassen 0, 1 und 2 erfordern.
- Verwenden Sie die Optimierungsklasse 9 nur, wenn Sie spezielle, über das normale Maß hinausgehende Optimierungsanforderungen für eine Abfrage haben.

Für komplexe Abfragen können andere Grade an Optimierung erforderlich sein, um den besten Zugriffsplan auszuwählen. Ziehen Sie höhere Optimierungsklassen für Abfragen mit den folgenden Merkmalen in Betracht:

- Zugriff auf große Tabellen
- Eine große Anzahl von Vergleichselementen
- Zahlreiche Unterabfragen
- Zahlreiche Joins
- Zahlreiche Mengenoperatoren wie UNION und INTERSECT
- Zahlreiche die Vergleichselemente erfüllende Zeilen
- Operationen GROUP BY und HAVING
- Verschachtelte Tabellenausdrücke
- Eine große Anzahl von Sichten

Abfragen zur Entscheidungshilfe oder Abfragen für Monatsberichte aus vollständig normalisierten Datenbanken sind gute Beispiele für komplexe Abfragen, für die zumindest die Standardoptimierungsklasse verwendet werden sollte.

Verwenden Sie höhere Optimierungsklassen für SQL- und XQuery-Anweisungen, die von einem Abfragegenerator erstellt wurden. Viele Abfragegeneratoren erstellen ineffiziente Abfragen. Ineffizient geschriebene Abfragen, einschließlich der von einem Abfragegenerator erstellten, können eine zusätzliche Optimierung erforderlich machen, um einen guten Zugriffsplan auszuwählen. Durch Verwendung der Abfrageoptimierungsklasse 2 oder einer höheren Abfrageoptimierungsklasse können solche SQL- und XQuery-Abfragen verbessert werden.

Anmerkung: In einer Abfrage für föderierte Datenbanken gilt die Optimierungsklasse nicht für das ferne Optimierungsprogramm.

Einstellen der Optimierungsklasse

Berücksichtigen Sie bei der Angabe einer Optimierungsstufe, ob eine Abfrage statische oder dynamische SQL- und XQuery-Anweisungen verwendet und ob die gleiche dynamische Abfrage wiederholt ausgeführt wird. Für statische SQL- und XQuery-Anweisungen werden die Kompilierzeit für Abfragen und die Ressourcen nur einmal aufgewendet, und der ausgegebene Plan kann mehrfach verwendet werden. Im Allgemeinen gilt, dass für SQL- und XQuery-Anweisungen stets die Standardoptimierungsklasse verwendet werden sollte. Da dynamische Anweisungen bei der Ausführung gebunden und ausgeführt werden, müssen Sie überlegen, ob der Aufwand für eine zusätzliche Optimierung der dynamischen Anweisungen die allgemeine Leistung verbessert. Wenn allerdings dieselbe dynamische SQL- oder XQuery-Anweisung wiederholt ausgeführt wird, wird der ausgewählte Zugriffsplan im Cache zwischengespeichert. Solche Anweisungen können dieselben Optimierungsstufen wie statische SQL- und XQuery-Anweisungen verwenden.

Wenn Sie annehmen, dass für eine Abfrage eine weitere Optimierung von Vorteil wäre, Sie aber nicht sicher sind oder Bedenken hinsichtlich der Kompilierzeit oder des Ressourcenbedarfs haben, können Sie einige Vergleichstests (Benchmarktests) durchführen.

Führen Sie folgende Schritte aus, um eine Abfrageoptimierungsklasse anzugeben:

1. Analysieren Sie die Leistungsfaktoren entweder informell oder wie folgt mit formalen Tests:

- Für **dynamische** Abfrageanweisungen sollte bei Tests die durchschnittliche Laufzeit für die Anweisung verglichen werden. Schätzen Sie eine durchschnittliche Laufzeit mithilfe folgender Formel ab:

  ```
  Kompilierzeit + Summe der Ausführungszeiten aller Iterationen
  -------------------------------------------------------------
                        Anzahl der Iterationen
  ```

  In dieser Formel entspricht die `Anzahl der Iterationen` der Häufigkeit, mit der die Abfrageanweisung Ihrer Schätzung nach jedes Mal, wenn sie kompiliert wird, ausgeführt werden könnte.

  Anmerkung: Nach der Erstkompilierung werden dynamische SQL- und XQuery-Anweisungen erneut kompiliert, wenn eine Änderung an der Umgebung dies erfordert. Wenn sich die Umgebung nicht ändert, nachdem eine Abfrageanweisung im Cache zwischengespeichert wurde, muss sie nicht neu kompiliert zu werden, da nachfolgende PREPARE-Anweisungen die zwischengespeicherte Anweisung wiederverwenden.

- Vergleichen Sie für **statische** SQL- und XQuery-Anweisungen die Laufzeiten der Anweisungen.

  Obwohl es vielleicht auch interessant wäre, die Kompilierzeit statischer SQL- und XQuery-Anweisungen zu kennen, ist die Gesamtzeit aus Kompilier- und Laufzeit der Anweisung in einem sinnvollen Kontext nur wenig aussagekräftig. Beim Vergleich der Gesamtzeit wird die Tatsache außer Acht gelassen, dass eine statische Abfrageanweisung nach jedem Binden viele Male ausgeführt werden kann und dass sie in der Regel nicht während der Laufzeit gebunden wird.

2. Geben Sie die Optimierungsklasse wie folgt an:

   - **Dynamische** SQL- und XQuery-Anweisungen verwenden die Optimierungsklasse, die im Sonderregister CURRENT QUERY OPTIMIZATION angegeben ist, das mit der SQL-Anweisung SET definiert wird. Beispielsweise setzt die folgende Anweisung die Optimierungsklasse auf 1:

     ```
     SET CURRENT QUERY OPTIMIZATION = 1
     ```

     Um sicherzustellen, dass eine dynamische SQL- oder XQuery-Anweisung immer dieselbe Optimierungsklasse verwendet, kann eine Anweisung SET in das Anwendungsprogramm integriert werden.

     Wenn das Register CURRENT QUERY OPTIMIZATION nicht definiert ist, werden die dynamischen Anweisungen mit der Standardoptimierungsklasse für Abfragen gebunden. Der Standardwert für dynamische und statische Abfragen wird durch den Wert des Datenbankkonfigurationsparameters *dft_queryopt* festgelegt. Der Standardwert für diesen Parameter ist Klasse 5. Die Standardwerte für die Bindeoption und das Sonderregister werden ebenfalls aus dem Datenbankkonfigurationsparameter *dft_queryopt* gelesen.

   - **Statische** SQL- und XQuery-Anweisungen verwenden die in den Befehlen PREP und BIND angegebene Optimierungsklasse. In der Spalte QUERYOPT in der Katalogtabelle SYSCAT.PACKAGES wird die zum Binden des Pakets verwendete Optimierungsklasse aufgezeichnet. Wenn das Paket erneut implizit oder mit dem Befehl REBIND PACKAGE gebunden wird, wird dieselbe Optimierungsklasse für die statischen Abfrageanweisungen verwendet. Verwenden Sie zum Ändern der Optimierungsklasse für solche statischen SQL- und XQuery-Anweisungen den Befehl BIND. Wenn Sie keine Optimierungsklasse angeben, verwendet DB2 die vom Datenbankkonfigurationsparameter *dft_queryopt* angegebene Standardoptimierungsklasse.

Profile und Richtlinien für das Optimierungsprogramm - Übersicht

Das DB2-Optimierungsprogramm ist eines der höchstentwickelten aufwandsbasierten Optimierungsprogramme am Markt. In seltenen Fällen ist es jedoch möglich, dass das Optimierungsprogramm einen nicht optimalen Ausführungsplan auswählt. Als Datenbankadministrator (DBA), der mit der Datenbank vertraut ist, können Sie Möglichkeiten wie 'db2advis', RUNSTATS, 'db2expln' und die Einstellung der Optimierungsklasse dazu verwenden, das Optimierungsprogramm im Hinblick auf bessere Datenbankleistung zu optimieren. Wenn sich nach Ausschöpfung aller Optimierungsoptionen nicht die gewünschten Ergebnisse einstellen, können Sie dem DB2-Optimierungsprogramm explizite Optimierungsrichtlinien zur Verfügung stellen.

Ein Optimierungsprofil ist ein XML-Dokument, das Optimierungsrichtlinien für eine oder auch mehrere SQL-Anweisungen enthalten kann. Die Beziehung zwischen einer SQL-Anweisung und den ihr zugeordneten Optimierungsrichtlinien wird durch den SQL-Text und andere relevante Informationen hergestellt, die zur eindeutigen Identifizierung einer SQL-Anweisung erforderlich sind. Abb. 31 zeigt, wie sich eine Optimierungsrichtlinie mithilfe eines Optimierungsprofils an das DB2-Optimierungsprogramm übergeben lässt.

```
<?xml version="1.0" encoding="UTF-8">

<OPTPROFILE VERSION="9.1.0.0">
<STMTPROFILE ID="Guidelines for TPCD Q9">
  <STMTKEY SCHEMA="TPCD">
    SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
    FROM PARTS P, SUPPLIERS S, PARTSUPP PS
    WHERE P_PARTKEY = PS.PS_PARTKEY
      AND S.S_SUPPKEY = PS.PS_SUPPKEY
      AND P.P_SIZE = 39
      AND P.P_TYPE = 'BRASS'
      AND S.S_NATION = 'MOROCCO'
      AND S.S_NATION IN ('MOROCCO', 'SPAIN')
      AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                              FROM PARTSUPP PS1, SUPPLIERS S1
                              WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                                AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                                AND S1.S_NATION = S.S_NATION))
  </STMTKEY>
  <OPTGUIDELINES><IXSCAN TABLE="S" INDEX="I_SUPPKEY"/></OPTGUIDELINES>
</STMTPROFILE>
</OPTPROFILE>
```

Abbildung 31. Übergeben von Richtlinien durch ein Optimierungsprofil

Jedes Element STMTPROFILE (Anweisungsprofil) stellt einen Satz von Optimierungsrichtlinien für eine Anwendungsanweisung bereit. Die betroffene Anweisung wird durch ein Unterelement STMTKEY (Anweisungsschlüssel) angegeben. Das Optimierungsprofil erhält einen durch ein Schema qualifizierten Namen und wird in die Datenbank eingefügt. Durch Angeben dieses Namens im Befehl BIND oder PRECOMPILE wird das Optimierungsprofil für die Optimierung der Anweisung in Kraft gesetzt.

Durch Optimierungsprofile können dem Optimierungsprogramm Optimierungsrichtlinien zur Verfügung gestellt werden, ohne Änderungen an Anwendungen oder an der Datenbankkonfiguration vornehmen zu müssen. Sie wählen lediglich das einfache XML-Dokument aus, fügen es in die Datenbank ein und geben den Namen des Optimierungsprofils im Befehl BIND oder PRECOMPILE an. Das Optimierungsprogramm ordnet die Optimierungsrichtlinien automatisch der richtigen Anweisung zu.

Optimierungsrichtlinien müssen nicht umfassend sein, jedoch sollten sie speziell auf einen gewünschten Ausführungsplan ausgerichtet werden. Das DB2-Optimierungsprogramm arbeitet auch weiterhin mit der Auswahl möglicher anderer Pläne unter Verwendung der vorhandenen Methoden zur Aufwandsberechnung. Optimierungsrichtlinien für bestimmte Tabellenverweise können Einstellungen allgemeiner Optimierungsparameter nicht überschreiben. Daher wäre zum Beispiel eine Optimierungsrichtlinie, die einen Mischjoin zwischen Tabelle A und Tabelle B angibt, bei Optimierungsklasse 0 nicht gültig.

Das Optimierungsprogramm ignoriert ungültige oder nicht anwendbare Optimierungsrichtlinien. Wenn Optimierungsrichtlinien ignoriert werden, wird ein Ausführungsplan generiert und eine Warnung SQL0437W mit dem Ursachencode 13 zurückgegeben. Anschließend können Sie mithilfe der Anweisung EXPLAIN detaillierte Diagnoseinformationen zur Verarbeitung von Optimierungsrichtlinien abrufen.

Optimierungsrichtlinien

Typen von Optimierungsrichtlinien und Verarbeitung - Übersicht: Das DB2-Optimierungsprogramm optimiert eine Anweisung in zwei Phasen. Die optimierte Anweisung wird durch die *Optimierungsphase der Abfrageumschreibung* bestimmt, in der die ursprüngliche Anweisung in eine semantisch äquivalente Anweisung umgeformt wird, die sich in der Planoptimierungsphase leichter optimieren lässt. In der *Planoptimierungsphase* werden die optimalen Zugriffsmethoden, Joinmethoden und Joinreihenfolgen für die *optimierte Anweisung* bestimmt, indem eine Anzahl von Alternativen aufgezählt und die Alternative ausgewählt wird, die den günstigsten Schätzwert für den Ausführungsaufwand liefert.

Die Abfrageumsetzungen, Zugriffsmethoden, Joinmethoden, Joinreihenfolgen und andere Optimierungsalternativen, die während der beiden Optimierungsphasen in Betracht gezogen werden, werden durch verschiedene DB2-Parameter gesteuert, wie zum Beispiel CURRENT QUERY OPTIMIZATION (ein Sonderregister), REOPT (eine Bindeoption) und *DB2_REDUCED_OPTIMIZATION* (eine Registrierdatenbankvariable). Die Gruppe der Optimierungsalternativen, die bei der Optimierung einer Anweisung in Betracht gezogen werden, wird als Suchbereich bezeichnet.

Die folgenden Typen von Optimierungsrichtlinien für Anweisungen werden unterstützt:

- Allgemeine Optimierungsrichtlinien
- Richtlinien für das Umschreiben von Abfragen
- Richtlinien zur Planoptimierung

Optimierungsrichtlinien werden in einer bestimmten Reihenfolge angewendet. Allgemeine Optimierungsrichtlinien werden zuerst angewendet, da sie sich auf den Suchbereich auswirken können. Als nächste werden Richtlinien für das Umschreiben von Abfragen angewendet, da sie sich auf die in der Planoptimierungsphase optimierte Anweisung auswirken können. Planoptimierungsrichtlinien werden zuletzt angewendet.

Allgemeine Optimierungsrichtlinien können verwendet werden, um die Einstellung allgemeiner Optimierungsparameter zu beeinflussen. *Richtlinien für das Umschreiben von Abfragen* können verwendet werden, um die Umsetzungen zu beeinflussen, die in der Optimierungsphase der Abfrageumschreibung in Betracht gezogen werden. *Planoptimierungsrichtlinien* können verwendet werden, um die Zugriffsmethoden, Joinmethoden und Joinreihenfolgen zu beeinflussen, die in der Planoptimierungsphase in Betracht gezogen werden.

Allgemeine Optimierungsrichtlinien: *Allgemeine Optimierungsrichtlinien* können zur Festlegung allgemeiner Optimierungsparameter verwendet werden. Geltungsbereich jeder dieser Richtlinien ist die Anweisungsebene.

Optimierungsrichtlinien für das Umschreiben von Abfragen: *Richtlinien für das Umschreiben von Abfragen* können verwendet werden, um die Umsetzungen zu beeinflussen, die in der Optimierungsphase der Abfrageumschreibung in Betracht gezogen werden. In der Optimierungsphase der Abfrageumschreibung wird die ursprüngliche Anweisung in eine semantisch äquivalente, *optimierte Anweisung* umgeformt. Der optimale Ausführungsplan für die optimierte Anweisung wird anschließend in der Phase der Planoptimierung bestimmt. Infolgedessen können sich Optimierungsrichtlinien für die Abfrageumschreibung auf die Anwendbarkeit der Planoptimierungsrichtlinien auswirken.

Jede Optimierungsrichtlinie für die Abfrageumschreibung entspricht einer der Abfrageumsetzungsregeln des Optimierungsprogramms. Die folgenden Abfrageumsetzungsregeln können durch Optimierungsrichtlinien für die Abfrageumschreibung beeinflusst werden:

- Regel zur Umsetzung von IN-Listen in Joins
- Regel zur Umsetzung von Unterabfragen in Joins
- Regel zur Umsetzung von NOT EXISTS-Unterabfragen in Antijoins
- Regel zur Umsetzung von NOT IN-Unterfragen in Antijoins

Optimierungsrichtlinien für das Umschreiben von Abfragen sind nicht immer anwendbar. Abfrageumschreiberegeln werden jeweils einzeln umgesetzt. Infolgedessen können sich einige Abfrageumschreiberegeln, die vor einer nachfolgenden Regel umgesetzt werden, auf die Optimierungsrichtlinie für das Umschreiben von Abfragen auswirken, die dieser Regel zugeordnet ist. Manchmal kann sich die Umgebungskonfiguration auf das Verhalten einiger Umschreiberegeln auswirken, was sich wiederum auf die Anwendbarkeit der Optimierungsrichtlinie für das Umschreiben von Abfragen für eine bestimmte Regel auswirken kann. Um sicherzustellen, dass jedes Mal dasselbe Ergebnis erzielt wird, unterliegen Abfrageumschreiberegeln einigen Bedingungen, bevor sie umgesetzt werden. Wenn die mit der Regel verbundenen Bedingungen zu dem Zeitpunkt, zu dem die Abfrageumschreibekomponente versucht, die Regel auf die Abfrage anzuwenden, nicht erfüllt sind, wird die Optimierungsrichtlinie für die Abfrageumschreibung für die Regel ignoriert. Wenn die Optimierungsrichtlinie für die Abfrageumschreibung nicht anwendbar ist und die Richtlinie eine aktivierende Richtlinie ist, wird eine Fehlernachricht SQL0437W mit Ursachencode 13 zurückgegeben. Wenn die Optimierungsrichtlinie für die Abfrageumschreibung nicht anwendbar ist und die Richtlinie eine inaktivierende Richtlinie ist, wird keine Fehlernachricht zurückgegeben. Die Abfrageumschreiberegel wird in diesem Fall nicht angewendet, weil die Regel so behandelt wird, als wäre sie inaktiviert worden.

Die Optimierungsrichtlinien für die Abfrageumschreibung können in zwei Kategorien untergliedert werden: Richtlinien auf Anweisungsebene und Richtlinien auf Vergleichselementebene. Alle Optimierungsrichtlinien für die Abfrageumschreibung unterstützen die Kategorie der Anweisungsebene. Nur INLIST2JOIN unterstützt die Kategorie der Vergleichselementebene. Die Optimierungsrichtlinie für die Abfrageumschreibung auf Anweisungsebene gilt für die gesamte Abfrage. Die Optimierungsrichtlinie für die Abfrageumschreibung auf Vergleichselementebene gilt nur für das bestimmte Vergleichselement. Wenn Optimierungsrichtlinien für die Abfrageumschreibung sowohl auf Anweisungs- als auch auf Vergleichselementebene angegeben werden, überschreibt die Richtlinie auf Vergleichselementebene die Richtlinie auf Anweisungsebene für das bestimmte Vergleichselement.

Jede Optimierungsrichtlinie für die Abfrageumschreibung wird durch ein entsprechendes *Umschreibeanforderungselement* im Schema der Optimierungsrichtlinie dargestellt.

Richtlinien zur Planoptimierung: Planoptimierungsrichtlinien werden in der auf Aufwandsberechnungen basierenden Phase der Optimierung angewendet, in der Zugriffsmethoden, Joinmethoden, Joinreihenfolgen und andere Details des Ausführungsplans für die Anweisung bestimmt werden. Planoptimierungsrichtlinien brauchen nicht alle Aspekte eines Ausführungsplans anzugeben. Nicht angegebene Aspekte des Ausführungsplans werden durch das Optimierungsprogramm auf der Grundlage von Aufwandsberechnungen ermittelt.

Es gibt zwei Kategorien von Planoptimierungsrichtlinien:

- *accessRequest*: Eine Zugriffsanforderung gibt eine gewünschte Zugriffsmethode zur Erfüllung eines Tabellenverweises in einer Anweisung an.
- *joinRequest*: Eine Joinanforderung gibt eine gewünschte Methode und Reihenfolge einer Joinoperation an. Joinanforderungen bestehen wiederum aus weiteren Zugriffs- oder Joinanforderungen.

Optimierungsrichtlinien für Zugriffsanforderungen entsprechen den Datenzugriffsmethoden des Optimierungsprogramms, wie zum Beispiel Tabellensuche, Indexsuche und Vorablesezugriff über Listen. Die Richtlinien für Joinanforderungen entsprechen den Joinmethoden des Optimierungsprogramms, wie zum Beispiel Join mit Verschachtelungsschleife (Nested Loop Join), Hash-Join und Mischjoin (Merge Join). Diese Methoden werden in *Systemverwaltung: Optimierung*.

Bildung von Tabellenverweisen in Optimierungsrichtlinien: Der Begriff *Tabellenverweis* wird in diesem Dokument in der Bedeutung eines beliebigen Ausdrucks für eine Tabelle oder Sicht bzw. eines Aliasnamens in einer SQL-Anweisung bzw. Sichtdefinition verwendet. Eine Optimierungsrichtlinie kann einen Tabellenverweis entweder durch den in der ursprünglichen Anweisung angegebenen exponierten Namen (engl. exposed name) oder durch den eindeutigen Korrelationsnamen angeben, der dem Tabellenverweis in der *optimierten Anweisung* zugeordnet wird. Erweiterte Namen, die aus Folgen von exponierten Namen bestehen, können verwendet werden, um die eindeutige Angabe von Tabellenverweisen zu unterstützen, die in Sichten eingebettet sind. Optimierungsrichtlinien, die exponierte Namen oder erweiterte Namen angeben, die im Kontext der gesamten Anweisung nicht eindeutig sind, werden als mehrdeutig angesehen und nicht angewendet. Wenn darüber hinaus derselbe Tabellenverweis von mehr als einer Optimierungsrichtlinie angegeben wird, werden alle Optimierungsrichtlinien, die diesen Tabellenverweis angeben, als gegenseitig unverträglich betrachtet und nicht angewendet. In den folgenden Abschnitten werden diese besonderen technologischen Aspekte der Optimierungsrichtlinien etwas eingehender beschrieben. Aufgrund möglicher Abfrageumsetzungen ist nicht garantiert, dass ein exponierter oder erweiterter Name während der Optimierung erhalten bleibt. In einem solchen Fall wird jede Richtlinie, die auf den entsprechenden Tabellenverweis Bezug nimmt, ignoriert.

Verwenden von exponierten Namen der ursprünglichen Anweisung zur Angabe von Tabellenverweisen

Ein Tabellenverweis wird unter Verwendung seines exponierten Namens angegeben. Der exponierte Name wird auf dieselbe Weise angegeben, in der auch eine Tabelle in einer SQL-Anweisung qualifiziert wird. Die Regeln zur Angabe von SQL-Kennungen gelten ebenso für den Wert des Attributs TABLE.

Der Wert des Attributs TABLE einer Optimierungsrichtlinie wird mit jedem exponierten Namen der Anweisung verglichen. In diesem Release von DB2 ist nur eine einzige Übereinstimmung zulässig. Wenn der Wert des Attributs TABLE mit dem Schemanamen qualifiziert ist, entspricht er jedem äquivalenten, qualifizierten exponierten Tabellennamen. Wenn der Wert des Attributs TABLE nicht qualifiziert ist, entspricht er jedem äquivalenten Korrelationsnamen oder exponierten Tabellennamen (das heißt, der Wert des Attributs TABLE wird implizit als mit dem für die Anweisung gültigen Standardschema qualifiziert betrachtet). Diese Konzepte werden durch die Beispielanweisung in Abb. 32 veranschaulicht. Nehmen Sie an, die Anweisung wird unter Verwendung des Standardschemas „Tpcd" optimiert.

```
SELECT S_NAME, S_ADDRESS, S_PHONE, S_COMMENT
FROM PARTS, SUPPLIERS, PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY AND
   P_SIZE = 39 AND P_TYPE = 'BRASS';
```

Abbildung 32. Verwenden des Werts des Attributs TABLE einer Optimierungsrichtlinie zum Vergleichen mit jedem exponierten Namen einer Anweisung

Werte des Attributs TABLE, die Tabellenverweise in der Anweisung ordnungsgemäß angeben, wären zum Beispiel: '"Tpcd".PARTS', 'PARTS', 'Parts' (da die Kennung keine Begrenzungszeichen hat, wird sie in Großschreibung umgesetzt). Werte des Attributs TABLE, die einen Tabellenverweis in der Anweisung nicht korrekt angeben, wären zum Beispiel: '"Tpcd2".SUPPLIERS', 'PARTSUPP' (kein exponierter Name) und 'Tpcd.PARTS' (die Kennung Tpcd muss mit Begrenzern angegeben werden, da sie ansonsten in Großbuchstaben umgesetzt wird).

Der exponierte Name kann zur Angabe eines beliebigen Tabellenverweises in der ursprünglichen Anweisung, Sicht, SQL-Funktion oder im ursprünglichen Trigger verwendet werden.

Verwenden von exponierten Namen der ursprünglichen Anweisung zur Angabe von Tabellenverweisen in Sichten

Optimierungsrichtlinien können eine erweiterte Syntax zur Angabe von Tabellenverweisen verwenden, die in Sichten eingebettet sind.

Eine erweiterte Syntax kann zur Angabe eines beliebigen Tabellenverweises in der ursprünglichen Anweisung, der ursprünglichen SQL-Funktion oder im ursprünglichen Trigger verwendet werden.

Angeben von Tabellenverweisen mit Korrelationsnamen in der optimierten Anweisung

Eine Optimierungsrichtlinie kann einen Tabellenverweis auch durch die eindeutigen Korrelationsnamen angeben, die dem Tabellenverweis in der *optimierten Anweisung* zugeordnet werden. Die optimierte Anweisung ist eine semantisch äquivalente Version der ursprünglichen Anweisung, wie dies in der Optimierungsphase der Abfrageumschreibung bestimmt wird. Die optimierte Anweisung kann aus den EXPLAIN-Tabellen abgerufen werden. Das Attribut TABID einer Optimierungsrichtlinie dient zur Angabe der Tabellenverweise in der optimierten Anweisung.

Wenn in einer Optimierungsrichtlinie sowohl das Attribut TABLE als auch das Attribut TABID angegeben sind, müssen sie sich auf denselben Tabellenverweis beziehen. Ansonsten wird die Optimierungsrichtlinie ignoriert.

Anmerkung: Zurzeit gibt es keine Garantie, dass Korrelationsnamen in der optimierten Anweisung bei einem Upgrade auf ein neues Release von DB2 stabil bleiben.

Mehrdeutige Tabellenverweise

Eine Optimierungsrichtlinie gilt als ungültig und wird nicht angewendet, wenn sie mehreren exponierten oder erweiterten Namen entspricht.

Zur Sicherstellung eindeutiger Namen kann die Sicht so umgeschrieben werden, dass sie eindeutige Korrelationsnamen verwendet, oder es kann das Attribut TABID verwendet werden.

Anmerkung: Tabellenverweise, die im Feld TABID angegeben werden, sind nie mehrdeutig, da alle Korrelationsnamen in der optimierten Anweisung eindeutig sind.

Unverträgliche Optimierungsrichtlinien

Der gleiche Tabellenverweis kann nicht von mehreren Optimierungsrichtlinien angegeben werden.

Wenn sich zwei oder mehr Richtlinien auf dieselbe Tabelle beziehen, wird nur die erste angewendet. Alle anderen Richtlinien werden unter Rückgabe einer Fehlernachricht ignoriert.

Es besteht eine Einschränkung bei mehreren Umschreibeanforderungselementen INLIST2JOIN, die die OPTION 'ENABLE' (Aktivieren) auf Vergleichselementebene angeben. In einer Abfrage kann nur ein solches Umschreibeanforderungselement INLIST2JOIN auf Vergleichselementebene angegeben werden.

Überprüfen, ob die Optimierungsrichtlinien verwendet wurden:

Das Optimierungsprogramm unternimmt jeden Versuch, die in einem Optimierungsprofil oder durch eingebettete SQL-Optimierungsrichtlinien (auch als Anweisungsprofil bezeichnet) angegebenen Optimierungsrichtlinien einzuhalten. Das Optimierungsprogramm kann jedoch ungültige oder nicht anwendbare Richtlinien zurückweisen.

Zur Verwendung der EXPLAIN-Einrichtung müssen die EXPLAIN-Tabellen vorhanden sein. Die DDL-Anweisungen zur Erstellung der EXPLAIN-Tabellen sind in der Datei EXPLAIN.DDL enthalten, die sich im Unterverzeichnis misc des Verzeichnisses sqllib befindet.

Führen Sie die folgenden Schritte aus, um zu prüfen, ob eine gültige Optimierungsrichtlinie verwendet wurde:

1. Führen Sie den Befehl EXPLAIN für die Anweisung aus. Wenn eine Optimierungsrichtlinie für die Anweisung entweder durch ein Optimierungsprofil oder einen SQL-Kommentar wirksam war, wird der Name des Optimierungsprofils als Argument des Operators RETURN in der Tabelle EXPLAIN_ARGUMENTS ausgewiesen. Und wenn die Optimierungsrichtlinie eine eingebettete SQL-Optimierungsrichtlinie bzw. ein Anweisungsprofil enthielt, die der aktuellen Anweisung entsprach, wird der Name des Anweisungsprofils als Argument eines Operators RETURN ausgegeben. Die Typen der beiden neuen Argumentwerte sind OPT_PROF und STMTPROF.
2. Untersuchen Sie die Ergebnisse der mit EXPLAIN bearbeiteten Anweisung. Die folgende Abfrage auf die EXPLAIN-Tabellen kann modifiziert werden, um den Namen des Optimierungsprofils und den Namen des Anweisungsprofils für Ihre bestimmte Kombination aus EXPLAIN_REQUESTER, EXPLAIN_TIME, SOURCE_NAME, SOURCE_VERSION und QUERYNO zurückzugeben:

```
SELECT VARCHAR(B.ARGUMENT_TYPE,   9) as TYPE,
            VARCHAR(B.ARGUMENT_VALUE, 24) as VALUE

   FROM   EXPLAIN_STATEMENT A, EXPLAIN_ARGUMENT B

   WHERE  A.EXPLAIN_REQUESTER = 'SIMMEN'
       AND  A.EXPLAIN_TIME      = '2003-09-08-16.01.04.108161'
       AND  A.SOURCE_NAME       = 'SQLC2E03'
       AND  A.SOURCE_VERSION    = ''
       AND  A.QUERYNO           = 1

       AND  A.EXPLAIN_REQUESTER = B.EXPLAIN_REQUESTER
       AND  A.EXPLAIN_TIME      = B.EXPLAIN_TIME
       AND  A.SOURCE_NAME       = B.SOURCE_NAME
       AND  A.SOURCE_SCHEMA     = B.SOURCE_SCHEMA
       AND  A.SOURCE_VERSION    = B.SOURCE_VERSION
       AND  A.EXPLAIN_LEVEL     = B.EXPLAIN_LEVEL
       AND  A.STMTNO            = B.STMTNO
       AND  A.SECTNO            = B.SECTNO

       AND  A.EXPLAIN_LEVEL     = 'P'

       AND  (B.ARGUMENT_TYPE  = 'OPT_PROF' OR ARGUMENT_TYPE = 'STMTPROF')
       AND  B.OPERATOR_ID = 1
```

Abbildung 33. Verwenden einer Abfrage zum Abrufen von Ergebnissen aus einer mit EXPLAIN bearbeiteten Anweisung

Wenn die Optimierungsrichtlinie aktiv ist und die mit EXPLAIN-bearbeitete Anweisung der Anweisung entspricht, die im Element STMTKEY (Anweisungsschlüssel) der Optimierungsrichtlinie enthalten ist, liefert eine Abfrage ähnlich der in Abb. 33 gezeigten eine Ausgabe wie in Abb. 34. Der Wert für das STMTPROF-Argument ist derselbe wie für das Attribut ID im Element STMTPROFILE.

```
TYPE      VALUE
--------- -------------------------
OPT_PROF  NEWTON.PROFILE1
STMTPROF  Guidelines for TPCD Q9
```

Abbildung 34. Ausgabe einer Abfrage zum Abrufen von Ergebnissen aus einer mit EXPLAIN bearbeiteten Anweisung

Optimierungsprofile

Aufbau eines Optimierungsprofils: Dieser Abschnitt enthält eine Einführung in den Inhalt eines Optimierungsprofils. Der gültige Inhalt eines Optimierungsprofils für ein bestimmtes DB2-Release wird durch ein XML-Schema beschrieben, das als aktuelles Optimierungsprofilschema (COPS - Current Optimization Profile Schema) bezeichnet wird. Beachten Sie, dass ein Optimierungsprofil nur für DB2 Database für Linux-, UNIX- und Windows-Server gilt.

Eine Darstellung des aktuellen Optimierungsprofilschemas für das aktuelle Release von DB2 finden Sie in „Aktuelles Schema für Optimierungsprofile" auf Seite 441.

Ein Optimierungsprofil kann globale Richtlinien enthalten, die für alle DML-Anweisungen gelten, während das Profil wirksam ist. Und es kann spezielle Richtlinien enthalten, die sich jeweils auf eine einzelne DML-Anweisung in einem Paket beziehen. Zum Beispiel:

- Sie könnten eine globale Optimierungsrichtlinie schreiben, die anfordert, dass das Optimierungsprogramm auf die MQTs (Materialized Query Tables) 'Test-.SumSales' und 'Test.AvgSales' für jede Anweisung zurückgreift, die angetroffen wird, während das aktuelle Optimierungsprofil aktiv ist.
- Sie könnten eine Anweisungsoptimierungsrichtlinie schreiben, die anfordert, dass der Index I_SUPPKEY für den Zugriff auf die Tabelle SUPPLIERS verwendet wird, immer wenn das Optimierungsprogramm auf die Zielanweisung trifft.

Ein Optimierungsprofil enthält daher zwei Elemente, die die Hauptabschnitte definieren, in denen Sie die beiden Typen von Richtlinien angeben können: ein globales Element *OPTGUIDELINES* und eine beliebige Anzahl von Elementen *STMTPROFILE*. Ein Optimierungsprofil muss außerdem ein Element *OPTPROFILE* enthalten, das den Abschnitt definiert, der Metadaten und Verarbeitungsanweisungen enthält.

Abb. 35 auf Seite 434 zeigt ein Beispiel für ein gültiges Optimierungsprofil für DB2 Version 9.1. Dieses Optimierungsprofil besitzt einen Abschnitt für globale Optimierungsrichtlinien und einen Abschnitt für Anweisungsprofile.

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE VERSION="9.1.0.0">

  <!--
       Abschnitt für globale Optimierungsrichtlinien.
       Optional, jedoch maximal ein Abschnitt dieser Art zulässig.
  -->
  <OPTGUIDELINES>
    <MQT NAME="Test.AvgSales"/>
    <MQT NAME="Test.SumSales"/>
  </OPTGUIDELINES>

  <!--
       Abschnitt für Anweisungsprofile.
       Null oder mehr Abschnitte möglich.
  -->
  <STMTPROFILE ID="Guidelines for TPCD Q9">
    <STMTKEY SCHEMA="TPCD">
      <![CDATA[SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE,
S.S_COMMENT FROM PARTS P, SUPPLIERS S, PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY AND P.P_SIZE = 39
AND P.P_TYPE = 'BRASS' AND S.S_NATION = 'MOROCCO' AND S.S_NATION IN ('MOROCCO', 'SPAIN')
AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST) FROM PARTSUPP PS1, SUPPLIERS S1
WHERE P.P_PARTKEY = PS1.PS_PARTKEY AND S1.S_SUPPKEY = PS1.PS_SUPPKEY AND
S1.S_NATION = S.S_NATION)]]>
    </STMTKEY>
        <OPTGUIDELINES>
      <IXSCAN TABID="Q1" INDEX="I_SUPPKEY"/>
    </OPTGUIDELINES>
  </STMTPROFILE>

</OPTPROFILE>
```

Abbildung 35. Ein gültiges Optimierungsprofil

Das Element OPTPROFILE

Das Optimierungsprofil beginnt mit dem Element OPTPROFILE. In dem in Abb. 35 gezeigten Beispiel besteht dieses Element aus einem Attribut VERSION, das angibt, dass die Optimierungsprofilversion 9.1 ist.

Abschnitt für globale Optimierungsrichtlinien

Globale Optimierungsrichtlinien geben Optimierungsrichtlinien an, die für alle Anweisungen gelten, für die das Optimierungsprofil wirksam ist. Der Abschnitt für globale Optimierungsrichtlinien wird durch das globale Element *OPTGUIDELINES* dargestellt. In dem in Abb. 35 gezeigten Beispiel enthält dieser Abschnitt nur eine globale Optimierungsrichtlinie, die angibt, dass die MQTs (Materialized Query Tables) 'Test.AvgSales' und 'Test.SumSales' zur Erfüllung aller Anweisungen in Betracht gezogen werden sollen, für die das Optimierungsprofil wirksam ist.

Abschnitt für Anweisungsprofile

Ein Anweisungsprofil definiert die Optimierungsrichtlinien, die für eine bestimmte Anweisung gelten. In einem Optimierungsprofil können null oder mehr Anweisungsprofile enthalten sein. Abschnitte für Anweisungsprofile werden durch das Element *STMTPROFILE* dargestellt. In dem in Abb. 35 gezeigten Beispiel enthält dieser Abschnitt Richtlinien für eine bestimmte Anweisung, für die das Optimierungsprofil wirksam ist.

Jedes Anweisungsprofil enthält einen Anweisungsschlüssel und Optimierungsrichtlinien auf Anweisungsebene, die durch die Elemente *STMTKEY* und *OPTGUIDELINES* dargestellt werden.

Der Anweisungsschlüssel gibt die Anweisung an, für die die Optimierungsrichtlinien auf Anweisungsebene gelten. Ebenfalls in dem in Abb. 35 auf Seite 434 gezeigten Beispiel enthält das Element *STMTKEY* den ursprünglichen Anweisungstext und weitere Informationen, die zur eindeutigen Angabe der gewünschten Anweisung erforderlich sind.

Das Optimierungsprogramm gleicht ein Anweisungsprofil automatisch anhand des Anweisungsschlüssels mit der entsprechenden Anweisung ab. Diese Beziehung bietet Ihnen die Möglichkeit, Optimierungsrichtlinien für eine Anweisung in einer Anwendung anzugeben, ohne die Anwendung modifizieren zu müssen.

Der Abschnitt für Optimierungsrichtlinien auf Anweisungsebene des Anweisungsprofils wird durch das Element *OPTGUIDELINES* dargestellt. Bei einem erfolgreichen Abgleich des Anweisungsschlüssels in einem Anweisungsprofil greift das Optimierungsprogramm auf die zugeordneten Anweisungsoptimierungsrichtlinien zurück, wenn es die Anweisung optimiert.

Abschnitt für Anweisungsoptimierungsrichtlinien

In dem in Abb. 35 auf Seite 434 gezeigten Beispiel enthält der Abschnitt für Anweisungsprofile einen Abschnitt für Anweisungsoptimierungsrichtlinien, der durch das Element *OPTGUIDELINES* dargestellt wird.

Dieser Abschnitt gibt Aspekte des gewünschten Abfrageausführungsplans für die angegebene Anweisung an. Er besteht aus einem oder mehreren *Zugriffs-* und *Joinanforderungen*, durch die die gewünschten Methoden für den Zugriff auf und die Joinverknüpfung von Tabellen in der Anweisung spezifiziert werden. Die Richtlinien in Abb. 35 auf Seite 434 enthalten eine Zugriffsanforderung, die angibt, dass für die in der verschachtelten Subselect-Anweisung angegebene Tabelle SUPPLIERS ein Index mit dem Namen I_SUPPKEY verwendet werden soll.

In Abb. 35 auf Seite 434 gibt das Element *indexScan* eine Indexzugriffsanforderung an. Das Element *tableReference* des Elements 'indexScan' gibt an, dass die Tabelle SUPPLIERS als Ziel für die Zugriffsanforderung verwendet werden soll. Das Element *index* gibt an, dass der Index I_SUPPKEY verwendet werden soll.

Richtlinien brauchen nur Teile des gewünschten Abfrageausführungsplans anzugeben. Das Optimierungsprogramm kann durch Anwendung seines aufwandsbasierten Modells die restlichen Aspekte des Plans entscheiden.

Erstellen eines Optimierungsprofils:

Ein Optimierungsprofil muss gemäß dem aktuellen Optimierungsprofilschema gültig sein.

Da ein Optimierungsprofil zahlreiche Kombinationen von Richtlinien enthalten kann, gibt diese Task nur die Schritte an, die bei jeder Erstellung eines Optimierungsprofils auszuführen sind.

Führen Sie folgende Schritte aus, um ein Optimierungsprofil zu erstellen:

1. Starten Sie einen XML-Editor. Verwenden Sie, falls möglich, einen Editor mit Schemaprüffunktionalität. Das Optimierungsprogramm führt keinerlei XML-Gültigkeitsprüfung durch.
2. Erstellen Sie ein neues XML-Dokument mit einem für Sie sinnvollen Namen. Sinnvoll wäre beispielsweise ein Name, der den Geltungsbereich der Anweisungen beschreibt, für die die Richtlinien gelten sollen, wie zum Beispiel: inventory_db.xml.
3. Fügen Sie dem Dokument die XML-Deklaration hinzu. Wenn Sie kein Codierformat ('encoding') angeben, wird UTF-8 angenommen. Speichern Sie das Dokument mit UTF-16-Codierung, sofern möglich. DB2 kann diese Codierung effizienter verarbeiten.

   ```
   <?xml version="1.0" encoding="UTF-16"?>
   ```

4. Fügen Sie dem Dokument den Abschnitt für Optimierungsprofile hinzu.

   ```
   <OPTPROFILE VERSION="9.1.0.0">
     </OPTPROFILE>
   ```

5. Erstellen Sie innerhalb des Elements für Optimierungsprofile nach Bedarf globale Richtlinien oder Richtlinien auf Anweisungsebene, und speichern Sie die Datei.

Erstellen von Anweisungsoptimierungsrichtlinien:

Die folgenden Schritte beschreiben, wie Anweisungsoptimierungsrichtlinien erstellt werden.

Erstellen Sie ein Optimierungsprofil, in das Sie die Anwendungsrichtlinien einfügen möchten (siehe „Erstellen eines Optimierungsprofils" auf Seite 435).

Führen Sie folgende Schritte aus, um eine Anwendungsoptimierungsrichtlinie zu erstellen:

1. Schöpfen Sie alle anderen Optimierungsoptionen aus. Entsprechende Informationen finden Sie in *Systemverwaltung: Optimierung*. Zum Beispiel:
   a. Stellen Sie sicher, dass die Statistikdaten zur Datenverteilung kürzlich durch das Dienstprogramm RUNSTATS aktualisiert wurden.
   b. Stellen Sie sicher, dass DB2 mit der für die Auslastung geeigneten Optimierungsklasse arbeitet.
   c. Stellen Sie sicher, dass dem Optimierungsprogramm die entsprechenden Indizes für den Zugriff auf die in der Abfrage referenzierten Tabellen zur Verfügung stehen.
2. Führen Sie die EXPLAIN-Einrichtung für die fragliche Anweisung aus, und analysieren Sie die Ausgabe, um festzustellen, ob Richtlinien Abhilfe schaffen können. Wenn Sie feststellen, dass eine Anweisungsoptimierungsrichtlinie von Nutzen sein könnte, fahren Sie fort.
3. Rufen Sie die *ursprüngliche* Anweisung ab, indem Sie eine Abfrage ähnlich der folgenden ausführen:

   ```
   SELECT STATEMENT TEXT
     FROM EXPLAIN_STATEMENT
     WHERE  EXPLAIN_LEVEL = '0' AND
       EXPLAIN_REQUESTER = 'SIMMEN' AND
       EXPLAIN_TIME      = '2003-09-08-16.01.04.108161' AND
       SOURCE_NAME       = 'SQLC2E03' AND
       SOURCE_VERSION    = '' AND
       QUERYNO           = 1
   ```

4. Bearbeiten Sie das Optimierungsprofil, und erstellen Sie ein Anweisungsprofil, indem Sie den Anweisungstext in den Anweisungsschlüssel einfügen. Beispiel:

```
<STMTPROFILE ID="Guidelines for TPCD Q9">
   <STMTKEY SCHEMA="TPCD"><![CDATA[SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE,
     S.S_COMMENT
     FROM PARTS P, SUPPLIERS S, PARTSUPP PS
     WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY
     AND P.P_SIZE  = 39 AND P.P_TYPE  = 'BRASS' AND S.S_NATION
     = 'MOROCCO' AND
     PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
     FROM PARTSUPP PS1, SUPPLIERS S1
     WHERE P.P_PARTKEY  = PS1.PS_PARTKEY AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
     AND S1.S_NATION  = S.S_NATION)]]>
   </STMTKEY>
</STMTPROFILE>
```

5. Stellen Sie unter dem Anweisungsschlüssel die Anweisungsoptimierungsrichtlinien zusammen. Verwenden Sie exponierte Namen (engl. exposed names), um die Objekte anzugeben, die in Zugriffs- und Joinanforderungen verwendet werden. Das folgende Beispiel zeigt eine Joinanforderung:

```
<OPTGUIDELINES>
  <HSJOIN>
    <TBSCAN TABLE='PS1'/>
    <IXSCAN TABLE='S1'
            INDEX='I1'/>
  </HSJOIN>
</OPTGUIDELINES>
```

6. Führen Sie eine Gültigkeitsprüfung für die Datei aus, und speichern Sie die Datei.

Zum Testen der Ergebnisse führen Sie die Prozeduren aus, die in „Konfigurieren von DB2 zur Verwendung eines Optimierungsprofils", „Angeben des vom Optimierungsprogramm zu verwendenden Optimierungsprofils" auf Seite 438 und „Überprüfen, ob die Optimierungsrichtlinien verwendet wurden" auf Seite 431 beschrieben werden. Wenn Sie nicht die gewünschten Ergebnisse erzielen, nehmen Sie Änderungen an den Richtlinien vor (oder geben weitere Aspekte des Ausführungsplans an) und aktualisieren das Optimierungsprofil wie in „Modifizieren eines Optimierungsprofils" auf Seite 439 beschrieben. Wiederholen Sie diese Schritte je nach Bedarf.

Konfigurieren von DB2 zur Verwendung eines Optimierungsprofils:

Nach der Erstellung einer Optimierungsprofildatei und der Gültigkeitsprüfung ihres Inhalts am aktuellen Optimierungsprofilschema (COPS) muss dem Inhalt ein eindeutiger, mit einem Schemanamen qualifizierter Name zugeordnet und der Inhalt selbst in der Tabelle SYSTOOLS.OPT_PROFILE gespeichert werden.

Führen Sie folgende Schritte aus, um DB2 zur Verwendung eines Optimierungsprofils zu konfigurieren:

1. Erstellen Sie die Optimierungsprofiltabelle wie in „Tabelle SYSTOOLS.OPT_PROFILE" auf Seite 466 gezeigt. Jede Zeile der Tabelle kann genau ein Optimierungsprofil enthalten: die Spalten SCHEMA und NAME geben den eindeutigen Namen des Optimierungsprofils an, und die Spalte PROFILE enthält den Text des Optimierungsprofils.
2. Optional: Sie können jede Berechtigung erteilen, die Ihre Anforderungen an die Datenbanksicherheit erfüllt. Das Optimierungsprogramm kann die Tabelle unabhängig von der festgelegten Berechtigung lesen.
3. Fügen Sie die gewünschten Optimierungsprofile in die Tabelle ein.

Angeben des vom Optimierungsprogramm zu verwendenden Optimierungsprofils: Zur Angabe eines Optimierungsprofils auf Paketebene können Sie die Bindeoption OPTPROFILE verwenden. Zur Angabe eines Optimierungsprofils auf Anweisungsebene können Sie das Sonderregister CURRENT OPTIMIZATION PROFILE verwenden. Dieses Sonderregister enthält den qualifizierten Namen des Optimierungsprofils, das für Anweisungen verwendet wird, die dynamisch zur Optimierung vorbereitet werden. Für CLI-Anwendungen können Sie dieses Sonderregister über die Konfigurationsoption CURRENTOPTIMIZATIONPROFILE für jede Verbindung definieren.

Die Einstellung der Bindeoption OPTPROFILE gibt außerdem das *Standardoptimierungsprofil* für das Sonderregister CURRENT OPTIMIZATION PROFILE an. Für die Auswertung der Standardwerte gilt die folgende Reihenfolge:

- Die Bindeoption OPTPROFILE gilt für alle statischen Anweisungen ungeachtet aller anderen Einstellungen.
- Für dynamische Anweisungen wird der Wert des Sonderregisters CURRENTOPTIMIZATIONPROFILE durch folgende Werte in der Reihenfolge von der niedrigsten zur höchsten Priorität bestimmt:
  - Bindeoption OPTPROFILE
  - Clientkonfigurationsoption CURRENTOPTIMIZATIONPROFILE
  - Letzte Anweisung SET CURRENT OPTIMIZATION PROFILE in der Anwendung

Binden eines Optimierungsprofils an ein Paket:

Wenn Sie ein Paket mithilfe des Befehls BIND oder PRECOMPILE vorbereiten, können Sie die Option OPTPROFILE verwenden, um das für das Paket zu verwendende Optimierungsprofil anzugeben.

Diese Methode ist die einzige Möglichkeit, ein Optimierungsprofil auf statische Anweisungen anzuwenden. Dabei gilt das angegebene Profil für alle statischen Anweisungen im Paket. Ein Optimierungsprofil, das auf diese Weise angegeben wird, ist gleichzeitig das Standardoptimierungsprofil, das für dynamische Anweisungen innerhalb des Pakets verwendet wird.

Sie können ein Optimierungsprofil in SQLJ und in eingebettetem SQL mithilfe von APIs (z. B. 'sqlaprep') oder über den Befehlszeilenprozessor (CLP) binden.

Führen Sie zum Beispiel den folgenden Befehl über den Befehlszeilenprozessor aus, um das Optimierungsprofil für die Inventardatenbank (INVENTDB) an die Inventaranwendung (INVENTAPP) zu binden:

```
db2 prep inventapp.sqc bindfile optprofile NEWTON.INVENTDB
db2 bind inventapp.bnd
db2 connect reset
db2 terminate
xlc -I$HOME/sqllib/include -c inventapp.c -o inventapp.o
xlc -o inventapp inventapp.o -ldb2 -L$HOME/sqllib/lib
```

Wenn Sie keinen Schemanamen für das Optimierungsprofil angeben, wird der Wert der Option QUALIFIER als implizites Qualifikationsmerkmal verwendet.

Festlegen eines Optimierungsprofils in einer Anwendung:

In einer Anwendung können Sie die Einstellung des aktuellen Optimierungsprofils für dynamische Anwendungen mithilfe der Anweisung SET CURRENT OPTIMIZATION PROFILE steuern. Der Optimierungsprofilname, den Sie in der Anwei-

sung angeben, muss ein mit einem Schemanamen qualifizierter Name sein. Wenn Sie keinen Schemanamen angeben, wird der Wert des Sonderregisters CURRENT SCHEMA als implizites Schemaqualifikationsmerkmal verwendet.

Das Optimierungsprofil, das Sie angeben, gilt für alle nachfolgenden dynamischen Anweisungen, bis eine andere Anweisung SET CURRENT OPTIMIZATION PROFILE angetroffen wird. Statische Anweisungen sind nicht betroffen, da sie vorverarbeitet und in ein Paket eingefügt werden, bevor diese Einstellung ausgewertet wird.

Gehen Sie wie folgt vor, um ein Optimierungsprofil in einer Anwendung festzulegen:

- Sie können die Anweisung SET CURRENT OPTIMIZATION PROFILE überall in Ihrer Anwendung verwenden. Zum Beispiel wird die letzte Anweisung in der folgenden Sequenz gemäß dem Optimierungsprofil JON.SALES optimiert.

```
EXEC SQL  SET CURRENT OPTIMIZATION PROFILE = 'NEWTON.INVENTDB';

/* Die folgenden beiden Anweisungen werden mit 'NEWTON.INVENTDB' optimiert. */
EXEC SQL PREPARE stmt FROM SELECT ... ;
EXEC SQL EXECUTE stmt;

EXEC SQL EXECUTE IMMEDIATE SELECT ... ;

EXEC SQL SET CURRENT OPTIMIZATION PROFILE = 'JON.SALES';

/* Die folgende Anweisung wird mit 'JON.SALES' optimiert. */
EXEC SQL EXECUTE IMMEDIATE SELECT ... ;
```

Abbildung 36. Verwenden der Anweisung SET CURRENT OPTIMIZATION PROFILE zum Ändern eines Profils

- Wenn das Optimierungsprogramm das Standardoptimierungsprofil verwenden soll, das wirksam war, als die Anwendung gestartet wurde, geben Sie den Wert NULL an. Beispiel:

  ```
  EXEC SQL SET CURRENT OPTIMIZATION PROFILE = NULL;
  ```

- Wenn das Optimierungsprogramm keine Optimierungsprofile verwenden soll, geben Sie die leere Zeichenfolge (’’) an. Beispiel:

  ```
  EXEC SQL SET CURRENT OPTIMIZATION PROFILE = '';
  ```

- Wenn es sich um eine CLI-Anwendung handelt, können Sie der Datei db2cli.ini den Parameter CURRENTOPTIMIZATIONPROFILE hinzufügen. Verwenden Sie den Konfigurationsassistenten oder den Befehl UPDATE CLI CONFIGURATION, um der Datei den Eintrag hinzuzufügen. Beispiel:

  ```
  DB2 UPDATE CLI CFG FOR SECTION SANFRAN USING CURRENTOPTIMIZATIONPROFILE JON.SALES
  ```

  Durch diesen Befehl wird der folgende Eintrag in der Datei db2cli.ini erstellt:

  ```
  [SANFRAN]
  CURRENTOPTIMIZATIONPROFILE=JON.SALES
  ```

  Anmerkung: Diese Einstellung wird von jeder Anweisung SET CURRENT OPTIMIZATION PROFILE in der Anwendung überschrieben.

Modifizieren eines Optimierungsprofils:

Sie können ein Optimierungsprofil modifizieren, indem Sie das Dokument bearbeiten, es am aktuellen Schema für Optimierungsprofile (COPS) prüfen und das ursprüngliche Dokument in der Tabelle SYSTOOLS.OPT_PROFILE durch die neue Version ersetzen. Wenn ein Optimierungsprofil angegeben wird, wird es jedoch kompiliert und im Cache gespeichert. Diese Verweise müssen ebenfalls entfernt werden. Verwenden Sie die Anweisung FLUSH OPTIMIZATION PROFILE CACHE, um sowohl das alte Optimierungsprofil aus dem Optimierungsprofilcache

zu entfernen als auch alle im Cache für dynamische Pläne befindlichen Anweisungen ungültig zu machen, die zur Verwendung des alten Profils vorbereitet wurden (*logische Ungültigmachung*).

Führen Sie folgende Schritte aus, um ein Optimierungsprofil zu modifizieren:

1. Editieren Sie die Optimierungsprofildatei, um die erforderlichen Änderungen auszuführen, und prüfen Sie das XML.
2. Aktualisieren Sie die Zeile in der Tabelle SYSTOOLS.OPT_PROFILE mit dem neuen Profil.
3. Wenn Sie keine „Trigger zum Löschen des Cache für das Optimierungsprofil" auf Seite 466 erstellen, führen Sie die Anweisung FLUSH OPTIMIZATION PROFILE CACHE aus.

   Anmerkung: Wenn Sie den Cache für das Optimierungsprofil löschen (FLUSH), werden auch alle dynamischen Anweisungen im Cache für dynamische Pläne ungültig gemacht, die mit dem alten Optimierungsprofil vorbereitet wurden.

Alle nachfolgenden Verweise auf das Optimierungsprofil veranlassen das Optimierungsprogramm, das neue Profil zu lesen und es erneut in den Cache für das Optimierungsprofil zu laden. Aufgrund der logischen Ungültigmachung von Anweisungen, die unter dem alten Optimierungsprofil vorbereitet wurden, werden alle Aufrufe an diese Anweisungen unter dem neuen Optimierungsprofil vorbereitet und erneut im Cache für dynamische Pläne zwischengespeichert.

Löschen eines Optimierungsprofils:

Sie können ein Optimierungsprofil entfernen, das nicht mehr benötigt wird, indem Sie es aus der Tabelle SYSTOOLS.OPT_PROFILE löschen. Wenn ein Optimierungsprofil angegeben wird, wird es kompiliert und im Cache gespeichert. Wenn also das ursprüngliche Profil bereits verwendet wurde, müssen Sie das gelöschte Optimierungsprofil auch aus dem Optimierungsprofilcache löschen (FLUSH).

Führen Sie folgende Schritte aus, um ein Optimierungsprofil zu löschen:

1. Löschen Sie das Optimierungsprofil aus der Tabelle SYSTOOLS.OPT_PROFILE. Beispiel:

   ```
   DELETE FROM SYSTOOLS.OPT_PROFILE
     WHERE SCHEMA = 'NEWTON' AND NAME = 'INVENTDB';
   ```

2. Wenn Sie keine „Trigger zum Löschen des Cache für das Optimierungsprofil" auf Seite 466 erstellt haben, löschen Sie alle Versionen des Optimierungsprofils, die eventuell im Optimierungsprofilcache enthalten sind, indem Sie den Befehl FLUSH OPTIMIZATION PROFILE CACHE über den Befehlszeilenprozessor (CLP) ausführen.

   Anmerkung: Diese Aktion sorgt außerdem dafür, dass alle Anweisungen, die mit dem alten Optimierungsprofil erstellt wurden, im Cache für dynamische Pläne ungültig gemacht werden.

Alle nachfolgenden Verweise auf das Optimierungsprofil veranlassen das Optimierungsprogramm, eine Warnung SQL0437W mit Ursachencode 13 auszugeben.

XML-Schema für Optimierungsprofile und -richtlinien

Aktuelles Schema für Optimierungsprofile:

Das Schema für Optimierungsprofile sieht wie folgt aus.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" version="1.0">
<!--********************************************************************************************-->
<!-- Lizensiertes Material - Eigentum von IBM                                                   -->
<!-- (C) Copyright International Business Machines Corporation 2007. Alle Rechte vorbehalten.-->
<!-- U.S. Government Users Restricted Rights;  Use, duplication or disclosure restricted by  -->
<!-- GSA ADP Schedule Contract with IBM Corp.                                                  -->
<!--********************************************************************************************-->
<!--********************************************************************************************-->
<!-- Definition des aktuellen Optimierungsprofilschemas für Version 9.5.0.0                    -->
<!--                                                                                           -->
<!--   Ein Optimierungsprofil setzt sich aus folgenden Abschnitten zusammen:                   -->
<!--                                                                                           -->
<!--    + Ein Abschnitt für globale Optimierungsrichtlinien (maximal einer), in dem            -->
<!--      Optimierungsrichtlinien für alle Anweisungen definiert werden, für die das           -->
<!--      Optimierungsprofil wirksam ist.                                                      -->
<!--                                                                                           -->
<!--    +  Null oder mehr Anweisungsprofilabschnitte, die jeweils Optimierungsrichtlinien      -->
<!--       für eine bestimmte Anweisung definieren, für die das Optimierungsprofil             -->
<!--       wirksam ist.                                                                        -->
<!--                                                                                           -->
<!-- Das Attribut VERSION gibt die Version dieses Optimierungsprofilschemas an.                -->
<!--                                                                                           -->
<!--********************************************************************************************-->
<xs:element name="OPTPROFILE">
   <xs:complexType>
      <xs:sequence>
         <!-- Abschnitt für globale Optimierungsrichtlinien. Nur ein solcher Abschnitt ist zulässig. -->
         <xs:element name="OPTGUIDELINES" type="globalOptimizationGuidelinesType" minOccurs="0"/>
         <!-- Anweisungsprofilabschnitt. Null oder mehr Abschnitte zulässig -->
         <xs:element name="STMTPROFILE" type="statementProfileType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- Attribut VERSION ist gegenwärtig optional. -->
      <xs:attribute name="VERSION" use="optional">
         <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
               <xs:enumeration value="9.5.0.0"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
   </xs:complexType>
</xs:element>
<!--********************************************************************************************-->
<!--  In dieser Version unterstützte globale Optimierungsrichtlinien:                          -->
<!--  + MQTOptimizationChoices-Elemente beeinflussen die vom Optimierungsprogramm in           -->
<!--    Betracht gezogenen MQTs (Materialized Query Tables).                                   -->
<!--  + computationalPartitionGroupOptimizationsChoices-Elemente können die Optimierungen      -->
<!--    im Hinblick auf die Neupartitionierung mit Kurznamen beeinflussen.                     -->
<!--  + Das REOPT-Element kann die Optimierung von Anweisungen steuern, die Variablen          -->
<!--    enthalten.                                                                             -->
<!-- ******************************************************************************************** -->
<xs:complexType name="globalOptimizationGuidelinesType">
   <xs:sequence>
      <xs:group ref="MQTOptimizationChoices" />
      <xs:group ref="computationalPartitionGroupOptimizationChoices" />
      <xs:group ref="generalRequest"/>
   </xs:sequence>
</xs:complexType>
<!-- ********************************************************************************************-->
<!--  Elemente zur Beeinflussung der Optimierung mit MQTs (Materialized Query Tables):         -->
<!--                                                                                           -->
<!--  + MQTOPT - Kann zur Inaktivierung der Optimierung mit MQTs verwendet werden.             -->
<!--           Bei Inaktivierung zieht das Optimierungsprogramm keine MQTs zur Optimierung     -->
<!--           der Anweisung in Betracht.                                                      -->
<!--                                                                                           -->
<!--  + MQT - Mehrere dieser Elemente können angegeben werden. Jedes gibt eine MQT an,         -->
<!--     die bei der Optimierung der Anweisung berücksichtigt werden sollte. Nur               -->
<!--     angegebene MQTs werden berücksichtigt.                                                -->
<!--                                                                                           -->
<!--********************************************************************************************-->
<xs:group name="MQTOptimizationChoices">
```

```
   <xs:choice>
      <xs:element name="MQTOPT" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="OPTION" type="optionType" use="optional"/>
         </xs:complexType>
      </xs:element>
      <xs:element name="MQT" minOccurs="0" maxOccurs="unbounded">
         <xs:complexType>
            <xs:attribute name="NAME" type="xs:string" use="required"/>
         </xs:complexType>
      </xs:element>
   </xs:choice>
</xs:group>
<!-- ********************************************************************************************-->
<!--  Elemente zur Beeinflussung der Rechenpartitionsgruppenoptimierung (CPG-Optimierung):  -->
<!--                                                                                            -->
<!--   + PARTOPT - Kann zur Inaktivierung der Rechenpartitionsgruppenoptimierung verwendet  -->
<!--               werden, die dazu dient, Eingaben an Join-, Aggregations-, und UNION-     -->
<!--               Operationen dynamisch neu zu verteilen, wenn diese Eingaben Ergebnisse   -->
<!--               von Fernabfragen sind.                                                   -->
<!--                                                                                        -->
<!--    + PART - Definiert die in CPG-Optimierungen zu verwendenden Partitionsgruppen.      -->
<!--                                                                                        -->
<!-- ********************************************************************************************-->
<xs:group name="computationalPartitionGroupOptimizationChoices">
   <xs:choice>
      <xs:element name="PARTOPT" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="OPTION" type="optionType" use="optional"/>
      </xs:complexType>
      </xs:element>
      <xs:element name="PART" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="NAME" type="xs:string" use="required"/>
         </xs:complexType>
      </xs:element>
   </xs:choice>
</xs:group>
<!-- ********************************************************************************************-->
<!--  Definition eines Anweisungsprofils.                                                   -->
<!--  Besteht aus einem Anweisungsschlüssel und Optimierungsrichtlinien.                    -->
<!--  Der Anweisungsschlüssel enthält semantische Informationen, die zur Angabe der Anweisung -->
<!--  dienen, für die die Optimierungsrichtlinien gelten. Das optionale Attribut ID stellt    -->
<!--  einen Namen für das Anweisungsprofil in EXPLAIN-Ausgaben bereit.                      -->
<!-- ********************************************************************************************-->
<xs:complexType name="statementProfileType">
   <xs:sequence>
      <!-- Anweisungsschlüsselelement -->
      <xs:element name="STMTKEY" type="statementKeyType"/>
      <xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
   </xs:sequence>
   <!-- Attribut ID. Dient in EXPLAIN-Ausgaben zur Angabe, dass das Anweisungsprofil verwendet wurde. -->
<xs:attribute name="ID" type="xs:string" use="optional"/>
</xs:complexType>
<!--*******************************************************************************************-->
<!--  Definition des Anweisungsschlüssels. Der Anweisungsschlüssel stellt semantische        -->
<!--  Informationen zur Angabe der Anweisung bereit, für die die Optimierungsrichtlinien     -->
<!--  gelten.                                                                                -->
<!-- Der Anweisungsschlüssel besteht aus:                                                    -->
<!--      + Anweisungstext (wie in der Anwendung geschrieben)                               -->
<!--      + Standardschema (zur Auflösung unqualifizierter Tabellennamen in der Anweisung)  -->
<!--      + Funktionspfad (zur Auflösung unqualifizierter Typen und Funktionen in der       -->
<!--        Anweisung)                                                                      -->
<!-- Der Anweisungstext wird in Form von Elementdaten bereitgestellt, während Standardschema -->
<!--  und Funktionspfad über die Elemente SCHEMA und FUNCPATH angegeben werden.             -->
<!--*******************************************************************************************-->
<xs:complexType name="statementKeyType" mixed="true">
   <xs:attribute name="SCHEMA" type="xs:string" use="optional"/>
   <xs:attribute name="FUNCPATH" type="xs:string" use="optional"/>
</xs:complexType>
<!--*******************************************************************************************-->
<!--                                                                                        -->
<!-- Optimierungsrichtlinienelemente können aus allgemeinen Anforderungen, Umschreibe-      -->
<!-- anforderungen, Zugriffsanforderungen oder Joinanforderungen ausgewählt werden.         -->
<!--                                                                                        -->
<!-- Allgemeine Anforderungen beeinflussen den Suchbereich, der die alternativen            -->
<!-- Abfrageumsetzungen, Zugriffsmethoden, Joinmethoden, Joinreihenfolgen und andere        -->
<!-- Optimierungen definiert, die vom Optimierungsprogramm in Betracht gezogen werden.      -->
<!--                                                                                        -->
```

```
<!-- Umschreibeanforderungen beeinflussen die Abfrageumsetzungen, die bei der Bestimmung    -->
<!-- der optimierten Anweisung verwendet werden.                                            -->
<!--                                                                                        -->
<!-- Zugriffsanforderungen beeinflussen die Zugriffsmethoden, die vom aufwandsbasierten     -->
<!-- Optimierungsprogramm in Betracht gezogen werden. Joinanforderungen beeinflussen        -->
<!-- die Joinmethoden und die Joinreihenfolge im Ausführungsplan.                           -->
<!--                                                                                        -->
<!--****************************************************************************************-->
<xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
<xs:complexType name="optGuidelinesType">
   <xs:sequence>
      <xs:group ref="generalRequest" minOccurs="0" maxOccurs="1"/>
      <xs:choice maxOccurs="unbounded">
         <xs:group ref="rewriteRequest" />
         <xs:group ref="accessRequest"/>
         <xs:group ref="joinRequest"/>
      </xs:choice>
   </xs:sequence>
</xs:complexType>
<!--****************************************************************************************** -->
<!-- Auswahlmöglichkeiten für allgemeine Anforderungselemente.                              -->
<!-- REOPT kann zum Überschreiben der Einstellung der Bindeoption REOPT verwendet werden. -->
<!--****************************************************************************************** -->
<xs:group name="generalRequest">
   <xs:sequence>
      <xs:element name="REOPT" type="reoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="DEGREE" type="degreeType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="QRYOPT" type="qryoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="RTS" type="rtsType" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:group>
<!--****************************************************************************************-->
<!--  Auswahlmöglichkeiten für Umschreibeanforderungselemente.                          -->
<!--****************************************************************************************-->
<xs:group name="rewriteRequest">
   <xs:sequence>
      <xs:element name="INLIST2JOIN" type="inListToJoinType" minOccurs="0"/>
      <xs:element name="SUBQ2JOIN" type="subqueryToJoinType" minOccurs="0"/>
      <xs:element name="NOTEX2AJ" type="notExistsToAntiJoinType" minOccurs="0"/>
      <xs:element name="NOTIN2AJ" type="notInToAntiJoinType" minOccurs="0"/>
   </xs:sequence>
</xs:group>
<!--****************************************************************************************** -->
<!--  Auswahlmöglichkeiten für Zugriffsanforderungselemente.                             -->
<!--  TBSCAN - Zugriffsanforderungselement für Tabellensuche                             -->
<!--  IXSCAN - Zugriffsanforderungselement für Indexsuche                                -->
<!--  LPREFETCH - Zugriffsanforderungselement für Vorablesezugriff über Listen           -->
<!--  IXAND - Zugriffsanforderungselement mit logischem Verknüpfen von Indizes über AND  -->
<!--  IXOR - Zugriffsanforderungselement mit logischem Verknüpfen von Indizes über OR    -->
<!--  ACCESS - Gibt an, dass das Optimierungsprogramm die Zugriffsmethode für die Tabelle -->
<!--           auswählen soll.                                                           -->
<!--****************************************************************************************** -->
<xs:group name="accessRequest">
   <xs:choice>
      <xs:element name="TBSCAN" type="tableScanType"/>
      <xs:element name="IXSCAN" type="indexScanType"/>
      <xs:element name="LPREFETCH" type="listPrefetchType"/>
      <xs:element name="IXAND" type="indexAndingType"/>
      <xs:element name="IXOR" type="indexOringType"/>
      <xs:element name="ACCESS" type="anyAccessType"/>
   </xs:choice>
</xs:group>
<!--****************************************************************************************** -->
<!-- Auswahlmöglichkeiten für Joinanforderungslemente.                                  -->
<!-- NLJOIN - Anforderungselement für Joins mit Verschachtelungsschleife (Nested-Loop)  -->
<!-- MSJOIN - Anforderungselement für Mischjoins (Sort-Merge)                           -->
<!-- HSJOIN - Anforderungselement für Hash-Joins                                        -->
<!-- JOIN - Gibt an, dass das Optimierungsprogramm die Joinmethode auswählen soll.      -->
<!--****************************************************************************************** -->
<xs:group name="joinRequest">
   <xs:choice>
      <xs:element name="NLJOIN" type="nestedLoopJoinType"/>
      <xs:element name="HSJOIN" type="hashJoinType"/>
      <xs:element name="MSJOIN" type="mergeJoinType"/>
      <xs:element name="JOIN" type="anyJoinType"/>
   </xs:choice>
</xs:group>
<!--****************************************************************************************** -->
<!--  Allgemeines REOPT-Anforderungselement. Kann die REOPT-Einstellung auf Paket-          -->
```

```
<!--  Datenbank- und Datenbankmanagerebene überschreiben.                                  -->
<!--************************************************************************************** -->
<xs:complexType name="reoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ONCE"/>
            <xs:enumeration value="ALWAYS"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
</xs:complexType>
<!--************************************************************************************-->
<!-- Allgemeines RTS-Anforderungselement zum Aktivieren, Inaktivieren der Echtzeitstatistik-   -->
<!-- erfassung bzw. zur Festlegung eines Zeitbudgets für diese Erfassung.                    -->
<!-- Attribut OPTION: Aktivieren oder Inaktivieren der Echtzeitstatistiken.                  -->
<!-- Attribut TIME: Zeitbudget in Millisekunden für Echtzeitstatistikerfassung.              -->
<!--************************************************************************************** -->
<xs:complexType name="rtsType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="TIME" type="xs:nonNegativeInteger" use="optional"/>
</xs:complexType>
<!--************************************************************************************-->
<!--   Definition einer Anforderung zum Umschreiben von "IN-Listen in Joins"                -->
<!--   Attribut OPTION ermöglicht Aktivierung oder Inaktivierung der Alternative.           -->
<!--   Attribut TABLE ermöglicht Anforderung gewünschter IN-Listenvergleichselemente, die auf -->
<!--   einen bestimmten Tabellenverweis angewendet werden. Attribut COLUMN ermöglicht       -->
<!--   Anforderung für ein bestimmtes IN-Listenvergleichselement.                           -->
<!--************************************************************************************** -->
<xs:complexType name="inListToJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="TABLE" type="xs:string" use="optional"/>
   <xs:attribute name="COLUMN" type="xs:string" use="optional"/>
</xs:complexType>
<!--************************************************************************************-->
<!--  Definition einer Anforderung zur Umschreibung von Unterabfragen in Joins              -->
<!--  Das Attribut OPTION ermöglicht die Aktivierung oder Inaktivierung der Alternative.    -->
<!--************************************************************************************** -->
<xs:complexType name="subqueryToJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
<!--************************************************************************************-->
<!--  Definition einer Anforderung zur Umschreibung von NOT EXISTS-Unterabfragen in Antijoins -->
<!--  Das Attribut OPTION ermöglicht die Aktivierung oder Inaktivierung der Alternative.    -->
<!--************************************************************************************** -->
<xs:complexType name="notExistsToAntiJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
<!--************************************************************************************-->
<!--  Definition einer Anforderung zur Umschreibung von NOT IN-Unterabfragen in Antijoins   -->
<!--  Das Attribut OPTION ermöglicht die Aktivierung oder Inaktivierung der Alternative.    -->
<!--************************************************************************************-->
<xs:complexType name="notInToAntiJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
<!--************************************************************************************-->
<!-- Effektiv die Superklasse, aus der alle Zugriffsanforderungselemente übernehmen.        -->
<!-- Dieser Typ definiert zurzeit die Attribute TABLE und TABID, die zur Bindung einer      -->
<!-- Zugriffsanforderung an einen Tabellenverweis in der Abfrage verwendet werden können.  -->
<!-- Der Wert des Attributs TABLE dient zur Angabe eines Tabellenverweises durch Kennungen  -->
<!-- in der ursprünglichen SQL-Anweisung. Der Wert des Attributs TABID dient zur Angabe eines -->
<!-- Tabellenverweises durch den eindeutigen Korrelationsnamen, der durch die optimierte    -->
<!-- Anweisung bereitgestellt wird. Wenn TABLE und TABID zusammen angegeben werden, wird das -->
<!-- Feld TABID ignoriert. Das Attribut FIRST gibt an, dass der Zugriff der erste Zugriff   -->
<!-- in der Joinreihenfolge für die FROM-Klausel sein soll.                                 -->
<!--************************************************************************************** -->
<xs:complexType name="accessType" abstract="true">
   <xs:attribute name="TABLE" type="xs:string" use="optional"/>
   <xs:attribute name="TABID" type="xs:string" use="optional"/>
   <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
</xs:complexType>
<!--************************************************************************************-->
<!-- Definition eines Zugriffsanforderungselements für Tabellensuche.                    -->
<!--************************************************************************************-->
<xs:complexType name="tableScanType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
</xs:complexType>
```

```
<!-- ********************************************************************************-->
<!-- Definition eines Zugriffsanforderungselements für Indexsuche.                  -->
<!-- Der Indexname ist optional.                                                    -->
<!--******************************************************************************** -->
<xs:complexType name="indexScanType">
   <xs:complexContent>
      <xs:extension base="accessType">
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<!--********************************************************************************-->
<!-- Definition eines Zugriffsanforderungselements für Vorablesezugriff über Listen.-->
<!-- Der Indexname ist optional.                                                    -->
<!--******************************************************************************** -->
<xs:complexType name="listPrefetchType">
   <xs:complexContent>
      <xs:extension base="accessType">
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<!--********************************************************************************-->
<!-- Definition eines Zugriffsanforderungselements für AND-Verknüpfung von Indizes. -->
<!-- Ein Index kann durch das Attribut INDEX angegeben werden. Mehrere Indizes können-->
<!-- durch INDEX-Elemente angegeben werden. Die Angabe von INDEX-Elementen überschreibt-->
<!-- die Attributangabe. Wenn nur ein Index angegeben wird, verwendet das Optimierungs-->
<!-- programm den Index als ersten Index bei der Zugriffsmethode mit AND-Verknüpfung -->
<!-- von Indizes. Weitere Indizes werden nach Aufwandsberechnung gewählt. Wenn mehrere-->
<!-- Indizes angegeben werden, verwendet das Optimierungsprogramm diese exakt in der -->
<!-- angegebenen Reihenfolge. Wenn keine Indizes durch das Attribut INDEX bzw. durch -->
<!-- INDEX-Elemente angegeben werden, wählt das Optimierungsprogramm alle Indizes nach-->
<!-- Aufwandsberechnung aus.                                                        -->
<!--******************************************************************************** -->
<xs:complexType name="indexAndingType">
   <xs:complexContent>
      <xs:extension base="accessType">
         <xs:sequence minOccurs="0">
            <xs:element name="INDEX" type="indexType" minOccurs="2" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<!--********************************************************************************-->
<!-- Definition einer INDEX-Elementmethode. Indexgruppe ist optional. Bei Angabe    -->
<!-- müssen mindestens 2 angegeben werden.                                          -->
<!--******************************************************************************** -->
<xs:complexType name="indexType">
   <xs:attribute name="IXNAME" type="xs:string" use="optional"/>
</xs:complexType>
<!--********************************************************************************-->
<!-- Zur Verwendung für den Zugriff auf abgeleitete Tabellen oder in anderen Fällen,-->
<!-- in denen es nicht auf die Zugriffsmethode ankommt.                             -->
<!--********************************************************************************-->
<xs:complexType name="anyAccessType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
</xs:complexType>
<!--********************************************************************************-->
<!-- Definition eines Zugriffs mit logischer OR-Verknüpfung von Indizes             -->
<!-- Angabe weiterer Details (z. B. Indizes) nicht möglich. Optimierungsprogramm wählt-->
<!-- die Details auf der Grundlage einer Aufwandsberechnung aus.                    -->
<!--******************************************************************************** -->
<xs:complexType name="indexOringType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
</xs:complexType>
<!--********************************************************************************-->
<!-- Effektiv die Superklasse, aus der Joinanforderungselemente übernehmen.         -->
<!-- Dieser Typ definiert zurzeit Joinelementeingaben und das Attribut FIRST.       -->
<!-- Eine Joinanforderung muss genau 2 verschachtelte Unterelemente enthalten. Die Unter- -->
<!-- elemente können entweder eine Zugriffsanforderung oder eine weitere Joinanforderung  -->
<!-- sein. Das erste Unterelement stellt die äußere Tabelle der Joinoperation dar, das-->
<!-- zweite die innere Tabelle. Das Attribut FIRST gibt an, dass das Joinergebnis der -->
<!-- erste Join in Beziehung zu anderen Tabellen in derselben FROM-Klausel sein soll.-->
<!--******************************************************************************** -->
```

```
<xs:complexType name="joinType" abstract="true">
   <xs:choice minOccurs="2" maxOccurs="2">
      <xs:group ref="accessRequest"/>
      <xs:group ref="joinRequest"/>
   </xs:choice>
   <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
</xs:complexType>
<!--****************************************************************************** -->
<!-- Definition einer Zugriffsanforderung für Join mit Verschachtelungsschleife.  -->
<!-- Unterklasse von 'joinType'. Fügt keine Elemente oder Attribute hinzu.        -->
<!--****************************************************************************** -->
<xs:complexType name="nestedLoopJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
<!--******************************************************************************-->
<!--  Definition einer Zugriffsanforderung mit Mischjoin.                       -->
<!-- Unterklasse von 'joinType'. Fügt keine Elemente oder Attribute hinzu.        -->
<!--****************************************************************************** -->
<xs:complexType name="mergeJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
<!--****************************************************************************** -->
<!--  Definition einer Zugriffsanforderung mit Hash-Join.                       -->
<!-- Unterklasse von 'joinType'. Fügt keine Elemente oder Attribute hinzu.        -->
<!--****************************************************************************** -->
<xs:complexType name="hashJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
<!--****************************************************************************** -->
<!-- Der Typ 'anyJoinType' ist eine Unterklasse des binären Joins. Erweitert diesen in    -->
<!-- keiner Weise. Fügt keine Elemente oder Attribute hinzu.                      -->
<!--****************************************************************************** -->
<xs:complexType name="anyJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
<!--******************************************************************************-->
<!--  Zulässige Werte für ein Attribut OPTION.                                  -->
<!--****************************************************************************** -->
<xs:simpleType name="optionType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="ENABLE"/>
      <xs:enumeration value="DISABLE"/>
   </xs:restriction>
</xs:simpleType>
<!--******************************************************************************-->
<!--   Definition von 'qryoptType': Zulässig sind nur die Werte 0, 1, 2, 3, 5, 7 und 9.      -->
<!--******************************************************************************-->
<xs:complexType name="qryoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="0"/>
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
            <xs:enumeration value="3"/>
            <xs:enumeration value="5"/>
            <xs:enumeration value="7"/>
            <xs:enumeration value="9"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
</xs:complexType>
<!--******************************************************************************-->
<!-- Definition von 'degreeType': Beliebiger Wert zwischen 1 und 32767 bzw. die             -->
<!-- Zeichenfolgen ANY oder '-1'                                                  -->
<!--****************************************************************************** -->
<xs:simpleType name="intStringType">
   <xs:union>
      <xs:simpleType>
         <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"></xs:minInclusive>
```

```
            <xs:maxInclusive value="32767"></xs:maxInclusive>
         </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ANY"/>
            <xs:enumeration value="-1"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:union>
</xs:simpleType>
<xs:complexType name="degreeType">
 <xs:attribute name="VALUE" type="intStringType"></xs:attribute>
</xs:complexType>
</xs:schema>
```

XML-Schema für das Element OPTPROFILE: Das Element OPTPROFILE ist das Stammelement (Root) eines Optimierungsprofils. Das Element OPTPROFILE ist wie folgt definiert:

XML-Schema

```
<xs:element name="OPTPROFILE">
    <xs:complexType>
       <xs:sequence>
          <xs:element name="OPTGUIDELINES" type="globalOptimizationGuidelinesType"
                    minOccurs="0"/>
          <xs:element name="STMTPROFILE" type="statementProfileType" minOccurs="0"
                    maxOccurs="unbounded"/>
       </xs:sequence>
       <xs:attribute name="VERSION" use="optional">
          <xs:simpleType>
             <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="9.1.0.0"/>
             </xs:restriction>
          </xs:simpleType>
       </xs:attribute>
    </xs:complexType>
 </xs:element>
```

Beschreibung

Das optionale Unterelement OPTGUIDELINES definiert den Abschnitt für globale Optimierungsrichtlinien des Optimierungsprofils. Jede Unterelement STMTPROFILE definiert einen Anweisungsprofilabschnitt. Das Attribut VERSION gibt das aktuelle Schema für Optimierungsprofile an, unter dem das jeweilige Optimierungsprofil erstellt und auf Gültigkeit geprüft wurde.

XML-Schema für das globale OPTGUIDELINES-Element: Der Typ *globalOptimizationGuidelinesType* definiert das Format eines globalen Elements OPTGUIDELINES.

XML-Schema

```
<xs:complexType name="globalOptimizationGuidelinesType">
   <xs:sequence>
      <xs:group ref="MQTOptimizationChoices"/>
      <xs:group ref="computationalPartitionGroupOptimizationChoices"/>
      <xs:group ref="generalRequest"/>
   </xs:sequence>
</xs:complexType>
```

Beschreibung

Globale Optimierungsrichtlinien können mit Elementen aus der Gruppe 'MQTOptimizationChoices', mit Elementen aus der Gruppe 'computationalPartitionGroupChoices' oder mit dem Element REOPT definiert werden.

Die Elemente der Gruppe 'MQTOptimizationChoices', die in „Auswahlmöglichkeiten für die MQT-Optimierung" definiert sind, können zur Beeinflussung der MQT-Substitution verwendet werden. Die Elemente der Gruppe 'computationalPartitionGroupOptimizationChoices', die in „Auswahlmöglichkeiten für die Rechenpartitionsgruppenoptimierung" auf Seite 449 definiert sind, können zur Beeinflussung der Rechenpartitionsgruppenoptimierung verwendet werden.

Die Rechenpartitionsgruppenoptimierung beinhaltet eine dynamische Neuverteilung von Daten, die aus fernen Datenquellen gelesen werden. Sie ist nur in Konfigurationen mit partitionierten und föderierten Datenbanken anwendbar.

Das Element REOPT, das in „Richtlinien für die globale REOPT-Optimierung" auf Seite 450 definiert ist, kann zur Beeinflussung des Optimierungszeitpunkts für Anweisungen verwendet werden, die Variablen enthalten.

Auswahlmöglichkeiten für die MQT-Optimierung: Die Gruppe *MQTOptimizationChoices* definiert einen Satz von Elementen, die zur Beeinflussung der Optimierung mit MQTs (Materialized Query Tables) verwendet werden können. Insbesondere können die Elemente dazu verwendet werden, die Prüfung, ob eine MQT-Substitution in Betracht kommt, zu aktivieren bzw. zu inaktivieren, oder den kompletten Satz von MQTs anzugeben, der vom Optimierungsprogramm in Betracht gezogen werden soll.

XML-Schema

```
<xs:group name="MQTOptimizationChoices">
   <xs:choice>
      <xs:element name="MQTOPT" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="OPTION" type="optionType" use="optional"/>
         </xs:complexType>
      </xs:element>
     <xs:element name="MQT" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:attribute name="NAME" type="xs:string" use="required"/>
         </xs:complexType>
     </xs:element>
   </xs:choice>
</xs:group>
```

Beschreibung

Das MQTOPT-Element dient zur Aktivierung bzw. Inaktivierung der Berücksichtigung der MQT-Optimierung. Das Attribut OPTION kann die Werte ENABLE oder DISABLE haben. Der Standardwert des Attributs OPTION ist ENABLE.

Alternativ können null oder mehr MQT-Elemente angegeben werden. Das Attribut NAME eines MQT-Elements gibt eine MQT an, die vom Optimierungsprogramm in Betracht gezogen werden soll. Die Regeln für die Bildung eines Verweises auf eine MQT im Attribut NAME sind die gleichen wie die für die Bildung von Verweisen auf exponierte Tabellennamen. Wenn ein oder mehrere MQT-Elemente angebeben werden, werden nur diese MQTs vom Optimierungsprogramm berücksichtigt. Die Entscheidung, eine MQT-Substitution mit einer oder mehreren der

angegebenen MQTs durchzuführen, erfolgt weiterhin auf der Basis einer Aufwandsberechnung.

Beispiele

Das folgende Beispiel zeigt, wie die MQT-Optimierung inaktiviert wird:

```
<OPTGUIDELINES>
     <MQTOPT OPTION='DISABLE'/>
</OPTGUIDELINES>
```

Das nachfolgende Beispiel zeigt, wie die MQT-Optimierung auf die MQTs ″Tpcd-.PARTSMQT″ und ″COLLEGE.STUDENTS″ eingegrenzt wird:

```
<OPTGUIDELINES>
     <MQT NAME='Tpcd.PARTSMQT'/>
     <MQT NAME='COLLEGE.STUDENTS'/>
<OPTGUIDELINES>
```

Auswahlmöglichkeiten für die Rechenpartitionsgruppenoptimierung: Die Gruppe *computationalPartitionGroupOptimizationChoices* definiert einen Satz von Elementen, die zur Beeinflussung der Rechenpartitionsgruppenoptimierung verwendet werden können. Insbesondere können die Elemente dazu verwendet werden, die Rechengruppenoptimierung zu aktivieren bzw. zu inaktivieren oder die spezielle Partitionsgruppe anzugeben, die zur Rechenpartitionsgruppenoptimierung verwendet werden soll.

XML-Schema

```
<xs:group name="computationalPartitionGroupOptimizationChoices">
   <xs:choice>
      <xs:element name="PARTOPT" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="OPTION" type="optionType" use="optional"/>
         </xs:complexType>
      </xs:element>
   <xs:element name="PART" minOccurs="0" maxOccurs="1">
      <xs:complexType>
         <xs:attribute name="NAME" type="xs:string" use="required"/>
      </xs:complexType>
   </xs:element>
  </xs:choice>
</xs:group>
```

Beschreibung

Das PARTOPT-Element dient zur Aktivierung bzw. Inaktivierung der Berücksichtigung der Rechenpartitionsgruppenoptimierung. Das Attribut OPTION kann die Werte ENABLE oder DISABLE haben. Der Standardwert des Attributs OPTION ist ENABLE.

Alternativ kann ein PART-Element verwendet werden, um die Partitionsgruppe anzugeben, die für die Rechenpartitionsgruppenoptimierung verwendet werden soll. Das Attribut NAME des PART-Elements gibt die Partitionsgruppe an und muss eine vorhandene Partitionsgruppe enthalten. Die Entscheidung, eine dynamische Neuverteilung unter Verwendung der angegebenen Partitionsgruppe auszuführen, erfolgt auf der Basis einer Aufwandsberechnung.

Beispiele

Das folgende Beispiel zeigt, wie die Rechenpartitionsgruppenoptimierung inaktiviert wird:

```
<OPTGUIDELINES>
     <PARTOPT OPTION='DISABLE'/>
</OPTGUIDELINES>
```

Das nächste Beispiel zeigt, wie angegeben wird, dass die Partitionsgruppe WORKPART für die Rechenpartitionsgruppenoptimierung zu verwenden ist:

```
<OPTGUIDELINES>
     <MQT NAME='Tpcd.PARTSMQT'/>
     <PART NAME='WORKPART'/>
<OPTGUIDELINES>
```

Richtlinien für die globale REOPT-Optimierung: Die globale REOPT-Optimierungsrichtlinie steuert, wann eine Optimierung für DML-Anweisungen erfolgt, die Variablen (Hostvariablen, Parametermarken, globale Variablen oder Sonderregister) enthalten. Die Beschreibung und Syntax des REOPT-Elements ist für globale Optimierungsrichtlinien und Anweisungsoptimierungsrichtlinien identisch. Weitere Informationen finden Sie in „REOPT-Anforderungen" auf Seite 454.

XML-Schema für das Element STMTPROFILE: Der Typ *statementProfileType* definiert das Format des Elements STMTPROFILE.

```
XML-Schema

   <xs:complexType name="statementProfileType">
   <xs:sequence>
      <xs:element name="STMTKEY" type="statementKeyType"/>
      <xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
         </xs:sequence>
   <xs:attribute name="ID" type="xs:string" use="optional"/>
</xs:complexType>
```

Beschreibung

Ein Anweisungsprofil gibt Optimierungsrichtlinien für eine bestimmte Anweisung an, für die das Optimierungsprofil wirksam ist. Ein Anweisungsprofil besteht aus folgenden Komponenten:

- Anweisungsschlüssel: Ein Optimierungsprofil kann für mehr als eine Anweisung in einer Anwendung wirksam sein. Das Optimierungsprogramm gleicht mithilfe des Anweisungsschlüssels jedes Anweisungsprofil automatisch mit der richtigen Anweisung der Anwendung ab. Dieses Verfahren bietet dem Benutzer die Möglichkeit, Optimierungsrichtlinien für eine Anweisung anzugeben, ohne die Anwendung zu bearbeiten.

  Der Anweisungsschlüssel besteht aus dem Text der Anweisung, wie er in der Anwendung geschrieben ist, sowie aus anderen Informationskomponenten, die zur eindeutigen Kennzeichnung der richtigen Anwendungsanweisung erforderlich sind. Der Anweisungsschlüssel wird durch das Unterelement STMTKEY dargestellt, das in „XML-Schema für das STMTKEY-Element" auf Seite 451 beschrieben wird.
- Optimierungsrichtlinien auf Anweisungsebene: Dieser Abschnitt des Anweisungsprofils gibt die Optimierungsrichtlinien an, die für die durch den Anweisungsschlüssel angegebene Anweisung wirksam sind. Weitere Informationen finden Sie in „XML-Schema für das Element OPTGUIDELINES auf Anweisungsebene" auf Seite 452.
- Anweisungsprofilname: Ein vom Benutzer angegebener Name, der in Diagnoseausgaben zur Angabe des Anweisungsprofils dient, das zur Optimierung einer Anweisung verwendet wurde.

XML-Schema für das STMTKEY-Element: Der Typ *statementKeyType* definiert das Format des STMTKEY-Elements.

```
XML-Schema

      <xs:complexType name="statementKeyType" mixed="true">
            <xs:attribute name="SCHEMA" type="xs:string" use="optional"/>
            <xs:attribute name="FUNCPATH" type="xs:string" use="optional"/>
   </xs:complexType>
   </xs:schema>
```

Beschreibung

Die Komponente des Anweisungstexts des Anweisungsschlüssels wird in Form von Daten zwischen dem Start- und dem Endtag des Elements STMTKEY angegeben.

Das optionale Attribut SCHEMA kann verwendet werden, um die Komponente des Standardschemas des Anweisungsschlüssels anzugeben.

Das optionale Attribut FUNCPATH kann verwendet werden, um die Komponente des Funktionspfads des Anweisungsschlüssels anzugeben. Es können mehrere Funktionspfade durch Kommata getrennt angegeben werden. Die angegebenen Funktionspfade müssen exakt mit den im Kompilierungsschlüssel angegebenen Funktionspfaden übereinstimmen.

Beispiele

Das folgende Beispiel eines Anweisungsschlüssels entspricht der Anweisung ″select * from orders where foo(orderkey) > 20″, sofern der Kompilierungsschlüssel das Standardschema ″COLLEGE″ und den Funktionspfad ″SYSIBM,SYSFUN,SYSPROC,DAVE″ enthält:

```
<STMTKEY SCHEMA='COLLEGE' FUNCPATH='SYSIBM,SYSFUN,SYSPROC,DAVE'>
   <![CDATA[select * from orders" where foo(orderkey) > 20]]>
   </STMTKEY>
```

Anmerkung: Der Anweisungstext wird innerhalb eines CDATA-Abschnitts (beginnend mit <![CDATA[und endend mit]]>) angeben, weil er das XML-Sonderzeichen '>' enthält.

Abgleich von Anweisungs- und Kompilierungsschlüsseln: Der Anweisungsschlüssel dient zur Angabe der jeweiligen Anwendungsanweisung, auf die die Optimierungsrichtlinien der Anweisungsebene angewendet werden. Wenn DB2 eine statische oder dynamische SQL-Anweisung kompiliert, steuert die Einstellung bestimmter Parameter (die durch Sonderregister sowie durch Binde- und Vorkompilierungsoptionen festgelegt werden), wie die Anweisung semantisch vom Compiler interpretiert wird. Die SQL-Anweisung und die Einstellungen dieser speziellen Parameter des SQL-Compilers bilden zusammen den so genannten *Kompilierungsschlüssel*. Jede Komponente des Anweisungsschlüssels entspricht einer Komponente des Kompilierungsschlüssels.

Ein *Anweisungsschlüssel* setzt sich aus folgenden Komponenten zusammen:

- Anweisungstext: Der Text der Anweisung, wie er in der Anwendung geschrieben wurde.
- Standardschema: Der Schemaname, der als implizites Qualifikationsmerkmal für nicht qualifizierte Tabellennamen verwendet wird. Diese Komponente ist optional. Sie sollte jedoch angegeben werden, wenn der Anweisungstext nicht qualifizierte Tabellennamen enthält.

- Funktionspfad: Der Funktionspfad wird bei der Auflösung von nicht qualifizierten Funktions- und Datentypverweisen verwendet. Diese Komponente ist optional. Sie sollte jedoch angegeben werden, wenn nicht qualifizierte benutzerdefinierte Funktionen oder benutzerdefinierte Datentypen in der Anweisung enthalten sind.

Wenn DB2 eine SQL-Anweisung kompiliert und ein aktives Optimierungsprofil findet, versucht DB2, jeden Anweisungsschlüssel im Optimierungsprofil mit dem aktuellen Kompilierungsschlüssel abzugleichen. Ein Anweisungsschlüssel und ein Kompilierungsschlüssel werden als übereinstimmend betrachtet, wenn jede angegebene Komponente des Anweisungsschlüssels mit der entsprechenden Komponente des Kompilierungsschlüssels übereinstimmt. Wenn eine Komponente des Anweisungsschlüssels nicht angegeben ist, wird die ausgelassene Komponente standardmäßig als übereinstimmend betrachtet. Effektiv wird jede nicht angegebene Komponente des Anweisungsschlüssels als Platzhalter behandelt und so interpretiert, dass sie mit der entsprechenden Komponente eines beliebigen Kompilierungsschlüssels übereinstimmt.

Wenn DB2 einen Anweisungsschlüssel findet, der mit dem aktuellen Kompilierungsschlüssel übereinstimmt, wird die Suche gestoppt. Daher wird, wenn mehrere Anweisungsprofile in einem Optimierungsprofil vorhanden sind, deren Anwendungsschlüssel mit dem aktuellen Kompilierungsschlüssel übereinstimmt, nur das erste dieser Anweisungsprofile verwendet (entsprechend der Dokumentreihenfolge). In diesem Fall wird außerdem keine Fehlernachricht oder Warnung ausgegeben.

XML-Schema für das Element OPTGUIDELINES auf Anweisungsebene: Das Element OPTGUIDELINES eines Anweisungsoptimierungsprofils definiert die wirksamen Optimierungsrichtlinien für die Anweisung, die durch den zugeordneten Anweisungsschlüssel (STMTKEY) angegeben wird. Das Element OPTGUIDELINES ist wie folgt mit dem Typ *optGuidelinesType* definiert:

```
XML-Schema

<xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
<xs:complexType name="optGuidelinesType">
   <xs:sequence>
      <xs:group ref="general request" minOccurs="0" maxOccurs="1"/>
      <xs:choice maxOccurs="unbounded">
         <xs:group ref="rewriteRequest"/>
         <xs:group ref="accessRequest"/>
         <xs:group ref="joinRequest"/>
      </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Beschreibung

Die Gruppe 'optGuidelinesType' definiert den Satz gültiger Unterelemente des Elements OPTGUIDELINES. Jedes Unterelement wird vom DB2-Optimierungsprogramm als Optimierungsrichtlinie verstanden. Unterelemente können entweder als allgemeines Anforderungselement, Umschreibeanforderungselement, Zugriffsanforderungselement oder Joinanforderungselement kategorisiert werden.

- Allgemeine Anforderungselemente dienen zur Angabe allgemeiner Optimierungsrichtlinien. Allgemeine Optimierungsrichtlinien können zum Ändern des Suchbereichs des Optimierungsprogramms verwendet werden.
- Umschreibeanforderungselemente dienen zur Angabe von Optimierungsrichtlinien für das Umschreiben von Abfragen. Mit Abfrageumschreiberichtlinien

können die Abfrageumsetzungen beeinflusst werden, die bei der Bestimmung der optimierten Anweisung angewendet werden.
- Zugriffs- und Joinanforderungselemente stellen Richtlinien für die Planoptimierung dar. Mit Planoptimierungsrichtlinien können die Zugriffsmethoden, Joinmethoden und Joinreihenfolgen beeinflusst werden, die im Ausführungsplan für die optimierte Anweisung verwendet werden.

Anmerkung: Optimierungsrichtlinien, die in einem Anweisungsprofil angegeben sind, haben Vorrang vor den Richtlinien, die im globalen Abschnitt eines Optimierungsprofils angegeben sind.

XML-Schema für allgemeine Optimierungsrichtlinien: Allgemeine Optimierungsrichtlinien dienen zur Angabe von Richtlinien, die sich nicht speziell auf eine bestimmte Phase des Optimierungsprozesses beziehen. Allgemeine Optimierungsrichtlinien können zum Ändern des Suchbereichs des Optimierungsprogramms verwendet werden. Sie bestehen aus der Gruppe *generalRequest*.

```
<!--******************************************************************************** --> \
<!-- Auswahlmöglichkeiten für allgemeine Anforderungselemente.                      --> \
<!-- REOPT kann zum Überschreiben der Einstellung der Bindeoption REOPT verwendet werden. --> \
<!--******************************************************************************** --> \
<xs:group name="generalRequest">
   <xs:sequence>
      <xs:element name="REOPT" type="reoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="DEGREE" type="degreeType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="QRYOPT" type="qryoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="RTS" type="rtsType" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:group>
```

Beschreibung

Allgemeine Anforderungselemente können zur Definition allgemeiner Optimierungsrichtlinien verwendet werden. Allgemeine Optimierungsrichtlinien beeinflussen den Optimierungssuchbereich und können daher die Anwendbarkeit von Optimierungsrichtlinien für das Umschreiben von Abfragen und von aufwandsbasierten Optimierungsrichtlinien beeinflussen.

DEGREE-Anforderungen: Das allgemeine Anforderungselement DEGREE kann zum Überschreiben der Einstellung der Bindeoption, des Datenbankkonfigurationsparameters 'dft_degree' oder einer vorherigen Anweisung SET CURRENT DEGREE verwendet werden. Das allgemeine Anforderungselement DEGREE wird nur berücksichtigt, wenn der Datenbankmanager für partitionsinterne Parallelität konfiguriert ist. Es wird eine Warnung zurückgegeben, wenn der Datenbankmanager nicht für partitionsinterne Parallelität konfiguriert ist. Das allgemeine Anforderungselement DEGREE ist durch den Typ 'degreeType' wie folgt definiert:

XML-Schema

```
<xs:simpleType name="intStringType">
   <xs:union>
      <xs:simpleType>
         <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"></xs:minInclusive>
            <xs:maxInclusive value="32767"></xs:maxInclusive>
         </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ANY"/>
            <xs:enumeration value="-1"/>
         </xs:restriction>
```

```
      </xs:simpleType>
   </xs:union>
</xs:simpleType>

<xs:complexType name="degreeType">
   <xs:attribute name="VALUE"
      type="intStringType"></xs:attribute>
</xs:complexType>
```

Beschreibung

Das allgemeine Anforderungselement DEGREE hat ein erforderliches Attribut VALUE, das die Einstellung der Option DEGREE angibt. Das Attribut kann einen ganzzahligen Wert von 1 bis 32767 oder die Zeichenfolgewerte ″-1″ bzw. ″ANY″ annehmen. Der Wert -1 (bzw. damit äquivalent ″ANY″) gibt an, dass der verwendete Grad der Parallelität durch DB2 bestimmt wird. Der Wert 1 gibt an, dass die Abfrage keine partitionsinterne Parallelität verwenden soll.

QRYOPT-Anforderungen: Das allgemeine Anforderungselement QRYOPT kann zum Überschreiben der Einstellung der Bindeoption, des Datenbankkonfigurationsparameters 'dft_qryopt' oder einer vorherigen Anweisung SET CURRENT QUERY OPTIMIZATION verwendet werden. Das allgemeine Anforderungselement QRYOPT ist durch den Typ 'qryoptType' wie folgt definiert:

XML-Schema

```
<xs:complexType name="qryoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="0"/>
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
            <xs:enumeration value="3"/>
            <xs:enumeration value="5"/>
            <xs:enumeration value="7"/>
            <xs:enumeration value="9"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
</xs:complexType>
```

Beschreibung

Das allgemeine Anforderungselement QRYOPT hat ein erforderliches Attribut VALUE, das die Einstellung der Option QRYOPT angibt. Das Attribut kann jeden Wert aus der Liste annehmen: 0, 1, 2, 3, 5, 7 und 9. Detaillierte Informationen zu jeder der möglichen Einstellungen finden Sie in *Systemverwaltung: Optimierung*.

REOPT-Anforderungen: Das allgemeine Anforderungselement REOPT kann zum Überschreiben der Einstellung der Bindeoption REOPT verwendet werden. Die Bindeoption REOPT beeinflusst die Optimierung von Anweisungen mit Parametermarken oder Hostvariablen. Das allgemeine Anforderungselement REOPT ist durch den Typ 'reoptType' wie folgt definiert:

XML-Schema

```
<xs:complexType name="reoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ONCE"/>
            <xs:enumeration value="ALWAYS"/>
```

```
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:complexType>
```

Beschreibung

Das allgemeine Anforderungselement REOPT hat ein erforderliches Attribut VALUE, das die Einstellung der Option REOPT angibt. Das Attribut kann entweder den Wert ONCE oder den Wert ALWAYS annehmen. Der Wert ONCE gibt an, dass die Anweisung für den ersten Satz von Werten für Hostvariablen bzw. Parametermarken optimiert werden soll. Der Wert ALWAYS gibt an, dass die Anweisung für jeden Satz von Werten für Hostvariablen bzw. Parametermarken optimiert werden soll. Detaillierte Informationen zur Funktionsweise jeder der möglichen Einstellungen für die Bindeoption REOPT finden Sie in *Systemverwaltung: Optimierung*.

RTS-Anforderungen:

Das allgemeine RTS-Anforderungselement kann zur Aktivierung bzw. Inaktivierung der Echtzeitstatistikerfassung (RTS - Real-Time Statistics) verwendet werden. Es kann außerdem dazu verwendet werden, den Zeitaufwand für die Echtzeitstatistikerfassung zu begrenzen. Für bestimmte Abfragen oder Auslastungen kann es wünschenswert sein, die Zeit, die zur Erfassung von Echtzeitstatistiken aufgewendet wird, ganz einzusparen oder zu begrenzen, um bei der Kompilierung von Anweisungen zusätzlichen Aufwand zu vermeiden.

```
<!--*******************************************************************************************--> \
<!-- Allgemeine RTS-Anforderung zum Aktivieren oder Inaktivieren der Echtzeitstatistik-       --> \
<!-- erfassung bzw. zur Festlegung eines Zeitbudgets für diese Erfassung.                     --> \
<!-- Attribut OPTION: Aktivieren oder Inaktivieren der Echtzeitstatistiken.                   --> \
<!-- Attribut TIME: Zeitbudget in Millisekunden für Echtzeitstatistikerfassung.               --> \
<!--******************************************************************************************* --> \
<xs:complexType name="rtsType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="TIME" type="xs:nonNegativeInteger" use="optional"/>
</xs:complexType>
```

Beschreibung

Das allgemeine Anforderungselement für die Echtzeitstatistikerfassung (RTS) hat zwei optionale Attribute.

- Das Attribut OPTION dient zum Aktivieren oder Inaktivieren der Echtzeitstatistikerfassung. Es kann die Werte ENABLE oder DISABLE annehmen. ENABLE ist der Standardwert, wenn keine Option angegeben wird.
- Das Attribut TIME gibt den maximalen Zeitaufwand in Millisekunden an, der für die Erfassung von Echtzeitstatistiken bei der Anwendungskompilierung für eine einzelne Anweisung aufzuwenden ist.

Wenn für das Attribut OPTION der Wert ENABLE angegeben wird, muss die automatische Statistikerfassung und die Echtzeitstatistikerfassung über die entsprechenden Konfigurationsparameter aktiviert sein. Ansonsten wird die Optimierungsrichtlinie nicht angewendet, und Sie empfangen eine Warnung SQL0437W (Ursachencode 13).

Die RTS-Anforderung im folgenden Beispiel aktiviert die Erfassung von Echtzeitstatistiken und begrenzt den Zeitaufwand für diese Erfassung auf 3,5 Sekunden.

```
<RTS OPTION="ENABLE" TIME="350" />
```

XML-Schema für Richtlinien zum Umschreiben von Abfragen: Optimierungsrichtlinien zum Umschreiben von Abfragen beeinflussen die Optimierungsphase der Abfrageumschreibung. Sie bestehen aus der Gruppe *rewriteRequest*.

Die Gruppe *rewriteRequest* definiert den Satz gültiger Auswahlmöglichkeiten für Umschreibeanforderungselemente. Die Umschreibeanforderungen sind *INLIST2JOIN*, *SUBQ2JOIN*, *NOTEX2AJ* und *NOTIN2AJ*:

```
XML-Schema

      <xs:group name="rewriteRequest">
      <xs:sequence>
                  <xs:element name="INLIST2JOIN" type="inListToJoinType" minOccurs="0"/>
                  <xs:element name="SUBQ2JOIN"   type="subqueryToJoinType" minOccurs="0"/>
                  <xs:element name="NOTEX2AJ"    type="notExistsToAntiJoinType" minOccurs="0"/>
                  <xs:element name="NOTIN2AJ"    type="notInToAntiJoinType" minOccurs="0"/>
      </xs:sequence>
   </xs:group>
```

Beschreibung

- INLIST2JOIN: Aktiviert oder inaktiviert die Umsetzung von IN-Listen in Joins in der Umschreibephase. Dieses Element kann als Richtlinie auf Anweisungsebene verwendet werden, um für alle IN-Listenvergleichselemente in einer Abfrage die Prüfung zu aktivieren bzw. zu inaktivieren, ob sie für eine Umsetzung von der IN-Liste in einen Join in Betracht kommen. Es kann außerdem als Richtlinie auf Vergleichselementebene verwendet werden, um für ein angegebenes IN-Listenvergleichselement die Prüfung zu aktivieren bzw. zu inaktivieren, ob es für die Umsetzung von einer IN-Liste in einen Join in Betracht kommt. Wenn Richtlinien sowohl auf Anweisungs- als auch auf Vergleichselementebene angegeben werden, überschreibt die Richtlinie auf Vergleichselementebene die Richtlinie auf Anwendungsebene.
- SUBQ2JOIN: Aktiviert oder inaktiviert die Umsetzung von Unterfragen in Joins in der Umschreibephase. Dieses Element kann nur als Richtlinie auf Anweisungsebene verwendet werden, um für alle Unterabfragen in einer Abfrage die Prüfung zu aktivieren bzw. zu inaktivieren, ob sie für die Umsetzung von Unterfragen in Joins in der Umschreibephase in Betracht kommen.
- NOTEX2AJ: Aktiviert oder inaktiviert die Umsetzung von NOT EXISTS-Unterabfragen in Antijoins in der Umschreibephase. Dieses Element kann nur als Richtlinie auf Anweisungsebene verwendet werden, um für alle NOT EXISTS-Unterabfragen in einer Abfrage die Prüfung zu aktivieren bzw. zu inaktivieren, ob sie für die Umsetzung von NOT EXISTS-Unterfragen in Antijoins in der Umschreibephase in Betracht kommen.
- NOTIN2AJ: Aktiviert oder inaktiviert die Umsetzung von NOT IN-Unterabfragen in Antijoins in der Umschreibephase. Dieses Element kann nur als Richtlinie auf Anweisungsebene verwendet werden, um für alle NOT IN-Unterabfragen in einer Abfrage die Prüfung zu aktivieren bzw. zu inaktivieren, ob sie für die Umsetzung von NOT IN-Unterfragen in Antijoins in der Umschreibephase in Betracht kommen.

Umschreibeanforderung INLIST2JOIN: Das Anforderungselement *INLIST2JOIN* kann zur Aktivierung bzw. Inaktivierung der Umsetzung von IN-Listenvergleichselementen in Joins in der Umschreibephase verwendet werden. Es kann als Richtlinie auf Anweisungsebene oder Vergleichselementebene angegeben werden. Bei Angabe als Richtlinie auf Vergleichselementebene ist nur eine Optimierungsrichtlinie mit der Option ENABLE in einer Abfrage zulässig.

Es ist durch den komplexen Typ *inListToJoinType* wie folgt definiert:

```
XML-Schema

     <xs:complexType name="inListToJoinType">
          <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
          <xs:attribute name="TABLE"  type="xs:string"  use="optional"/>
          <xs:attribute name="COLUMN" type="xs:string"  use="optional"/>
   </xs:complexType>
```

Beschreibung

Das Element *INLIST2JOIN* besitzt drei optionale Attribute und kein Unterelement. Das Attribut OPTION hat den Typ *optionType*, der wiederum den Wert „ENABLE" oder „DISABLE" hat. Wenn kein Attribut OPTION angegeben wird, lautet der Standardwert „ENABLE". Das Attribut für den Tabellennamen (TABLE) und das Attribut für den Spaltennamen (COLUMN) werden zur Angabe des IN-Listen-vergleichselements verwendet, das sich auf die angegebene Tabelle und Spalte bezieht. Wenn das Tabellennamenattribut und das Spaltennamenattribut nicht angegeben werden oder beide Attribute als leere Zeichenfolge „" angegeben werden, wird das Element wie eine Richtlinie auf Anweisungsebene behandelt. Wenn das Tabellennamenattribut angegeben wird oder beide Attribute (Tabellenname und Spaltenname) angegeben werden, wird das Element wie eine Richtlinie auf Vergleichselementebene behandelt. Wenn das Tabellennamenattribut nicht oder als leere Zeichenfolge „" angegeben wird, das Spaltennamenattribut jedoch angegeben wird, wird die Warnung SQLO437W mit Ursachencode 13 ausgegeben und die Optimierungsrichtlinie ignoriert.

Umschreibeanforderungen mit dem Element NOTEX2AJ: Das Anforderungselement *NOTEX2AJ* kann zur Aktivierung bzw. Inaktivierung der Umsetzung von NOT EXITS-Unterabfragen in Antijoins in der Umschreibephase verwendet werden. Es kann nur als Richtlinie auf Anweisungsebene angegeben werden.

Es ist durch den komplexen Typ *notExisitsToAntiJoinType* wie folgt definiert:

```
XML-Schema

     <xs:complexType name="notExistsToAntiJoinType">
          <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   </xs:complexType>
```

Beschreibung

Das Element *NOTEX2AJ* besitzt ein optionales Attribut und kein Unterelement. Das Attribut OPTION hat den Typ *optionType*, der wiederum den Wert „ENABLE" oder „DISABLE" hat. Wenn kein Attribut OPTION angegeben wird, lautet der Standardwert „ENABLE".

Umschreibeanforderungen mit dem Element NOTIN2AJ: Das Anforderungselement *NOTIN2AJ* kann zur Aktivierung bzw. Inaktivierung der Umsetzung von NOT IN-Unterabfragen in Antijoins in der Umschreibephase verwendet werden. Es kann nur als Richtlinie auf Anweisungsebene angegeben werden.

Es ist durch den komplexen Typ *notInToAntiJoinType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="notInToAntiJoinType">
           <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   </xs:complexType>
```

Beschreibung

Das Element *NOTIN2AJ* besitzt ein optionales Attribut und kein Unterelement. Das Attribut OPTION hat den Typ *optionType*, der wiederum den Wert „ENABLE" oder „DISABLE" hat. Wenn kein Attribut OPTION angegeben wird, lautet der Standardwert „ENABLE".

Umschreibeanforderungen mit dem Element SUBQ2JOIN: Das Anforderungselement *SUBQ2JOIN* kann zur Aktivierung bzw. Inaktivierung der Umsetzung von Unterfragen in Joins in der Umschreibephase verwendet werden. Es kann nur als Richtlinie auf Anweisungsebene angegeben werden.

Es ist durch den komplexen Typ *subqueryToJoinType* wie folgt definiert:

XML-Schema

```
    <xs:complexType name="subqueryToJoinType">
          <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
  </xs:complexType>
```

Beschreibung

Das Element *SUBQ2JOIN* besitzt ein optionales Attribut und kein Unterelement. Das Attribut OPTION hat den Typ *optionType*, der wiederum den Wert „ENABLE" oder „DISABLE" hat. Wenn kein Attribut OPTION angegeben wird, lautet der Standardwert „ENABLE".

XML-Schema für Planoptimierungsrichtlinien: Planoptimierungsrichtlinien lassen sich in Zugriffsanforderungen und Joinanforderungen untergliedern:

- Zugriffsanforderung: Eine Zugriffsanforderung gibt eine gewünschte Zugriffsmethode für einen Tabellenverweis an.
- Joinanforderung: Eine Joinanforderung gibt eine gewünschte Methode und Reihenfolge einer Joinoperation an. Joinanforderungen bestehen wiederum aus weiteren Zugriffs- oder Joinanforderungen.

Die meisten Auswahlmöglichkeiten für Zugriffsanforderungen entsprechen den Datenzugriffsmethoden des Optimierungsprogramms, wie zum Beispiel Tabellensuche, Indexsuche und Vorablesezugriff über Listen. Die meisten verfügbaren Joinanforderungen entsprechen den Joinmethoden des Optimierungsprogramms, wie zum Beispiel Join mit Verschachtelungsschleife (Nested Loop Join), Hash-Join und Mischjoin (Merge Join). Diese Methoden werden in *Systemverwaltung: Optimierung*. Dieser Abschnitt enthält detaillierte Informationen zu den einzelnen Elementen von Zugriffsanforderungen und Joinanforderungen, die zur Beeinflussung der Planoptimierung verwendet werden können.

Zugriffsanforderungen: Die Gruppe *accessRequest* definiert den Satz gültiger Auswahlmöglichkeiten für Zugriffsanforderungselemente. Eine Zugriffsanforderung gibt eine gewünschte Methode zur Erfüllung eines Tabellenverweises in einer Anweisung an.

XML-Schema

```
    <xs:group name="accessRequest">
          <xs:choice>
                <xs:element name="TBSCAN" type="tableScanType"/>
                <xs:element name="IXSCAN" type="indexScanType"/>
                <xs:element name="LPREFETCH" type="listPrefetchType"/>
                <xs:element name="IXAND" type="indexAndingType"/>
```

```
                <xs:element name="IXOR" type="indexOringType"/>
                <xs:element name="ACCESS" type="anyAccessType"/>
    </xs:choice>
  </xs:group>
```

Beschreibung

- TBSCAN, IXSCAN, LPREFETCH, IXAND und IXOR

  Diese Elemente entsprechen den DB2-Datenzugriffsmethoden und können nur auf lokale Tabellen angewendet werden, die in der Anweisung angegeben werden. Sie können sich nicht auf Kurznamen (ferne Tabellen) oder abgeleitete Tabellen (Ergebnis eines Subselects) beziehen.
- ACCESS

  Dieses Element wird in Fällen verwendet, in denen es in erster Linie auf die Joinreihenfolge, und nicht so sehr auf die Zugriffsmethode ankommt. Dieses Element muss verwendet werden, wenn der Zieltabellenverweis eine abgeleitete Tabelle ist. Das Optimierungsprogramm wählt die Zugriffsmethode für den Zieltabellenverweis auf der Grundlage von Aufwandsberechnungen aus.

Zugriffstypen: Allgemeine Aspekte der Elemente TBSCAN, IXSCAN, LPREFETCH, IXAND, IXOR und ACCESS werden durch den nachfolgend gezeigten abstrakten Typ 'accessType' definiert.

XML-Schema

```
    <xs:complexType name="accessType" abstract="true">
          <xs:attribute name="TABLE"  type="xs:string"  use="optional"/>
          <xs:attribute name="TABID" type="xs:string" use="optional"/>
          <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
  </xs:complexType>
```

Beschreibung

Alle Zugriffsanforderungselemente erweitern den komplexen Typ *accessType*. Jedes Element dieser Art muss den Zieltabellenverweis entweder durch das Attribut *TABLE* oder das Attribut *TABID* angeben. Im Abschnitt „Bildung von Tabellenverweisen in Optimierungsrichtlinien" auf Seite 429 wird beschrieben, wie korrekte Tabellenverweise aus einem Zugriffsanforderungselement gebildet werden. Diese Elemente können außerdem das optionale Attribut *FIRST* angeben. Wenn das Attribut FIRST angegeben wird, muss es den Wert TRUE haben. Durch Hinzufügen des Attributs FIRST in einem Zugriffsanforderungselement wird angegeben, dass ein Ausführungsplan gewünscht wird, der angibt, dass die von dieser Zugriffsanforderung angegebene Tabelle als erste Tabelle in der Joinreihenfolge für die entsprechende FROM-Klausel verwendet wird. Nur in einer Zugriffs- oder Joinanforderung pro FROM-Klausel kann das Attribut FIRST angegeben werden. Wenn mehrere Zugriffs- oder Joinanforderungen, die Tabellen derselben FROM-Klausel angeben, das Attribut FIRST enthalten, werden alle außer der ersten Anforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Anforderungen für beliebigen Zugriffstyp: Das Zugriffsanforderungselement *ACCESS* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm die geeignete Methode für den Zugriff auf eine Tabelle auf der Grundlage einer Aufwandsberechnung auswählen soll. Diese Auswahlmöglichkeit wird in der Regel zur Angabe einer Zugriffsanforderung verwendet, die nur angibt, wie die lokale Tabelle mit anderen Tabellen in der Anweisung verknüpft wird. Dieses Zugriffsanforderungselement muss verwendet werden, wenn auf eine abgeleitete Tabelle verwiesen wird. Eine abgeleitete Tabelle ist das Ergebnis einer anderen Subselect-Anweisung. Ein Zugriffsanforderungselement des Typs 'ACCESS' ist durch den komplexen Typ *anyAccessType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="anyAccessType">
      <xs:complexContent>
         <xs:extension base="accessType"/>
      </xs:complexContent>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *anyAccessType* ist eine einfache Erweiterung des abstrakten Typs *accessType*. Es werden keine neuen Elemente oder Attribute hinzugefügt.

Zugriffsanforderungen mit logischem Verknüpfen von Indizes über AND (Index ANDing): Das Zugriffsanforderungselement *indexAnding* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm die Datenzugriffsmethode mit logischer Verknüpfung von Indizes über AND (Index ANDing) für den Zugriff auf eine lokale Tabelle verwendet. Es ist durch den komplexen Typ *indexAndingType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="indexAndingType">
      <xs:complexContent>
         <xs:extension base="accessType">
            <xs:sequence minOccurs="0">
               <xs:element name="INDEX" type="indexType" minOccurs="2" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="INDEX" type="xs:string" use="optional"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="indexType">
      <xs:attribute name="IXNAME" type="xs:string" use="optional"/>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *indexAndingType* erweitert den abstrakten Typ *localAccessType*, indem er das optionale Attribut *INDEX* und optionale Unterelemente INDEX hinzufügt. Mit dem Attribut INDEX kann der Index angegeben werden, der als erster Index in einer Operation zum logischen Verknüpfen von Indizes über AND (Index ANDing) verwendet werden soll. Wenn das Attribut INDEX verwendet wird, wählt das Optimierungsprogramm die zusätzlichen Indizes und die Zugriffsreihenfolge auf der Grundlage einer Aufwandsberechnung aus. Mit den INDEX-Elementen können der exakte Satz von Indizes und die Zugriffsreihenfolge angegeben werden. Die Reihenfolge, in der die INDEX-Unterelemente angegeben sind, bestimmt die Reihenfolge, in der die einzelnen Indexsuchen ausgeführt werden sollen. Die Angabe von INDEX-Unterelementen überschreibt die Angabe des Attributs INDEX.

- Wenn keine Indizes angegeben werden, wählt das Optimierungsprogramm sowohl die Indizes als auch die Zugriffsreihenfolge auf der Grundlage einer Aufwandsberechnung aus.
- Wenn Indizes entweder durch das Attribut oder durch Unterelemente angegeben werden, muss es sich um Indizes handeln, die für die durch das Attribut TABLE oder TABID angegebene Tabelle definiert sind.
- Wenn für die Tabelle keine Indizes definiert sind, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Blockindizes müssen vor Satzindizes in einer Zugriffsanforderung mit logischer AND-Verknüpfung von Indizes angegeben werden. Wenn diese Voraussetzung

nicht erfüllt ist, wird eine Warnung SQL0437W mit Ursachencode 13 ausgegeben. Die Zugriffsmethode mit logischer AND-Verknüpfung von Indizes erfordert mindestens ein Vergleichselement, das indexiert werden kann. Wenn der Zugriff mit logischer AND-Verknüpfung von Indizes nicht auswählbar ist, weil das erforderliche Vergleichselement nicht vorhanden ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben. Wenn die Zugriffsmethode mit logischer AND-Verknüpfung von Indizes nicht im Suchbereich enthalten ist, der für die Anweisung wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Zugriffsanforderungen mit logischem Verknüpfen von Indizes über OR (Index ORing): Das Zugriffsanforderungselement *IXOR* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm die Datenzugriffsmethode mit logischer Verknüpfung von Indizes über OR (Index ORing) für den Zugriff auf eine lokale Tabelle verwendet. Es ist durch den komplexen Typ *indexOringType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="indexOringType">
      <xs:complexContent>
         <xs:extension base="accessType"/>
      </xs:complexContent>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *indexOringType* ist eine einfache Erweiterung des abstrakten Typs *accessType*. Es werden keine neuen Elemente oder Attribute hinzugefügt. Wenn die Zugriffsmethode mit logischer OR-Verknüpfung von Indizes nicht im Suchbereich enthalten ist, der für die Anweisung wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben. Das Optimierungsprogramm wählt die Vergleichselemente und Indizes, die in der OR-Verknüpfungsoperationen von Indizes verwendet werden, auf der Grundlage einer Aufwandsberechnung aus. Die Zugriffsmethode mit logischer OR-Verknüpfung von Indizes erfordert mindestens ein IN-Vergleichselement, das indexiert werden kann, oder ein Vergleichselement mit Termen, die indexiert und durch eine logische OR-Operation verbunden werden können. Wenn der Zugriff mit logischer OR-Verknüpfung von Indizes nicht auswählbar ist, weil das erforderliche Vergleichselement bzw. die erforderlichen Indizes nicht vorhanden sind, wird die Anforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Zugriffsanforderungen für Indexsuchen: Das Zugriffsanforderungselement *IXSCAN* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm eine Indexsuche für den Zugriff auf eine lokale Tabelle verwendet. Es ist durch den komplexen Typ *indexScanType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="indexScanType">
      <xs:complexContent>
         <xs:extension base="accessType">
            <xs:attribute name="INDEX" type="xs:string" use="optional"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *indexScanType* erweitert den abstrakten Typ 'accessType', indem er das optionale Attribut *INDEX* hinzufügt. Das Attribut INDEX gibt den Namen des Index an, der für den Zugriff auf die Tabelle zu verwenden ist.

- Wenn die Zugriffsmethode der Indexsuche nicht im Suchbereich enthalten ist, der für die Anweisung wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.
- Wenn das Attribut INDEX angegeben wird, muss es einen Index angeben, der für die durch das Attribut TABLE oder TABID angegebene Tabelle definiert ist.
- Wenn der Index nicht vorhanden ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.
- Wenn das Attribut INDEX nicht angegeben wird, wählt das Optimierungsprogramm einen Index auf der Grundlage einer Aufwandsberechnung aus.
- Wenn für die Zieltabelle keine Indizes definiert sind, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Anforderungen für Vorablesezugriff über Listen: Das Element *listPrefetch* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm eine Indexsuche mit Vorablesezugriff über Listen für den Zugriff auf eine lokale Tabelle verwendet. Es ist durch den komplexen Typ *listPrefetchType* wie folgt definiert:

```
XML-Schema

    <xs:complexType name="listPrefetchType">
    <xs:complexContent>
       <xs:extension base="accessType">
          <xs:attribute name="INDEX" type="xs:string" use="optional"/>
       </xs:extension>
    </xs:complexContent>
 </xs:complexType>
```

Beschreibung

Der komplexe Typ *listPrefetchType* erweitert den abstrakten Typ *accessType*, indem er das optionale Attribut *INDEX* hinzufügt. Das Attribut INDEX gibt den Namen des Index an, der für den Zugriff auf die Tabelle zu verwenden ist.

- Wenn die Zugriffsmethode mit Vorablesezugriff über Listen nicht im Suchbereich enthalten ist, der für die Anweisung wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.
- Die Zugriffsmethode mit Vorablesezugriff über Listen erfordert mindestens ein Vergleichselement, das indexiert werden kann.
- Wenn der Vorablesezugriff über Listen nicht auswählbar ist, weil das erforderliche Vergleichselement nicht vorhanden ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.
- Wenn das Attribut INDEX angegeben wird, muss es einen Index angeben, der für die durch das Attribut TABLE oder TABID angegebene Tabelle definiert ist.
- Wenn der Index nicht vorhanden ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.
- Wenn das Attribut INDEX nicht angegeben wird, wählt das Optimierungsprogramm einen Index auf der Grundlage einer Aufwandsberechnung aus.
- Wenn für die Zieltabelle keine geeigneten Indizes definiert sind, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Zugriffsanforderungen für Tabellensuchen: Das Zugriffsanforderungselement *TBSCAN* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm eine sequenzielle Tabellensuche für den Zugriff auf eine lokale Tabelle verwendet. Es ist durch den komplexen Typ *tableScanType* wie folgt definiert:

XML-Schema

```
<xs:complexType name="tableScanType">
<xs:complexContent>
   <xs:extension base="accessType"/>
</xs:complexContent>
</xs:complexType>
```

Beschreibung

Der komplexe Typ *tableScanType* ist eine einfache Erweiterung des abstrakten Typs *accessType*. Es werden keine neuen Elemente oder Attribute hinzugefügt. Wenn die Zugriffsmethode der Tabellensuche nicht im Suchbereich enthalten ist, der für die Anweisung wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Joinanforderungen: Die Gruppe *joinRequest* definiert den Satz gültiger Auswahlmöglichkeiten für Joinanforderungselemente. Eine Joinanforderung gibt eine gewünschte Methode zum Verknüpfen zweier Tabellen an.

XML-Schema

```
<xs:group name="joinRequest">
      <xs:choice>
            <xs:element name="NLJOIN" type="nestedLoopJoinType"/>
            <xs:element name="HSJOIN" type="hashJoinType"/>
            <xs:element name="MSJOIN" type="mergeJoinType"/>
            <xs:element name="JOIN" type="anyJoinType"/>
</xs:choice>
</xs:group>
```

Beschreibung

Die Joinanforderungselemente *NLJOIN*, *MSJOIN* und *HSJOIN* entsprechen den Methoden Join mit Verschachtelungsschleife (Nested Loop Join), Mischjoin (Merge Join) und Hash-Join. In *Systemverwaltung: Optimierung* werden die Eignungskriterien und Ausführungsmerkmale dieser Joinmethoden detaillierter beschrieben. Das Joinanforderungselement *JOIN* gibt an, dass das Optimierungsprogramm die Joinmethode frei auswählen kann. Diese Auswahlmöglichkeit ist in Fällen nützlich, in denen in erster Linie die Angabe einer bestimmten Joinreihenfolge von Bedeutung ist.

Alle Joinanforderungselemente enthalten zwei Unterelemente, die die Eingabetabellen der Joinoperation darstellen. Joinanforderungen können außerdem das optionale Attribut FIRST angeben.

Beliebige Joinanforderungen: Das Joinanforderungselement *JOIN* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm zwei Tabellen in einer bestimmten Reihenfolge, jedoch mit einer beliebigen, vom Optimierungsprogramm ausgewählten Joinmethode verknüpft. Bei jeder Eingabetabelle kann es sich um eine lokale oder abgeleitete Tabelle handeln, wie dies durch ein Unterelement einer Zugriffsanforderung angegeben wird, oder sie können das Ergebnis einer Joinoperation sein, wie dies durch ein Unterelement einer Joinanforderung angegeben wird. Das Joinanforderungselement *JOIN* ist durch den komplexen Typ *anyJoinType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="anyJoinType">
      <xs:complexContent>
         <xs:extension base="joinType"/>
      </xs:complexContent>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *anyJoinType* ist eine einfache Erweiterung des abstrakten Typs *joinType*. Es werden keine neuen Elemente oder Attribute hinzugefügt.

HSJOIN-Anforderungen: Das Joinanforderungselement *HSJOIN* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm zwei Tabellen durch die Hash-Join-Methode verknüpft. Bei jeder Eingabetabelle kann es sich um eine lokale oder abgeleitete Tabelle handeln, wie dies durch ein Unterelement einer Zugriffsanforderung angegeben wird, oder sie können das Ergebnis einer Joinoperation sein, wie dies durch ein Unterelement einer Joinanforderung angegeben wird. Das Joinanforderungselement *HSJOIN* ist durch den komplexen Typ *hashJoinType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="hashJoinType">
      <xs:complexContent>
         <xs:extension base="joinType"/>
      </xs:complexContent>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *hashJoinType* ist eine einfache Erweiterung des abstrakten Typs *joinType*. Es werden keine neuen Elemente oder Attribute hinzugefügt. Wenn die Hash-Join-Methode nicht im Suchbereich enthalten ist, der für die Anweisung wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

MSJOIN-Anforderungen: Das Joinanforderungselement *MSJOIN* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm zwei Tabellen durch die Mischjoinmethode (Merge Join) verknüpft. Bei jeder Eingabetabelle kann es sich um eine lokale oder abgeleitete Tabelle handeln, wie dies durch ein Unterelement einer Zugriffsanforderung angegeben wird, oder sie können das Ergebnis einer Joinoperation sein, wie dies durch ein Unterelement einer Joinanforderung angegeben wird. Das Joinanforderungselement *MSJOIN* ist durch den komplexen Typ *mergeJoinType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="mergeJoinType">
      <xs:complexContent>
         <xs:extension base="joinType"/>
      </xs:complexContent>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *mergeJoinType* ist eine einfache Erweiterung des abstrakten Typs *joinType*. Es werden keine neuen Elemente oder Attribute hinzugefügt. Wenn die Mischjoinmethode nicht im Suchbereich enthalten ist, der für die Anweisung

wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

NLJOIN-Anforderungen: Das Joinanforderungselement *NLJOIN* kann verwendet werden, um anzufordern, dass das Optimierungsprogramm zwei Tabellen durch die Joinmethode mit Verschachtelungsschleife (Nested Loop Join) verknüpft. Bei jeder Eingabetabelle kann es sich um eine lokale oder abgeleitete Tabelle handeln, wie dies durch ein Unterelement einer Zugriffsanforderung angegeben wird, oder sie können das Ergebnis einer Joinoperation sein, wie dies durch ein Unterelement einer Joinanforderung angegeben wird. Das Joinanforderungselement *NLJOIN* ist durch den komplexen Typ *nestedLoopJoinType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="nestedLoopJoinType">
      <xs:complexContent>
         <xs:extension base="joinType"/>
      </xs:complexContent>
   </xs:complexType>
```

Beschreibung

Der komplexe Typ *nestedLoopJoinType* ist eine einfache Erweiterung des abstrakten Typs *joinType*. Es werden keine neuen Elemente oder Attribute hinzugefügt. Wenn die Joinmethode mit Verschachtelungsschleife nicht im Suchbereich enthalten ist, der für die Anweisung wirksam ist, wird die Zugriffsanforderung ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Typen von Joinanforderungen: Allgemeine Aspekte aller Joinanforderungselemente werden durch den abstrakten Typ *joinType* wie folgt definiert:

```
XML-Schema

      <xs:complexType name="joinType" abstract="true">
            <xs:choice minOccurs="2" maxOccurs="2">
         <xs:group ref="accessRequest"/>
         <xs:group ref="joinRequest"/>
      </xs:choice>
      <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
   </xs:complexType>
```

Beschreibung

Joinanforderungselemente, die den Typ *joinType* erweitern, müssen genau zwei Unterelemente besitzen. Jedes der beiden Unterelemente kann ein aus der Gruppe *accessRequest* ausgewähltes Zugriffsanforderungselement oder ein weiteres, aus der Gruppe *joinRequest* ausgewähltes Joinanforderungselement sein. Das erste Unterelement, dass in der Joinanforderung auftritt, gibt die äußere Tabelle der Joinoperation an. Das zweite Element gibt die innere Tabelle an. Wenn das Attribut FIRST angegeben wird, muss es den Wert TRUE haben. Durch Hinzufügen des Attributs FIRST in einem Joinanforderungselement wird angeben, dass ein Ausführungsplan gewünscht wird, bei dem die durch die Joinanforderung angegebenen Tabellen die äußersten Tabellen in der Joinreihenfolge für die entsprechende FROM-Klausel sind. Nur in einer Zugriffs- oder Joinanforderung pro FROM-Klausel kann das Attribut FIRST angegeben werden. Wenn mehrere Zugriffs- oder Joinanforderungen, die Tabellen derselben FROM-Klausel angeben, das Attribut FIRST enthalten, werden alle außer der ersten Anforderung dieser Art ignoriert und eine Warnung SQL0437W mit Ursachencode 13 ausgegeben.

Tabelle SYSTOOLS.OPT_PROFILE

Die Tabelle SYSTOOLS.OPT_PROFILE enthält alle Optimierungsprofildateien. Diese Tabelle kann mit zwei Methoden erstellt werden:

- Rufen Sie die Prozedur SYSINSTALLOBJECTS auf:

```
db2 "call sysinstallobjects('opt_profiles', 'c', '', '')"
```

- Führen Sie den Befehl CREATE TABLE aus:

```
CREATE TABLE SYSTOOLS.OPT_PROFILE (
            SCHEMA  VARCHAR(128) NOT NULL,
            NAME    VARCHAR(128) NOT NULL,
            PROFILE BLOB (2M)    NOT NULL,
            PRIMARY KEY ( SCHEMA, NAME )
          );
```

Die Spalten in der Tabelle SYSTOOLS.OPT_PROFILE sind wie folgt definiert:

SCHEMA

- Gibt das Schemaqualifikationsmerkmal des Optimierungsprofils an. Der Schemaname kann bis zu 30 alphanumerische Zeichen und Unterstreichungszeichen enthalten, muss jedoch wie gezeigt als Typ VARCHAR(128) definiert werden.

NAME

- Gibt den Basisnamen des Optimierungsprofils an. Der Name kann bis zu 128 alphanumerische Zeichen oder Unterstreichungszeichen enthalten.

PROFILE

- Das XML-Dokument, in dem das Optimierungsprofil definiert ist.

Anmerkung:

1. Auf ein Optimierungsprofil kann in mehreren verschiedenen Kontexten verwiesen werden. Die Spalten NAME und SCHEMA der Tabelle SYSTOOLS.OPT-_PROFILE geben zusammen den zweiteiligen Namen des Optimierungsprofils an. Die Spalte PROFILE enthält das XML-Dokument, in dem das Optimierungsprofil definiert ist.
2. Im Hinblick auf Tabellenbereiche oder Partitionsgruppen sind keine Einschränkungen zu beachten.

Trigger zum Löschen des Cache für das Optimierungsprofil: Die folgende SQL-Prozedur und die folgenden Trigger sollten erstellt werden, um sicherzustellen, dass der Inhalt des Profilcache automatisch gelöscht (FLUSH) wird, wenn ein Eintrag in der Tabelle SYSTOOLS.OPT_PROFILE aktualisiert oder gelöscht wird:

```
CREATE PROCEDURE SYSTOOLS.OPT_FLUSH_CACHE( IN SCHEMA VARCHAR(128),
                                           IN NAME VARCHAR(128) )
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN ATOMIC
 -- FLUSH stmt (33) + angegebenes Schema (130) + Punkt (1) + angegebener Name (130) = 294
 DECLARE FSTMT VARCHAR(294) DEFAULT 'FLUSH OPTIMIZATION PROFILE CACHE '; --

 IF NAME IS NOT NULL THEN
   IF SCHEMA IS NOT NULL THEN
     SET FSTMT = FSTMT || '"' || SCHEMA || '".'; --
   END IF; --

   SET FSTMT = FSTMT || '"' || NAME || '"'; --

   EXECUTE IMMEDIATE FSTMT; --
 END IF; --
```

```
END;

CREATE TRIGGER SYSTOOLS.OPT_PROFILE_UTRIG AFTER UPDATE ON SYSTOOLS.OPT_PROFILE
REFERENCING OLD AS O
FOR EACH ROW
  CALL SYSTOOLS.OPT_FLUSH_CACHE( O.SCHEMA, O.NAME );

CREATE TRIGGER SYSTOOLS.OPT_PROFILE_DTRIG AFTER DELETE ON SYSTOOLS.OPT_PROFILE
REFERENCING OLD AS O
FOR EACH ROW
  CALL SYSTOOLS.OPT_FLUSH_CACHE( O.SCHEMA, O.NAME );
```

Verwalten der Tabelle SYSTOOLS.OPT_PROFILE: Optimierungsprofildateien müssen einen eindeutigen, durch ein Schema qualifizierten Namen haben und in der Tabelle SYSTOOLS.OPT_PROFILE gespeichert werden. Zur Verwaltung der Dateien in dieser Tabelle können die Befehle LOAD, IMPORT und EXPORT verwendet werden. Zum Beispiel können mithilfe des Befehls IMPORT von einem beliebigen DB2-Client aus Daten aus den Dateien in die Tabelle SYSTOOLS.OPT-_PROFILE eingefügt bzw. aktualisiert werden. Der Befehl EXPORT kann zum Abrufen eines Profils aus der Tabelle OPT_PROFILE in eine Datei verwendet werden.

Das folgende Beispiel zeigt, wie drei neue Zeilen aus separaten Eingabedateien in die Tabelle SYSTOOLS.OPT_PROFILE eingefügt werden. Dabei wird angenommen, dass sich die Dateien im aktuellen Verzeichnis befinden.

1. Erstellen Sie eine Eingabedatei (z. B. profiledata) mit dem Schema, dem Namen und dem Dateinamen jeder Datenzeile in einer separaten Zeile in der Datei:

   ```
   "ROBERT","PROF1","ROBERT.PROF1.xml"
   "ROBERT","PROF2","ROBERT.PROF2.xml"
   "DAVID", "PROF1","DAVID.PROF1.xml"
   ```

2. Führen Sie den Befehl IMPORT aus:

   ```
   IMPORT FROM profiledata OF DEL MODIFIED BY LOBSINFILE INSERT INTO SYSTOOLS.OPT_PROFILE
   ```

Zur Aktualisierung vorhandener Zeilen können Sie diese entweder zunächst löschen und erneut, wie oben gezeigt, einfügen, oder Sie verwenden die Option INSERT_UPDATE im Befehl IMPORT:

```
IMPORT FROM profiledata OF DEL MODIFIED BY LOBSINFILE
        INSERT_UPDATE INTO SYSTOOLS.OPT_PROFILE
```

Zum Abrufen des Profils für ROBERT.PROF1 in die Datei ROBERT.PROF1.xml unter der Annahme, dass das Profil kleiner als 32.700 Byte ist, führen Sie zum Beispiel folgenden Befehl aus:

```
EXPORT TO ROBERT.PROF1.xml OF DEL SELECT PROFILE FROM SYSTOOLS.OPT_PROFILE
  WHERE SCHEMA='ROBERT' AND NAME='PROF1'
```

Informationen zum Exportieren von Daten, die 32.700 Byte überschreiten, sowie weitere Informationen finden Sie in der Dokumentation zum Befehl EXPORT in *Command Reference*.

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.

- Zwischenspeicher für Sortierlisten (sortheap)

  Wenn die zu sortierenden Zeilen mehr als den im Zwischenspeicher für Sortierlisten 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.

Auswirkung von Datenbankpartitionsgruppen auf die Abfrageoptimierung

In Umgebungen mit partitionierten Datenbanken erkennt das Optimierungsprogramm die Kollokation von Tabellen und nutzt sie bei der Bestimmung des besten Zugriffsplans für eine Abfrage. Wenn Tabellen häufig in Joinabfragen einbezogen werden, sollten sie auf Datenbankpartitionen in einer Umgebung mit partitionierten Datenbanken aufgeteilt werden, sodass sich die Zeilen aus jeder Tabelle, die verknüpft wird, in der gleichen Datenbankpartition befinden. Während der Joinoperation wird durch die Kollokation der Daten aus beiden verknüpften Tabellen eine Verschiebung der Daten von einer Datenbankpartition in die andere vermieden. Speichern Sie beide Tabellen in derselben Datenbankpartitionsgruppe, um sicherzustellen, dass die Daten aus den Tabellen durch Kollokation zusammengefasst werden.

In einer Umgebung mit partitionierten Datenbanken wird durch das Verteilen von Daten auf mehrere Datenbankpartitionen, abhängig von der Größe der Tabelle, die geschätzte Dauer (bzw. der Aufwand) zur Ausführung einer Abfrage verringert. Die Anzahl der Tabellen, die Größe der Tabellen, die Speicherposition der Daten in diesen Tabellen und die Art der Abfrage (d. h., ob ein Join erforderlich ist) wirken sich alle auf den Aufwand für die Abfrage aus.

Spaltenkorrelation für mehrere Vergleichselemente

Ihre Anwendungen enthalten möglicherweise Abfragen, die mit Joins so konstruiert sind, dass zwei Tabellen über mehr als ein Joinvergleichselement verknüpft werden. Dies ist nicht ungewöhnlich, wenn Abfragen die Beziehungen zwischen ähnlichen zusammengehörigen Spalten in verschiedenen Tabellen bestimmen müssen.

Betrachten Sie zum Beispiel einen Hersteller, der Produkte aus Rohmaterialien verschiedener Farben, Elastizitäten und Qualitäten produziert. Das fertige Produkt hat dieselbe Farbe und Elastizität wie das Rohmaterial, aus dem es hergestellt ist. Der Hersteller führt die folgende Abfrage aus:

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY
FROM PRODUCT, RAWMATERIAL
   WHERE PRODUCT.COLOR       =  RAWMATERIAL.COLOR
     AND PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Diese Abfrage liefert die Namen und die Qualität der Rohmaterialien aller Produkte. Zwei Vergleichselemente werden für den Join verwendet:

```
PRODUCT.COLOR       =  RAWMATERIAL.COLOR
PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Wenn das Optimierungsprogramm einen Plan zur Ausführung dieser Abfrage auswählt, berechnet es die Selektivität jedes der beiden Vergleichselemente. Dabei geht es davon aus, dass sie unabhängig sind, das heißt, dass alle Variationen von Elastizität für jede Farbe vorkommen und umgekehrt für jede Elastizitätsstufe Rohmaterial in jeder Farbe verfügbar ist. Dann schätzt es die Gesamtselektivität des Vergleichselementpaares ab, indem es auf Katalogstatistikdaten für jede Tabelle auf der Basis der Anzahl der Elastizitätsstufen und der Anzahl verschiedener Farben zurückgreift. Ausgehend von diesem Schätzwert kann es beispielsweise einen Join mit Verschachtelungsschleife (Nested Loop Join) einem Mischjoin (Merge Join) vorziehen oder umgekehrt.

Es kann jedoch vorkommen, dass diese beiden Vergleichselemente nicht unabhängig sind. Zum Beispiel könnte es der Fall sein, dass die hochelastischen Materialien nur in wenigen Farben verfügbar sind und die sehr wenig elastischen Materialien nur in einigen anderen Farben verfügbar sind, die sich von den Farben der elastischen Materialien unterscheiden. In diesem Fall eliminiert die kombinierte Selektivität der Vergleichselemente weniger Zeilen, sodass die Abfrage mehr Zeilen liefert. Betrachten Sie den extremen Fall, dass es nur eine Stufe von Elastizität für jede Farbe gibt und umgekehrt. Hier könnte jedes der beiden Vergleichselemente logisch ganz weggelassen werden, da es vom jeweils anderen impliziert wird. Das Optimierungsprogramm wählt vielleicht nicht mehr den besten Plan. Zum Beispiel könnte es einen Plan mit Joins mit Verschachtelungsschleife wählen, obwohl ein Mischjoin schneller wäre.

Das DB2-Optimierungsprogramm versucht, eine Korrelation von Joinvergleichselementen zu ermitteln und zu kompensieren, wenn Sie einen Index für diese Spalten definieren oder wenn Sie Gruppenspaltenstatistiken für die entsprechenden Spalten erfassen und pflegen.

In dem obigen Beispiel mit Farben und Elastizitäten von Materialien könnten Sie einen oder beide der folgenden Indizes definieren:

```
IX1 PRODUCT.COLOR, PRODUCT.ELASTICITY
```

oder

```
IX2 RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

oder beide.

Damit das Optimierungsprogramm die Korrelation erkennen kann, dürfen die Nicht-INCLUDE-Spalten dieses Index nur die korrelierten Spalten sein. Der Index kann darüber hinaus auch INCLUDE-Spalten enthalten, um Abfragen nur über

Indexsuchen erfüllen zu können. Wenn es mehr als zwei korrelierte Spalten in Joinvergleichselementen gibt, stellen Sie sicher, dass Sie den Index so definieren, dass er alle diese Spalten abdeckt.

Eine Bedingung, die erfüllt sein muss, damit das Optimierungsprogramm die Kardinalität eines Indexschlüssels in Betracht zieht, um eine Korrelation zu erkennen, besteht darin, dass die Anzahl der unterschiedlichen Werte in jeder Spalte für jede Spalte aus der zu vergleichenden Tabelle höher sein muss. Nehmen Sie zum Beispiel an, Sie haben die Indizes IX1 und IX2 wie oben gezeigt definiert. Wenn die Anzahl der unterschiedlichen Werte der Spalte PRODUCT.COLOR kleiner ist als die Anzahl der unterschiedlichen Werte der Spalte RAWMATERIAL.COLOR und die Anzahl der unterschiedlichen Werte der Spalte PRODUCT.ELASTICITY kleiner ist als die der Spalte RAWMATERIAL.ELASTICITY, dann verwendet das Optimierungsprogramm den Index IX2 zur Erkennung der Korrelation. Ein Vergleich der Spaltenkardinalitäten legt nahe, dass die unterschiedlichen Werte in den PRODUCT-Spalten sehr wahrscheinlich, wenn auch nicht mit absoluter Sicherheit, in den unterschiedlichen Werten der RAWMATERIAL-Spalten enthalten sind. Um die Wahrscheinlichkeit, dass ein Wertebereich einen anderen voraussetzt, noch weiter zu erhärten, kann das Optimierungsprogramm außerdem die HIGH2KEY- und LOW2KEY-Statistikdaten für diese Indexspalten zu Rate ziehen.

Stellen Sie nach der Erstellung der geeigneten Indizes sicher, dass die Statistikdaten für Tabellen präzise und aktuell sind.

Das Optimierungsprogramm verwendet die Informationen in den Spalten FIRSTnKEYCARD und FULLKEYCARD der Statistiktabellen für eindeutige Indizes, um Fälle von Korrelation zu erkennen und die errechneten kombinierten Selektivitäten der korrelierten Vergleichselemente dynamisch anzupassen, um so eine realistischere Schätzung der Größe und des Aufwands des Joins zu erhalten.

Alternativ können Spaltengruppenstatistiken für eine Gruppe von Spalten gesammelt werden. Im obigen Elastizitätsbeispiel könnten Sie Statistiken für die Spalten PRODUCT.COLOR, PRODUCT.ELASTICITY und/oder RAWMATERIAL.COLOR, RAWMATERIAL.ELASTCITY sammeln.

Spaltengruppenstatistiken werden mit der Option ON COLUMNS des Dienstprogramms RUNSTATS erfasst. Wenn Sie zum Beispiel die Spaltengruppenstatistiken für PRODUCT.COLOR und PRODUCT.ELASTICITY sammeln wollen, führen Sie den folgenden RUNSTATS-Befehl aus:

```
RUNSTATS ON TABLE product ON COLUMNS ((color, elasticity))
```

Korrelation einfacher Gleichheitsvergleichselemente

Zusätzlich zur Korrelation von Joinvergleichselementen verwaltet das Optimierungsprogramm auch die Korrelation mit einfachen Gleichheitsvergleichselementen des Typs SPALTE =. Betrachten Sie zum Beispiel eine Tabelle mit verschiedenen Typen von Fahrzeugen vor, die jeweils mit MARKE (d. h. Hersteller), MODELL, JAHR, FARBE und AUSFÜHRUNG (z. B. Limousine, Kombi, Sport- oder Nutzfahrzeug) eingetragen sind. Da die weitaus meisten Hersteller die gleichen Standardfarben für jedes ihrer Modelle und jede ihrer Ausführungen Jahr für Jahr liefern, sind die Vergleichselemente für die Spalte FARBE wahrscheinlich von denen für die Spalten MARKE, MODELL, AUSFÜHRUNG oder JAHR unabhängig. Die Vergleichselemente MARKE und MODELL sind jedoch in keinem Fall unabhängig voneinander, da nur jeweils ein Fahrzeughersteller ein Modell mit einem bestimmten Namen produziert. Identische Modellnamen, die von zwei oder mehr

Fahrzeugherstellern verwendet werden, sind sehr unwahrscheinlich und mit Sicherheit von den Fahrzeugherstellern nicht gewollt.

Wenn für die beiden Spalten MARKE und MODELL ein Index vorhanden ist oder Spaltengruppenstatistiken erfasst sind, verwendet das Optimierungsprogramm die Statistikdaten über diesen Index oder die Spalten, um die kombinierte Zahl an unterschiedlichen Werten festzustellen und die Selektivitäts- oder Kardinalitätsschätzung für die Korrelation zwischen diesen beiden Spalten anzupassen. Wenn die Vergleichselemente lokale Gleichheitsvergleichselemente sind, benötigt das Optimierungsprogramm keinen eindeutigen Index, um die Anpassung vorzunehmen.

Berechnen von Gruppierungsschlüsselkardinalitäten mit Index- und Spaltengruppenstatistikdaten

Wenn eine Abfrage eine bestimmte Gruppierung von Daten erfordert, muss das Optimierungsprogramm die Anzahl der unterschiedlichen Gruppierungen, d. h. die *Gruppierungsschlüsselkardinalitäten* ('grouping keycard') berechnen. Eine Gruppierungsanforderung kann aus solchen Operationen wie GROUP BY oder DISTINCT resultieren.

Betrachten Sie die folgende Abfrage:

```
SELECT DEPTNO, YEARS, AVG(SALARY)
  FROM EMPLOYEE
  GROUP BY DEPTNO, MGR, YEAR_HIRED
```

Ohne Index- oder Spaltengruppenstatistiken entspricht die vom Optimierungsprogramm geschätzte Anzahl von Gruppierungen (in diesem Fall auch die Anzahl der zurückgegebenen Zeilen) dem Produkt aus den Anzahlen unterschiedlicher Werte der Spalten DEPTNO, MGR und YEAR_HIRED. Dieser Schätzwert basiert auf der Annahme, dass die Gruppierungsschlüsselspalten unabhängig sind. Diese Annahme kann jedoch abwegig sein, wenn jeder Manager genau eine Abteilung leitet. Darüber hinaus ist es unwahrscheinlich, dass jede Abteilung jedes Jahr Mitarbeiter angestellt hat. Daher könnte das Produkt der unterschiedlichen Werte der Spalten DEPTNO, MGR und YEAR_HIRED als Schätzwert für die tatsächliche Anzahl der Gruppen mit unterschiedlichen Werten zu hoch sein.

Betrachten Sie nun einen Index mit der folgenden Definition:

```
INDEX IX1:  DEPTNO, MGR, YEAR_HIRED
```

In diesem Fall liefert der FULLKEYCARD-Wert von IX1 dem Optimierungsprogramm die genaue Anzahl der unterschiedlichen Gruppierungen für die obige Abfrage.

Betrachten Sie eine weitere Indexdefinition:

```
INDEX IX2:  DEPTNO, MGR, YEAR_HIRED, COMM
```

Der Index IX2 könnte ebenfalls helfen, die Gruppierungsschlüsselkardinalität zu berechnen, da der Wert FIRST3KEYCARD angibt, wie viele unterschiedliche Gruppen sich aus (DEPTNO,MGR,YEAR_HIRED) ergeben.

Neben den Indexstatistikdaten (FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD und FULLKEYCARD) könnten auch Spaltengruppenstatistiken in ähnlicher Weise vom Optimierungsprogramm zur Berechnung der Gruppierungsschlüsselkardinalität genutzt werden. Spaltengruppenstatistikdaten, die für

die Spalten DEPTNO, MGR und YEAR_HIRED erfasst werden, bieten den gleichen Nutzen wie die oben gezeigten Indizes IX1 und IX2:

```
RUNSTATS ON TABLE EMPLOYEE ON COLUMNS ((DEPTNO, MGR, YEAR_HIRED))
```

Beachten Sie, dass, wenn die Gruppierungsschlüssel aus fünf und mehr Spalten bestehen, eine Erfassung von Spaltengruppenstatistiken möglicherweise vorzuziehen ist. Dies liegt daran, dass das Dienstprogramm RUNSTATS nur Statistiken für die ersten vier Spalten sowie für alle Indexschlüsselspalten eines bestimmten Index erfasst.

Statistische Sichten

Statistische Sichten geben dem Optimierungsprogramm die Möglichkeit, präzisere Kardinalitätsschätzwerte zu berechnen. Eine Kardinalitätsschätzung ist der Prozess, bei dem das Optimierungsprogramm mithilfe von Statistikdaten die Größe von Abfrageteilergebnissen nach der Anwendung von Vergleichselementen oder nach einer Spaltenberechnung bestimmt. Die Genauigkeit der Kardinalitätsschätzungen hängt von den Vergleichselementen und den verfügbaren Statistikdaten ab. Es sind Statistikdaten verfügbar, welche die Verteilung von Datenwerten innerhalb einer Spalte darstellen und dadurch die Kardinalitätsschätzung verbessern können, wenn die Datenwerte ungleichmäßig verteilt sind. Darüber hinaus sind Statistikdaten verfügbar, welche die Anzahl der unterschiedlichen Werte in einer Gruppe von Spalten angeben. Durch diese Informationen lassen sich bessere Kardinalitätsschätzungen erzielen, wenn zwischen Spalten eine statistische Korrelation besteht. Sehr häufig allerdings können diese Statistikdaten komplexere Beziehungen möglicherweise nicht darstellen. Dies gilt zum Beispiel für den Filtereffekt von Vergleichselementen oder von Spaltenberechnungen mit korrelierten und ungleichmäßig verteilten Attributen (z. B. *make* = 'Honda' AND *model* = 'Accord'), für Vergleiche mit Ausdrucksergebnissen (z. B. *price* > MSRP + Dealer_markup), für Beziehungen, die mehrere Tabellen mit einbeziehen (z. B. *product.name* = 'Alloy wheels' AND *product.key* = *sales.product_key*), sowie für alle anderen Vergleichselemente oder Spaltenberechnungen mit unabhängigen Attributen und einfachen Vergleichsoperationen.

Statistische Sichten sind Sichten mit zugehörigen Statistikdaten, die zur Verbesserung von Kardinalitätsschätzungen für Abfragen genutzt werden können, bei denen sich die Sichtdefinition mit der Abfragedefinition überschneidet. Dies ist insofern eine leistungsfähige Funktionalität, als dass sie das Optimierungsprogramm mit präzisen Statistikdaten ausstattet, die zur Bestimmung von Kardinalitätsschätzwerten für Abfragen mit komplexen Gruppen von (möglicherweise korrelierten) Vergleichselementen verwendet werden können, die sich auf eine oder auch mehrere Tabellen beziehen.

Eine statistische Sicht braucht in einer Abfrage, die sie optimiert, nicht direkt angegeben zu werden. In den meisten Fällen können die Statistikdaten der Sicht genutzt werden, wenn sich die Definition der Sicht mit der Definition der Abfrage überschneidet. Voraussetzung für die Nutzung dieser neuen Funktionalität ist, dass eine Sicht mithilfe der Anweisung ALTER VIEW für die Optimierung aktiviert und die Systemkatalogtabellen mit Statistikdaten für diese Sicht gefüllt werden.

Verwenden statistischer Sichten

Eine Sicht muss für die Optimierung aktiviert werden, bevor die auf sie bezogenen Statistikdaten zur Optimierung einer Abfrage herangezogen werden können. Eine Sicht, die für die Optimierung aktiviert ist, wird als *statistische Sicht* bezeichnet.

Eine Sicht, die keine statistische Sicht ist, wird als für die Optimierung inaktiviert bezeichnet. Der Begriff *reguläre Sicht* bezieht sich auf eine Sicht, die für die Optimierung inaktiviert ist. Nach der Erstellung ist eine Sicht für die Optimierung zunächst inaktiviert.

- Zur Aktivierung einer Sicht für die Optimierung müssen Sie das Zugriffsrecht ALTER sowohl für die Sicht als auch für die Tabelle besitzen, für die die Sicht definiert ist.
- Zum Aufrufen des Dienstprogramms RUNSTATS für eine Sicht benötigen Sie eine der folgenden Berechtigungen:
  - Berechtigung SYSADM
  - Berechtigung SYSCTRL
  - Berechtigung SYSMAINT
  - Berechtigung DBADM
  - Zugriffsrecht CONTROL für die Sicht

  Darüber hinaus benötigen Sie die geeigneten Zugriffsrechte für den Zugriff auf Zeilen der Sicht. Insbesondere benötigen Sie für jede Tabelle, jede Sicht oder jeden Kurznamen, auf den in der Sichtdefinition verwiesen wird, eines der folgenden Zugriffsrechte (bzw. Berechtigungen):
  - Berechtigung SYSADM
  - Berechtigung DBADM
  - Zugriffsrecht CONTROL
  - Zugriffsrecht SELECT

Eine Sicht kann nicht für die Optimierung aktiviert werden, wenn eine der folgenden Bedingungen zutrifft:

- Die Sicht verweist direkt oder indirekt auf eine MQT (Materialized Query Table, gespeicherte Abfragetabelle). (Eine MQT oder statistische Sicht kann auf eine statistische Sicht verweisen.)
- Es handelt sich um eine funktionsunfähige Sicht.
- Es handelt sich eine typisierte Sicht.
- Eine weitere Sichtänderung wird in der gleichen Anweisung ALTER VIEW vorgenommen.

Wenn auf die Definition einer Sicht, die zur Aktivierung der Optimierung geändert wird, eine der folgenden Bedingungen zutrifft, wird die Anweisung ALTER VIEW ENABLE OPTIMIZATION zwar erfolgreich mit einer Warnung ausgeführt, jedoch werden die Statistikdaten vom Optimierungsprogramm nicht genutzt:

- Sie enthält Spaltenberechnungen (Aggregation) oder DISTINCT-Operationen.
- Sie enthält Gruppenoperationen mit UNION, EXCEPT oder INTERSECT.
- Sie enthält skalare Aggregatfunktionen (OLAP-Funktionen).

1. Aktivieren Sie die Sicht für die Optimierung.

   Eine Sicht kann mithilfe der neuen Klausel ENABLE OPTIMIZATION in der Anweisung ALTER VIEW für die Optimierung aktiviert werden. Eine Sicht, die für die Optimierung aktiviert wurde, kann nachfolgend mit der Klausel DISABLE OPTIMIZATION in der Anweisung ALTER VIEW für die Optimierung inaktiviert werden. Zum Beispiel würden Sie zur Aktivierung der Sicht `meinesicht` für die Optimierung, die folgende Anweisung eingeben:

   ```
   ALTER VIEW meinesicht ENABLE QUERY OPTIMIZATION
   ```

   Für eine Sicht, die für die Optimierung aktiviert ist, enthält die Spalte PROPERTY des entsprechenden SYSTABLES-Eintrags den Wert 'Y' an Zeichen-

position 13. Bei einer Sicht, die für die Optimierung inaktiviert ist, enthält die Spalte PROPERTY des entsprechenden SYSTABLES-Eintrags eine leere Stelle an Zeichenposition 13.

2. Führen Sie das Dienstprogramm RUNSTATS aus. Um zum Beispiel Statistikdaten für die Sicht `meinesicht` zu erfassen, geben Sie den folgenden Befehl ein:

   ```
   RUNSTATS ON TABLE db2dba.meinesicht
   ```

   Wenn Sie bei der Erfassung der Sichtstatistiken statistische Stichproben, einschließlich Verteilungstatistiken, für 10 % der Zeilen durch eine Stichprobenentnahme auf Zeilenebene erfassen möchten, geben Sie den folgenden Befehl ein:

   ```
   RUNSTATS ON TABLE db2dba.meinesicht WITH DISTRIBUTION TABLESAMPLE BERNOULLI (10)
   ```

   Anmerkung: Vor DB2 Version 9.1 bewirkte die Ausführung von RUNSTATS für eine statistische Sicht nur eine Vorbereitung der Katalogstatistiktabellen für manuelle Aktualisierungen und keine tatsächliche Erfassung von Statistikdaten. In DB2 Version 9.1 bewirkt die Ausführung von RUNSTATS für eine statistische Sicht die Erfassung von Statistikdaten. Dies bedeutet, dass das Dienstprogramm RUNSTATS möglicherweise wesentlich mehr Zeit zur Ausführung benötigt als früher, je nachdem, wie viel Daten von der Sicht zurückgegeben werden.

3. Aktualisierungen an Statistikdaten einer Sicht können zu Änderungen an den Plänen für Abfragen führen, deren Definition sich mit der Sichtdefinition überschneidet. Wenn solche Abfragen Teil statischer SQL-Pakete sind, müssen diese Pakete erneut gebunden (REBIND) werden, um die Vorteile der Pläne zu nutzen, die sich aus den neuen Statistiken ergeben.

Anzeigen der für die Optimierung relevanten Statistikdaten

Nur Statistikdaten, welche die Datenverteilung der Abfrage kennzeichnen, die eine statistische Sicht definiert, zum Beispiel CARD und COLCARD, werden vom Optimierungsprogramm für die Optimierung in Betracht gezogen. Die folgenden Statistikdaten, die sich auf Sichtdatensätze beziehen, können erfasst und durch das Optimierungsprogramm ausgewertet werden.

Tabellenstatistiken (SYSCAT.TABLES, SYSSTAT.TABLES)

- CARD - Die Anzahl von Zeilen im Ergebnis für eine Sicht

Spaltenstatistiken (SYSCAT.COLUMNS, SYSSTAT.COLUMNS)

- COLCARD - Die Anzahl unterschiedlicher Werte einer Spalte im Ergebnis für eine Sicht
- AVGCOLLEN - Durchschnittliche Länge von Spalten im Ergebnis für eine Sicht
- HIGH2KEY - Zweithöchster Wert einer Spalte im Ergebnis für eine Sicht
- LOW2KEY - Zweitniedrigster Wert einer Spalte im Ergebnis für eine Sicht
- NUMNULLS - Anzahl der NULL-Werte in einer Spalte des Ergebnisses für eine Sicht
- SUB_COUNT - Durchschnittliche Anzahl von Unterelementen in einer Spalte des Ergebnisses für eine Sicht
- SUB_DELIM_LENGTH - Durchschnittliche Länge der einzelnen Begrenzer, die die Unterelemente voneinander trennen

Spaltenverteilungsstatistiken (SYSCAT.COLDIST, SYSSTAT.COLDIST)

- DISTCOUNT - Wenn TYPE den Wert Q hat, die Anzahl unterschiedlicher Werte, die kleiner oder gleich dem Statistikwert COLVALUE sind

- SEQNO - Die Stelle in der Häufigkeitsrangfolge einer Folgenummer, die als Hilfe zur eindeutigen Bestimmung der Zeile in der Tabelle verwendet werden kann
- COLVALUE - Datenwert, für den Statistikdaten zur Häufigkeit oder zu Quantilwerten erfasst werden
- VALCOUNT - Häufigkeit, mit der der Datenwert in der Spalte einer Sicht auftritt, oder bei Quantilwerten die Anzahl der Werte, die kleiner oder gleich dem Datenwert (COLVALUE) sind

Statistikdaten, die keine Datenverteilung beschreiben, wie NPAGES und FPAGES, können zwar erfasst werden, werden jedoch vom Optimierungsprogramm ignoriert.

Szenario: Verbessern der Kardinalitätsschätzung mithilfe statistischer Sichten

In einem Data-Warehouse ändern sich die Informationen von Fakttabellen in der Regel recht dynamisch, während die Daten von Dimensionstabellen eher statischen Charakter haben. Daher können Attributdaten für Dimensionen häufig positiv oder negativ mit Attributdaten von Fakttabellen korreliert sein. Die gegenwärtig für das Optimierungsprogramm verfügbaren traditionellen Statistikdaten für Basistabellen geben dem Optimierungsprogramm keine Möglichkeit, tabellenübergreifende Beziehungen zu erkennen. Verteilungsstatistiken zu Spalten und Tabellen für Statistische Sichten (und MQTs) können dazu genutzt werden, die erforderlichen Informationen für das Optimierungsprogramm bereitzustellen, um derartige Kardinalitätsschätzfehler zu korrigieren.

Betrachten Sie die folgende Abfrage, die den Jahresumsatz für Golfschläger berechnet, die im Monat Juli jedes Jahres verkauft wurden:

```
SELECT  sum(f.sales_price), d2.year
FROM  product d1, period d2, daily_sales f
WHERE  d1.prodkey = f.prodkey
    AND  d2.perkey = f.perkey
    AND  d1.item_desc = 'golf club'
    AND  d2.month = 'JUL'
GROUP BY  d2.year
```

Ein Abfrageausführungsplan mit Sternjoin kann eine gute Wahl für diese Abfrage sein, sofern das Optimierungsprogramm feststellen kann, ob die einfache Gleichheitsverknüpfung (Semi-Join) von PRODUCT und DAILY_SALES oder die einfache Gleichheitsverknüpfung von PERIOD und DAILY_SALES die höhere Selektivität erzielt. Um einen effizienten Sternjoinplan generieren zu können, muss das Optimierungsprogramm in der Lage sein, die selektivste Semi-Join-Verknüpfung für den äußeren Part der logischen Indexverknüpfungsoperation über AND auszuwählen.

Data-Warehouses enthalten häufig Datensätze für Produkte, die nicht länger in den Geschäftsregalen verfügbar sind. Dies kann dazu führen, dass die Verteilung von PRODUCT-Spalten nach dem Join völlig anders aussieht als ihre Verteilung vor dem Join. Da das Optimierungsprogramm mangels besserer Informationen die Selektivität lokaler Vergleichselemente allein auf der Grundlage von Statistiken zu den Basistabellen bestimmt, schätzt es die Selektivität des Vergleichselements *item_desc = 'golf club'* möglicherweise viel zu hoch ein.

Wenn Golfschläger historisch betrachtet zum Beispiel 1 % der hergestellten Produkte darstellen, jedoch jetzt 20 % der aktuellen Umsätze ausmachen, würde das

Optimierungsprogramm mit hoher Wahrscheinlichkeit die Selektivität des Vergleichselements *item_desc = 'golf club'* überschätzen, da keine Statistikdaten vorhanden sind, die die Verteilung von *item_desc* nach dem Join beschreiben. Und wenn außerdem die Umsätze in allen zwölf Monaten gleichermaßen wahrscheinlich sind, läge die Selektivität des Vergleichselements *month = 'JUL'* bei ca. 8 %. Daher würde der Schätzfehler für die Selektivität des Vergleichselements *item_desc = 'golf club'* das Optimierungsprogramm fälschlicherweise dazu veranlassen, die scheinbar selektivere Semi-Join-Operation zwischen PRODUCT und DAILY_SALES als äußeren Part der logischen AND-Verknüpfung der Indizes des Sternjoinplans auszuführen.

Im folgenden Beispiel wird die Bereitstellung statistischer Sichten zur Lösung dieses Typs von Problem schrittweise veranschaulicht.

Das folgende Beispiel zeigt eine Datenbank aus einem typischen Data-Warehouse, in dem STORE, CUSTOMER, PRODUCT, PROMOTION und PERIOD die Dimensionstabellen sind und DAILY_SALES die Fakttabelle ist.

Tabelle 67. STORE (63 Zeilen)

Spalte	storekey	store_number	city	Status	district	...
Attribut	integer	char(2)	char(20)	char(5)	char(14)	...
	not null					
	Primärschlüssel					

Tabelle 68. CUSTOMER (1.000.000 Zeilen)

Spalte	custkey	name	address	age	gender	...
Attribut	integer	char(30)	char(40)	smallint	char(1)	...
	not null					
	Primärschlüssel					

Tabelle 69. PRODUCT (19.450 Zeilen)

Spalte	prodkey	category	item_desc	price	cost	...
Attribut	integer	integer	char(30)	decimal(11)	decimal(11)	...
	not null					
	Primärschlüssel					

Tabelle 70. PROMOTION (35 Zeilen)

Spalte	promokey	promotype	promodesc	promovalue	...
Attribut	integer	integer	char(30)	decimal(5)	...
	not null				
	Primärschlüssel				

Tabelle 71. PERIOD (2922 Zeilen)

Spalte	perkey	calendar_date	month	period	year	...
Attribut	integer	date	char(3)	smallint	smallint	...
	not null					
	Primärschlüssel					

Tabelle 72. DAILY_SALES (754.069.426 Zeilen)

Spalte	storekey	custkey	prodkey	promokey	perkey	sales_price	...
Attribut	integer	integer	integer	integer	integer	decimal(11)	...

Nehmen Sie an, die Unternehmensmanager möchten herausfinden, ob Kunden ein Produkt noch einmal kaufen, wenn ihnen ein Rabatt für das gleiche Produkt bei einem weiteren Besuch angeboten wird. Nehmen Sie außerdem an, dass die Studie nur für das Geschäft (STORE) '01' durchgeführt wird, das 18 Niederlassungen im Land hat. Tabelle 73 zeigt die verschiedenen Kategorien von Verkaufsförderungstypen (Promotionstypen) mit den Prozentsätzen ihres jeweiligen Anteils an allen Verkaufsfördermaßnahmen.

Tabelle 73. PROMOTION (35 Zeilen)

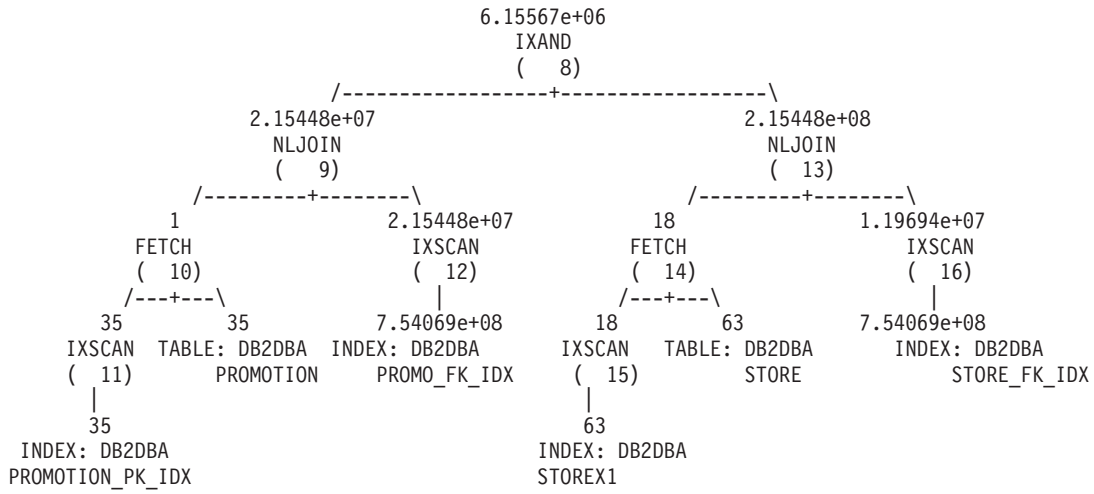
promotype	promodesc	COUNT(promotype)	percentage of promotions
1	Return customers	1	2,86%
2	Coupon	15	42.86%
3	Advertisement	5	14.29%
4	Manager's special	3	8.57%
5	Overstocked items	4	11.43%
6	End aisle display	7	20.00%

Die Tabelle zeigt, dass nur 2,86 % aller 35 Typen von angebotenen Verkaufsförderungsmaßnahmen Rabatte für wiederkehrende Kunden (Return customers) sind (1÷35 ≈ 0,0286).

Nun wird die folgende Abfrage ausgeführt, die die Anzahl 12.889.514 zurückgibt:

```
SELECT  count(*)
FROM  store d1, promotion d2, daily_sales f
WHERE  d1.storekey = f.storekey
    AND  d2.promokey = f.promokey
    AND  d1.store_number = '01'
    AND  d2.promotype = 1
```

Diese Abfrage wird nach dem folgenden, vom Optimierungsprogramm generierten Plan ausgeführt. Bei jedem Knoten dieses Diagramms stellt der obere numerische Wert die Kardinalitätsschätzung, die zweite Zeile den Operatortyp und die Nummer in Klammern die Operator-ID dar.



Beim Join mit Verschachtelungsschleife (NLJOIN) Nummer 9 schätzt das Optimierungsprogramm, dass rund 2,86 % des Produktverkaufs auf Kunden zurückzuführen ist, die zurückkommen, um die gleichen Produkte zu einem reduzierten Preis zu kaufen (2,15448e+07 ÷7,54069e+08 ≈ 0,0286). Beachten Sie, dass dies vor und nach dem Join der Tabelle PROMOTION mit der Tabelle DAILY_SALES der gleiche Prozentsatz ist. In Tabelle 74 sind die Kardinalitätsschätzwerte und ihre Prozentsätze (Filtereffekt) vor und nach dem Join zusammengefasst.

Tabelle 74. Geschätzte Kardinalitäten vor und nach dem Join mit der Tabelle DAILY_SALES

	Vor dem Join		**Nach dem Join**	
Vergleichselemente	**Anzahl**	**Prozentsatz qualifizierter Zeilen**	**Anzahl**	**Prozentsatz qualifizierter Zeilen**
store_number = '01'	18	28,57%	2,15448e+08	28,57%
promotype=1	1	2,86%	2,15448e+07	2,86%

Da die Wahrscheinlichkeit von *promotype = 1* geringer ist als die von *store_number = '01'*, wählt das Optimierungsprogramm die Semi-Join-Verknüpfung zwischen den Tabellen PROMOTION und DAILY_SALES als äußeren Teil der logischen AND-Verknüpfung von Indizes des Sternjoinplans. Dies führt zu einer geschätzten Anzahl von ca. 6.155.670 Produkten, die über den Verkaufsförderungstyp 1 verkauft wurden. Diese inkorrekte Kardinalitätsschätzung liegt um den Faktor 2,09 unter der tatsächlichen Anzahl (12.889.514 ÷ 6.155.670 ≈ 2,09).

Was ist die Ursache dafür, dass das Optimierungsprogramm die Anzahl der durch die beiden Vergleichselemente qualifizierten Datensätze nur auf die Hälfte der tatsächlichen Anzahl schätzt? Das Geschäft '01' repräsentiert ca. 28,57 % aller Geschäfte. Was wäre, wenn andere Geschäfte mehr Umsätze hätten als Geschäft '01' (weniger als 28,57 %)? Oder wenn das Geschäft '01' tatsächlich die meisten Produktumsätze erzielt hätte (mehr als 28,57 %)? In ähnlicher Weise können auch die 2,86 % der durch Verkaufsförderungstyp 1 verkauften Produkte in Tabelle 74 irreführend sein. Der tatsächliche Prozentsatz in der Tabelle DAILY_SALES könnte sehr wohl ein anderer Wert sein als der projektierte.

Statistische Sichten können dem Optimierungsprogramm helfen, die Schätzwerte zu korrigieren und Über- bzw. Unterschätzungen zu verringern. Zunächst müssen zwei statistische Sichten erstellt werden, welche die beiden Semi-Join-Verknüpfungen in der obigen Abfrage darstellen. Die erste statistische Sicht stellt die Verteilungsdaten von Geschäften für alle täglichen Umsätze bereit. Die zweite statistische Sicht stellt die Verteilung von Verkaufsförderungstypen für alle täglichen Umsätze dar. Beachten Sie, dass diese statistischen Sichten die Verteilungsdaten für eine beliebige Geschäftsnummer bzw. einen beliebigen Verkaufsförderungstyp bereitstellen kann. In diesem Beispiel wird eine Stichprobenrate von 10 % verwendet, um die Datensätze aus der Tabelle DAILY_SALES für die jeweilige Sicht abzurufen und in globalen temporären Tabellen zu speichern. Anschließend werden diese Tabelle abgefragt, um die erforderlichen Statistikdaten zur Aktualisierung der beiden statistischen Sichten zu sammeln.

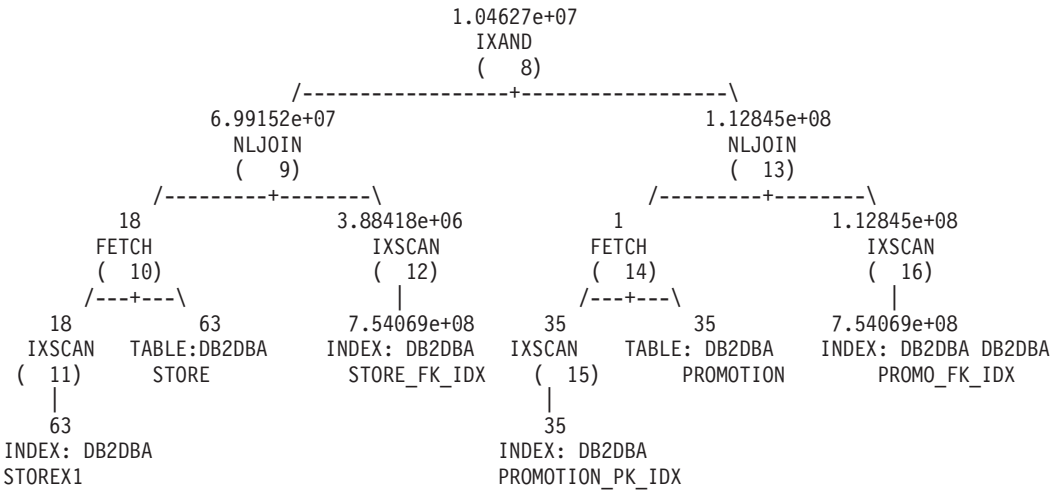
1. Erstellen Sie eine Sicht, die den Join der Tabelle STORE mit der Tabelle DAILY_SALES darstellt.

```
CREATE  view sv_store_dailysales as
   (SELECT  s.*
    FROM  store s, daily_sales ds
    WHERE  s.storekey = ds.storekey)
```

2. Erstellen Sie eine Sicht, die den Join der Tabelle PROMOTION mit der Tabelle DAILY_SALES darstellt.

```
CREATE  view sv_promotion_dailysales as
   (SELECT  p.*
    FROM  promotion.p, daily_sales ds
    WHERE  p.promokey = ds.promokey)
```

3. Machen Sie die Sichten zu statistischen Sichten, indem Sie die Abfrageoptimierung für sie aktivieren:
   - `ALTER VIEW  sv_store_dailysales  ENABLE QUERY OPTIMIZATION`
   - `ALTER VIEW  sv_promotion_dailysales  ENABLE QUERY OPTIMIZATION`
4. Führen Sie das Dienstprogramm RUNSTATS aus, um Statistikdaten für die Sichten zu erfassen:
   - `RUNSTATS  on table db2dba.sv_store_dailysales  WITH DISTRIBUTION`
   - `RUNSTATS  on table db2dba.sv_promotion_dailysales WITH DISTRIBUTION`
5. Führen Sie die Abfrage erneut aus, damit sie erneut optimiert werden kann. Bei der Reoptimierung gleicht das Optimierungsprogramm die Sicht SV_STORE-_DAILYSALES und die Sicht SV_PROMOTION_DAILYSALES mit der Abfrage ab und verwendet die Statistikdaten für die Sichten, um die Kardinalitätsschätzung der Semi-Join-Verknüpfungen zwischen der Fakttabelle und den Dimensionstabellen anzupassen. Dadurch wird es veranlasst, die Reihenfolge der Semi-Join-Verknüpfungen, die ohne diese Statistiken ausgewählt wurden, umzukehren. Der neue Plan sieht folgendermaßen aus:



Beachten Sie, dass jetzt die Semi-Join-Verknüpfung zwischen den Tabellen STORE und DAILY_SALES im äußeren Part des Plans zur logischen Verknüpfung der Indizes über AND ausgeführt wird. Dies ist darauf zurückzuführen, dass die beiden statistischen Sichten dem Optimierungsprogramm im Wesentlichen mitteilen, dass das Vergleichselement 'store_number = '01'' mehr Zeilen herausfiltert als das Vergleichselement 'promotype = 1'. In Tabelle 75 auf Seite 482 sind die Kardinalitätsschätzwerte und ihre Prozentsätze (Filtereffekt) vor und nach dem Join für jeden Semi-Join zusammengefasst.

Beachten Sie, dass jetzt die Semi-Join-Verknüpfung zwischen den Tabellen STORE und DAILY_SALES im äußeren Part des Plans zur logischen Verknüpfung der Indizes über AND ausgeführt wird. Dies ist darauf zurückzuführen, dass die beiden statistischen Sichten dem Optimierungsprogramm im Wesentlichen mitteilen, dass das Vergleichselement 'store_number = '01'' mehr Zeilen herausfiltert als das Vergleichselement 'promotype = 1'. Hier schätzt das Optimierungsprogramm die Anzahl der verkauften Produkte auf ca. 10.462.700. Dieser Schätzwert liegt um den Faktor 1,23 (12.889.514 ÷ 10.462.700 ≈ 1,23) unter der tatsächlichen Anzahl, was eine erhebliche Verbesserung gegenüber dem Schätzwert in Tabelle 74 auf Seite 480

darstellt. In Tabelle 75 sind die Kardinalitätsschätzwerte und ihre Prozentsätze (der Filtereffekt) für jedes Vergleichselement vor und nach dem Join zusammengefasst.

Tabelle 75. Geschätzte Kardinalitäten vor und nach dem Join mit der Tabelle DAILY_SALES

	Vor dem Join		**Nach dem Join** *Ohne statistische Sichten*		**Nach dem Join** *Mit statistischen Sichten*	
Vergleichs-elemente	**Anzahl**	**Prozentsatz qualifizierter Zeilen**	**Anzahl**	**Prozentsatz qualifi-zierter Zeilen**	**Anzahl**	**Prozentsatz quali-fizierter Zeilen**
store_number = '01'	18	28,57%	2,15448e+08	28,57%	6,99152e+07	9,27%
promotype=1	1	2,86%	2,15448e+07	2,86%	1,12845e+08	14,96%

Jetzt schätzt das Optimierungsprogramm die Anzahl der verkauften Produkte auf ca. 10.462.700. Dieser Schätzwert weicht um den Faktor 1,23 von der tatsächlichen Anzahl ab, was eine erhebliche Verbesserung gegenüber dem Schätzwert ohne statistische Sichten ist.

Teil 6. 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.

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 76. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar
Administrative API Reference	SC23-5842-01	Ja
Administrative Routines and Views	SC23-5843-01	Nein
Call Level Interface Guide and Reference, Volume 1	SC23-5844-01	Ja
Call Level Interface Guide and Reference, Volume 2	SC23-5845-01	Ja
Command Reference	SC23-5846-01	Ja
Dienstprogramme für das Versetzen von Daten Handbuch und Referenz	SC12-3917-01	Ja
Datenrecovery und hohe Verfügbarkeit Handbuch und Referenz	SC12-3919-01	Ja
Datenserver, Datenbanken und Datenbankobjekte	SC12-3912-01	Ja
Datenbanksicherheit	SC12-3914-01	Ja
Developing ADO.NET and OLE DB Applications	SC23-5851-01	Ja
Developing Embedded SQL Applications	SC23-5852-01	Ja
Developing Java Applications	SC23-5853-01	Ja
Developing Perl and PHP Applications	SC23-5854-01	Nein
Developing User-defined Routines (SQL and External)	SC23-5855-01	Ja
Getting Started with Database Application Development	GC23-5856-01	Ja

Tabelle 76. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GC12-3922-01	Ja
Internationalisierung	SC12-3916-01	Ja
Fehlernachrichten, Band 1	GI11-3098-00	Nein
Fehlernachrichten, Band 2	GI11-3099-00	Nein
Migration	GC12-3921-01	Ja
Net Search Extender Verwaltung und Benutzerhandbuch	SC12-3979-01	Ja
Partitionierung und Clustering	SC12-3915-01	Ja
Query Patroller Verwaltung und Benutzerhandbuch	SC12-3977-00	Ja
IBM Data Server-Clients - Einstieg	GC12-3924-01	Nein
DB2-Server - Einstieg	GC12-3923-01	Ja
Spatial Extender und Geodetic Data Management Feature Benutzer- und Referenzhandbuch	SC12-3978-01	Ja
SQL Reference, Volume 1	SC23-5861-01	Ja
SQL Reference, Volume 2	SC23-5862-01	Ja
Systemmonitor Handbuch und Referenz	SC12-3918-01	Ja
Fehlerbehebung	GI11-3097-01	Nein
Optimieren der Datenbankleistung	SC12-3913-01	Ja
Lernprogramm für Visual Explain	SC12-3932-00	Nein
Neue Funktionen	SC12-3928-01	Ja
Workload-Manager Handbuch und Referenz	SC12-3929-01	Ja
pureXML - Handbuch	SC12-3930-01	Ja
XQuery - Referenz	SC12-3931-01	Nein

Tabelle 77. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar
DB2 Connect Personal Edition - Einstieg	GC12-3926-01	Ja
DB2 Connect-Server - Einstieg	GC12-3927-01	Ja
DB2 Connect Benutzerhandbuch	SC12-3925-01	Ja

Tabelle 78. Technische Informationen zu Information Integration

Name	IBM Form	In gedruckter Form verfügbar
Information Integration: Föderierte Systeme - Verwaltung	SC12-3759-01	Ja
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Ja
Information Integration: Konfiguration föderierter Datenquellen	SC12-3777-01	Nein
Information Integration: SQL Replication - Handbuch und Referenz	SC12-3782-01	Ja
Information Integration: Replikation und Event-Publishing - Einführung	GC12-3779-01	Ja

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 486.

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 eine 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> das Verzeichnis der Programmdateien (Program Files) angibt.
     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-Produkten zu unterstützen.

DB2-Dokumentation
Informationen zur Fehlerbehebung stehen im Handbuch DB2-Fehlerbehebung oder im Abschnitt zur Unterstützung und Fehlerbehebung 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-Produkte 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/udb/support.html 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, Anwendungs-

programme 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:

Marken

Folgende Namen sind Marken oder eingetragene Marken der International Business Machines Corporation in den USA und/oder anderen Ländern.

pureXML	OS/390
DB2 Connect	DB2 Universal Database
Redbooks	z/OS
developerWorks	System i
Informix	IBM
DB2	zSeries
AIX	System z9
Lotus	Tivoli
DRDA	System z
Domino	ibm.com
POWER	WebSphere

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 Java-basierten Marken 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/oder anderen Ländern.
- Microsoft und Windows 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

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

SC12-3913-01