IBM DB2 10.5
for Linux, UNIX and Windows

Partitionierung und Clustering

Aktualisierung: Juli 2013

SC12-4906-00

IBM DB2 10.5
for Linux, UNIX and Windows

Partitionierung und Clustering

Aktualisierung: Juli 2013

SC12-4906-00

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in Anhang E, „Bemerkungen", auf Seite 491 gelesen werden.

Impressum

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 10.5 for Linux, UNIX, and Windows, Partitioning and Clustering Guide,
IBM Form SC27-5532-00,
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:
TSC Germany
Kst. 2877
Juli 2013

Inhaltsverzeichnis

Zu diesem Handbuch

Für die Funktionalität des DB2-Verwaltungssystems für relationale Datenbanken spielen die Partitionierungs- und Clusteringfunktionen eine bedeutende Rolle. Sie bieten Administratoren und Systembedienern die Möglichkeit, die Leistung der Datenbank effektiv zu verbessern und viele der Datenbankobjekte auf Hardwareressourcen zu verteilen. Ein schnellerer Datenabruf sowie die Möglichkeit, Objekte über ständig wachsende Hardwareressourcen verteilen und so von Parallelität und Speicherkapazität profitieren zu können, führen letztendlich zu einer höheren Produktivität. Dieses Handbuch stellt durch die spezielle Auswahl von Themen aus der DB2-Datenbankbibliothek eine zentrale Quelle für umfassende Informationen bereit, die ausschließlich den Bereich Planung, Entwurf, Implementierung, Verwendung und Pflege von Datenbankpartitionen, Tabellenpartitionen, Tabellenclustern, Tabellenbereichsclustern, MDC-Tabellen und Parallelität behandeln.

Zielgruppe

Dieses Handbuch ist in erster Linie für Datenbankadministratoren, Systemadministratoren, Sicherheitsadministratoren und Systembediener vorgesehen, die partitionierte Datenbanken bzw. Datenbanken mit Clustering für den Zugriff durch lokale und ferne Clients entwerfen, implementieren oder verwalten müssen. Es bietet sich auch für Anwendungsentwickler und andere Benutzer an, die eine umfassende Informationsquelle benötigen, um sich Kenntnisse über die Verwaltung und den Betrieb des DB2-Verwaltungssystems für relationale Datenbanken in Bezug auf die Partitionierungs-, Clustering- und Parallelitätsfunktionalität anzueignen. Für alle, die eine zukünftige Implementierung einer beliebigen oder auch aller erläuterten Hauptfunktionen in Betracht ziehen, stellt dieses Handbuch eine hervorragende Informationsressource dar.

Aufbau des Handbuchs

Diese Auswahl von Themen aus der DB2-Bibliothek stellt eine zentrale Quelle für umfassende Informationen bereit, die ausschließlich den Bereich der Partitionierungs-, Clustering- und Parallelitätsfunktionen von DB2 behandeln. Dieses Handbuch ist aus Gründen des Nutzungskomforts und der Effizienz in sechs Hauptteile untergliedert, deren erste fünf die primären Verwaltungsthemen darlegen, die für Administratoren, Systembediener und Anwendungsentwickler von Interesse sind. Ein Thema innerhalb eines Hauptteiles dieses Handbuchs kann Themen zugeordnet werden, die den Inhalt anderer Handbücher in der DB2-Bibliothek darstellen. Dies erleichtert die Arbeit mit Querverweisen auf allgemeinere Informationen, die eine Vielzahl anderer DB2-Funktionen und -Objekte behandeln. Wenn Sie zum Beispiel ein Thema in Teil 4, Kapitel 20, zu Optimierungsstrategien für MDC-Tabellen und den durch sie erzielbaren Leistungsverbesserungen gelesen haben, möchten Sie sich vielleicht über weitere allgemeine Leistungsverbesserungen informieren, die für reguläre Tabellen konfiguriert werden können, indem Sie die Informationen des Handbuchs *Optimieren der Datenbankleistung* lesen, die diesem bestimmten Beispielthema zugeordnet sind. In der nachfolgenden Tabelle 1 finden Sie die Hauptteile dieses Handbuchs und ihre Querverweise auf andere Handbücher, die zusätzliche Informationen zu anderen DB2-Objekten und -Funktionen im Hinblick auf ein ähnliches Thema enthalten.

Tabelle 1. Zuordnung der Teile dieses Handbuchs zu anderen Handbüchern in der DB2-Bibliothek

Teile im Handbuch 'Partitionierung und Clustering'	**Zuordnung zu Handbüchern in der DB2-Bibliothek**
Teil 1. Planung und Entwurf	*Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen* *Datenbanksicherheit*
Teil 2. Installationshinweise	*Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen* *DB2-Server - Installation*
Teil 3. Implementierung und Pflege	*Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz* *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen* *Upgrade auf DB2 Version 10.5* *Datenbanküberwachung - Handbuch und Referenz* *XQuery - Referenz*
Teil 4. Leistungsaspekte	*Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen* *Fehlerbehebung und Optimieren der Datenbankleistung*
Teil 5. APIs, Befehle und SQL-Anweisungen zur Verwaltung	*Administrative API Reference* *Administrative Routines and Views* *Command Reference* *Developing ADO.NET and OLE DB Applications* *Developing Embedded SQL Applications* *Developing Java Applications* *Developing Perl, PHP, Python, and Ruby on Rails Applications* *Developing User-defined Routines (SQL and External)* *Getting Started with Database Application Development* *SQL Reference Volume 1* *SQL Reference Volume 2*
Teil 6. Anhänge	*Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* *DB2-Server - Installation* *SQL Reference Volume 1*

Die folgenden übergeordneten Themenbereiche werden in den Kapiteln dieses Handbuchs behandelt:

Teil 1. Planung und Entwurf

Alle folgenden Kapitel enthalten Informationen zu Konzepten, die für die Planung und den Entwurf von Datenbanken bzw. Tabellen relevant sind, die entweder partitioniert, mit Clustering genutzt oder in parallelen Datenbanksystemen verwendet werden sollen.

- Kapitel 1, „Partitionierte Datenbanken und Tabellen", enthält eine Einführung in relevante Konzepte in Bezug auf die Funktionsmerkmale und Vorteile der Partitionierung von Datenbanken und Tabellen.
- Kapitel 2, „Bereichsclustertabellen", enthält allgemeine Informationen zu den Funktionsmerkmalen und Vorteilen der Verwendung von Bereichsclustertabellen.
- Kapitel 3, „Tabellen mit mehrdimensionalem Clustering (MDC)", enthält eine Beschreibung der Verwendung des mehrdimensionalen Clusterings als elegante Methode für das Clustering von Daten in Tabellen.
- Kapitel 4, „Parallele Datenbanksysteme", erläutert, wie sich Parallelität nutzen lässt, um eine beträchtliche Leistungsverbesserung zu erzielen.

Teil 2. Installationshinweise

Die folgenden Kapitel enthalten Informationen zu den Tasks der Installationsvorbereitung sowie der eigentlichen Installation, die zur Vorbereitung der Datenbankpartitionierung erforderlich sind.

- Kapitel 5, „Installationsvoraussetzungen", beschreibt die Voraussetzungen und Einschränkungen im Hinblick auf die Vorbereitung eines DB2-Servers, der in eine Umgebung mit partitionierten Datenbanken integriert werden soll.
- Kapitel 6, „Installationsvorbereitung", behandelt zusätzliche Tasks zur Installationsvorbereitung bei Verwendung von UNIX- und Linux-Betriebssystemen.
- Kapitel 7, „Installieren des DB2-Serverprodukts", beschreibt die Installation von Datenbankpartitionsservern und die Konfiguration einer Umgebung mit partitionierten Datenbanken.
- Kapitel 8, „Installationsnachbereitung", beschreibt, wie die Installation auf Windows-, UNIX- und Linux-Betriebssystemen geprüft wird.

Teil 3. Implementierung und Pflege

Nach Abschluss der Schritte zur Planung, zum Entwurf und zur Installation wird in den folgenden Kapiteln erläutert, wie die Funktionen bzw. Objekte, für die zuvor Vorbereitungen getroffen wurden, implementiert und gepflegt werden.

- Kapitel 9, „Vorbereitung zum Erstellen einer Datenbank", beschreibt die Gesichtspunkte, die vor der Erstellung einer Datenbank zu beachten sind, wie zum Beispiel die Aktivierung der Parallelität, die Erstellung von Umgebungen mit partitionierten Datenbanken, die Erstellung und Konfiguration von Datenbankpartitionen sowie die Einrichtung der Kommunikation zwischen Datenbankpartitionen.
- Kapitel 10, „Erstellen und Verwalten von Umgebungen mit partitionierten Datenbanken", beschreibt die Erstellung und Verwaltung von Datenbankpartitionen und Partitionsgruppen.
- Kapitel 11, „Erstellen von Tabellen und anderen zugehörigen Tabellenobjekten", enthält Informationen zur Erstellung und Einrichtung von partitionierten Tabellen, Bereichsclustertabellen (RCT) und MDC-Tabellen.

- Kapitel 12, „Ändern einer Datenbank", beschreibt, wie eine Instanz bzw. eine Datenbank geändert wird.
- Kapitel 13, „Ändern von Tabellen und anderen zugehörigen Tabellenobjekten", enthält Informationen zur Änderung von partitionierten Tabellen.
- Kapitel 14, „Laden von Daten", behandelt Aspekte für das Laden von Daten im Hinblick auf Parallelität, mehrdimensionales Clustering und partitionierte Tabellen.
- Kapitel 15, „Laden von Daten in einer Umgebung mit partitionierten Datenbanken", beschreibt, wie Datenladeoperationen in Umgebungen mit partitionierten Datenbanken gestartet, fortgesetzt, erneut gestartet oder beendet werden.
- Kapitel 16, „Migration von Umgebungen mit partitionierten Datenbanken", enthält eine kurze Übersicht über die Migration partitionierter Datenbanken und Querverweise auf detailliertere Informationen.
- Kapitel 17, „Verwenden von Momentaufnahme- und Ereignismonitoren", enthält relevante Informationen zur Verwendung von Snapshot Monitor-Ergebnissen zur Überwachung einer Tabellenreorganisation und zur Bewertung des globalen Status eines partitionierten Datenbanksystems. Darüber hinaus wird die Verwendung der Anweisung CREATE EVENT MONITOR erläutert.
- Kapitel 18, „Entwickeln einer geeigneten Backup- und Recovery-Strategie", beschreibt die Konzepte, die hinter einer Recovery nach Systemabsturz in einer Umgebung mit partitionierten Datenbanken stehen. Diese Informationen sind bei der Entwicklung von Backup- und Recovery-Strategien hilfreich, bevor ein Fehler auftritt.
- Kapitel 19, „Fehlerbehebung", enthält eine kurze Übersicht über die Fehlerbehebung sowie nützliche Informationen dazu, wie Befehle, die bei der Fehlerbehebung hilfreich sind, wie zum Beispiel **db2trc**, auf allen Computern in der Instanz bzw. auf allen Datenbankpartitionsservern ausgeführt werden.

Teil 4. Leistungsaspekte

Die folgenden Kapitel enthalten relevante Informationen, mit deren Hilfe Sie die Leistung Ihrer Umgebung mit partitionierten Datenbanken bzw. Ihrer Clusterumgebung verbessern können.

- Kapitel 20, „Leistungsaspekte beim Datenbankentwurf", beschreibt die Funktionsmerkmale der Tabellenpartitionierung und des mehrdimensionalen Clustering (MDC) hinsichtlich der Leistungsverbesserung und erläutert jeweils entsprechende Optimierungsstrategien.
- Kapitel 21, „Indizes", enthält Informationen zu Konzepten, die für das Verständnis von Indizes für partitionierte Tabellen hilfreich sind.
- Kapitel 22, „Designadvisor", beschreibt die Verwendung des Designadvisors zum Abrufen von Informationen zur Migration von einer Einzelpartitionsdatenbank auf eine Mehrpartitionsdatenbank sowie zum Abrufen von Empfehlungen zur Verteilung der Daten und zur Erstellung neuer Indizes, MQTs und MDC-Tabellen.
- Kapitel 23, „Verwalten des gemeinsamen Zugriffs", enthält Informationen zu Sperrmodi.
- Kapitel 24, „Agentenverwaltung", beschreibt die Optimierung von Datenbankagenten, die zur Verarbeitung von Anwendungsanforderungen verwendet werden.

- Kapitel 25, „Optimieren von Zugriffsplänen", beschreibt, wie ein Zugriffsplan verbessert wird, erläutert, wie das Optimierungsprogramm Informationen aus verschiedenen Suchen zur Optimierung von Datenzugriffsstrategien nutzt, und enthält Informationen zu Joinstrategien. Alle diese Optimierungsmethoden dienen der Verbesserung der Leistung für Umgebungen mit partitionierten Datenbanken, für Clustertabellen sowie für Systeme, auf denen Parallelität genutzt wird.
- Kapitel 26, „Datenumverteilung", enthält Informationen, die Ihnen bei der Feststellung helfen, ob eine Datenumverteilung durchgeführt werden sollte, und beschreibt, wie Daten auf Datenbankpartitionen umverteilt werden, wenn dies erforderlich ist.
- Kapitel 27, „Konfigurieren des Speichers mit automatischer Leistungsoptimierung", behandelt die Verwendung der Funktion für die automatische Speicherleistungsoptimierung in einer Umgebung mit partitionierten Datenbanken und gibt Empfehlungen zur Konfiguration.
- Kapitel 28, „DB2-Konfigurationsparameter und -Variablen", enthält Informationen dazu, wie Datenbankkonfigurationsparameter und Umgebungsvariablen über mehrere Partitionen hinweg definiert werden, und listet die Parameter und Variablen auf, die für die Funktionalität von Umgebungen mit partitionierten Datenbanken und der Parallelität relevant sind.

Teil 5. APIs, Befehle und SQL-Anweisungen zur Verwaltung

In den folgenden Kapiteln werden Informationen zu APIs, Befehlen und SQL-Elementen für die Verwaltung zusammengefasst, die für Umgebungen mit partitionierten Datenbanken relevant sind.

- Kapitel 29, „APIs zur Verwaltung", enthält Informationen zu den APIs, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 30, „Befehle", enthält Informationen zu den Befehlen, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 31, „SQL-Sprachelemente", beschreibt datenbankpartitionskompatible Datentypen und Sonderregister.
- Kapitel 32, „SQL-Funktionen", beschreibt SQL-Funktionen, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 33, „SQL-Anweisungen", beschreibt SQL-Anweisungen, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 34, „Unterstützte SQL-Verwaltungsroutinen und Verwaltungssichten", beschreibt SQL-Routinen und Sichten, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.

Teil 6. Anhänge

- Anhang A, „Installieren als Benutzer ohne Rootberechtigung", beschreibt die Installation des DB2-Datenbankprodukts durch einen Benutzer, der keine Rootberechtigung hat, auf UNIX- und Linux-Betriebssystemen.
- Anhang B, „Verwenden von Backup", beschreibt die Verwendungsweise des Befehls **BACKUP DATABASE**.
- Anhang C, „Katalogsichten für Umgebungen mit partitionierten Datenbanken", listet die Katalogsichten auf, die speziell für eine Umgebung mit partitionierten Datenbanken relevant sind.

Hervorhebungskonventionen

In diesem Handbuch werden die folgenden Hervorhebungskonventionen verwendet.

Fett	Kennzeichnet Befehle, Schlüsselwörter und andere Elemente, deren Namen vom System vordefiniert werden.
Kursiv	Kennzeichnet folgende Elemente: • Namen oder Werte (Variablen), die vom Benutzer angegeben werden müssen • Begriffe, die allgemein hervorgehoben werden sollen • Neue Begriffe, die eingeführt werden • Verweise auf andere Informationsquellen
`Monospace`	Kennzeichnet folgende Elemente: • Dateien und Verzeichnisse • Informationen, die Sie laut Anweisung in eine Eingabeaufforderung oder ein Fenster eingeben sollen • Beispiele für bestimmte Datenwerte • Beispiele für Text, der einer Ausgabe des Systems ähnlich ist • Beispiele für Systemnachrichten • Muster für Programmcode

Teil 1. Planung und Entwurf

Kapitel 1. Partitionierte Datenbanken und Tabellen

Einrichten von Umgebungen mit partitionierten Datenbanken

Die Entscheidung, eine Mehrpartitionsdatenbank zu erstellen, muss vor der Erstellung der betreffenden Datenbank getroffen werden. Im Rahmen Ihrer Entscheidungen beim Datenbankentwurf müssen Sie festlegen, ob die Leistungsverbesserungen genutzt werden sollen, die eine Datenbankpartitionierung zu bieten hat.

Informationen zu diesem Vorgang

Auch in einer Umgebung mit partitionierten Datenbanken verwenden Sie den Befehl **CREATE DATABASE** oder die Funktion 'sqlecrea()' zum Erstellen einer Datenbank. Unabhängig von der verwendeten Methode kann die Anforderung über eine beliebige der Partitionen erfolgen, die in der Datei `db2nodes.cfg` aufgelistet sind. Die Datei `db2nodes.cfg` ist die Konfigurationsdatei des Datenbankpartitionsservers.

Außer in der Windows-Betriebssystemumgebung kann jeder Editor zum Anzeigen und Ändern des Inhalts der Konfigurationsdatei des Datenbankpartitionsservers (`db2nodes.cfg`) verwendet werden. In der Windows-Betriebssystemumgebung müssen Sie die Befehle **db2ncrt** und **db2nchg** zum Erstellen und Ändern der Konfigurationsdatei des Datenbankpartitionsservers verwenden.

Vor der Erstellung einer Mehrpartitionsdatenbank müssen Sie die Datenbankpartition auswählen, die als Katalogknoten für die Datenbank fungieren soll. Anschließend können Sie die Datenbank direkt von dieser Datenbankpartition oder von einem fernen Client aus erstellen, der mit dieser Datenbankpartition verbunden ist. Die Datenbankpartition, zu der Sie die Verbindung (mit ATTACH) herstellen, um den Befehl **CREATE DATABASE** auszuführen, wird zur *Katalogpartition* für diese spezielle Datenbank.

Die Katalogpartition ist die Datenbankpartition, in der alle Systemkatalogtabellen gespeichert werden. Jeglicher Zugriff auf die Systemtabellen muss über diese Datenbankpartition erfolgen. Alle Objekte föderierter Datenbanken (z. B. Wrapper, Server und Kurznamen) werden in den Systemkatalogtabellen in dieser Datenbankpartition gespeichert.

Wenn möglich, sollten Sie jede Datenbank in einer getrennten Instanz erstellen. Falls dies nicht möglich ist (d. h., Sie müssen mehr als eine Datenbank pro Instanz erstellen), sollten Sie die Katalogpartitionen auf die verfügbaren Datenbankpartitionen verteilen. Dadurch verringern sich Konkurrenzsituationen beim Abrufen von Katalogdaten in einer einzelnen Datenbankpartition.

Anmerkung: Sie sollten regelmäßig ein Backup der Katalogpartition durchführen und nach Möglichkeit vermeiden, Benutzerdaten in ihr zu speichern, da diese Daten die für das Backup benötigte Zeit verlängern.

Wenn Sie eine Datenbank erstellen, wird sie automatisch in allen Datenbankpartitionen erstellt, die in der Datei `db2nodes.cfg` definiert sind.

Wenn die erste Datenbank im System erstellt wird, wird ein Systemdatenbankverzeichnis gebildet. An dieses werden Informationen zu anderen Datenbanken, die Sie erstellen, angehängt. Wenn Sie mit UNIX arbeiten, heißt das Systemdatenbank-

verzeichnis sqldbdir und befindet sich im Verzeichnis sqllib unter Ihrem Ausgangsverzeichnis bzw. unter dem Verzeichnis, in dem die DB2-Datenbank installiert wurde. Dieses Verzeichnis muss sich unter UNIX in einem gemeinsam genutzten Dateisystem (z. B. NFS auf UNIX-Plattformen) befinden, weil es nur ein Systemdatenbankverzeichnis für alle Datenbankpartitionen gibt, die zu einer Umgebung mit partitionierten Datenbanken gehören. Unter Windows befindet sich das Systemdatenbankverzeichnis im Instanzverzeichnis.

Ebenfalls im Verzeichnis sqldbdir befindet sich die Systemintentionsdatei. Sie hat den Namen sqldbins und stellt sicher, dass die Datenbankpartitionen synchronisiert bleiben. Diese Datei muss ebenfalls in einem gemeinsam genutzten Dateisystem gespeichert sein, da es innerhalb aller Datenbankpartitionen nur ein Verzeichnis gibt. Die Datei wird von allen Datenbankpartitionen, die die Datenbank bilden, gemeinsam genutzt.

Zur Nutzung der Datenbankpartitionierung müssen Konfigurationsparameter geändert werden. Mithilfe der Befehle **GET DATABASE CONFIGURATION** und **GET DATABASE MANAGER CONFIGURATION** können Sie die Werte für einzelne Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers ermitteln. Zur Änderung einzelner Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers werden die Befehle **UPDATE DATABASE CONFIGURATION** bzw. **UPDATE DATABASE MANAGER CONFIGURATION** verwendet.

Zu den Konfigurationsparametern des Datenbankmanagers, die sich auf eine Umgebung mit partitionierten Datenbanken auswirken, gehören **conn_elapse**, **fcm_num_buffers**, **fcm_num_channels**, **max_connretries**, **max_coordagents**, **max_time_diff**, **num_poolagents** und **start_stop_time**.

Datenbankpartitionierung in mehrere Datenbankpartitionen

Der Datenbankmanager ermöglicht große Flexibilität bei der Verteilung von Daten über mehrere Datenbankpartitionen einer partitionierten Datenbank.

Die Benutzer können wählen, wie ihre Daten zu verteilen sind, indem sie Verteilungsschlüssel deklarieren, und sie können bestimmen, auf welche und auf wie viele Datenbankpartitionen ihre Tabellendaten verteilt werden können, indem Sie die Datenbankpartitionsgruppe und den Tabellenbereich auswählen, in denen die Daten gespeichert werden sollen.

Darüber hinaus gibt eine Verteilungszuordnung (die aktualisierbar ist) die Zuordnung zwischen Verteilungsschlüsselwerten und Datenbankpartitionen an. Dies ermöglicht eine flexible Parallelverarbeitung von Auslastungen in einer partitionierten Datenbank für große Tabellen, während gleichzeitig kleinere Tabellen in einer oder nur einer kleinen Anzahl von Datenbankpartitionen gespeichert werden können, wenn der Anwendungsentwickler dies so vorsieht. Jede lokale Datenbankpartition kann über lokale Indizes für die in ihr gespeicherten Daten verfügen, um einen lokalen Datenzugriff mit hoher Leistung bereitzustellen.

In einer partitionierten Datenbank dient der Verteilungsschlüssel zur Verteilung der Tabellendaten auf eine Gruppe von Datenbankpartitionen. Indexdaten werden ebenfalls zusammen mit den zugehörigen Tabellen partitioniert und lokal in den einzelnen Datenbankpartitionen gespeichert.

Bevor Datenbankpartitionen zum Speichern von Daten genutzt werden können, müssen Sie für den Datenbankmanager definiert werden. Die Datenbankpartitio-

nen werden in einer Datei mit dem Namen db2nodes.cfg definiert.
Der Verteilungsschlüssel für eine Tabelle in einem Tabellenbereich in einer Datenbankpartitionsgruppe einer partitionierten Datenbank wird in der Anweisung CREATE TABLE oder der Anweisung ALTER TABLE angegeben. Wenn kein Verteilungsschlüssel angegeben wird, wird standardmäßig ein Verteilungsschlüssel für die Tabelle aus der ersten Spalte des Primärschlüssels erstellt. Wenn kein Primärschlüssel definiert ist, wird als Standardverteilungsschlüssel die erste in der Tabelle definierte Spalte verwendet, die einen anderen Datentyp als einen LONG- oder LOB-Datentyp besitzt. Tabellen in partitionierten Datenbanken müssen mindestens eine Spalte enthalten, die weder ein LONG-Datentyp noch ein LOB-Datentyp ist. Eine Tabelle in einem Tabellenbereich, der sich in einer Datenbankpartitionsgruppe einer Einzelpartitionsdatenbank befindet, erhält nur dann einen Verteilungsschlüssel, wenn dieser explizit angegeben wird.

Zeilen werden in eine Datenbankpartition nach folgendem Verfahren eingefügt:
1. Ein Hashalgorithmus (d. h. eine Datenbankpartitionierungsfunktion) wird auf alle Spalten des Verteilungsschlüssels angewendet und generiert einen Indexwert für die Verteilungszuordnung.
2. Die Datenbankpartitionsnummer an diesem Indexwert in der Verteilungszuordnung gibt die Datenbankpartition an, in der die Zeile zu speichern ist.

Der Datenbankmanager unterstützt eine *Teilentclusterung*. Dies bedeutet, dass eine Tabelle auf eine Untermenge der Datenbankpartitionen im System (d. h. in einer Datenbankpartitionsgruppe) verteilt werden kann. Tabellen brauchen nicht auf alle Datenbankpartitionen im System verteilt zu werden.

Der Datenbankmanager besitzt die Fähigkeit zu erkennen, wenn sich Daten, auf die für einen Join oder eine Unterabfrage zugegriffen wird, in derselben Datenbankpartition in derselben Datenbankpartitionsgruppe befinden. Diese Situation wird als *Tabellenkollokation* bezeichnet. Zeilen in kollokierten (zusammengefassten) Tabellen mit denselben Verteilungsschlüsselwerten befinden sich in derselben Datenbankpartition. Der Datenbankmanager kann entscheiden, die Join- oder Unterabfrageverarbeitung in der Datenbankpartition auszuführen, in der die Daten gespeichert sind. Dies kann erhebliche Leistungsvorteile mit sich bringen.
Für kollokierte Tabellen gelten folgende Voraussetzungen:
- Sie müssen sich in derselben Datenbankpartitionsgruppe befinden, und zwar in einer, die nicht umverteilt wird. (Bei der Umverteilung verwenden Tabellen in der Datenbankpartitionsgruppe möglicherweise unterschiedliche Verteilungszuordnungen, sodass sie nicht kollokiert sind.)
- Sie müssen Verteilungsschlüssel mit derselben Anzahl von Spalten besitzen.
- Ihre entsprechenden Spalten des Verteilungsschlüssels müssen datenbankpartitionskompatibel sein.
- Sie müssen sich in einer Datenbankpartitionsgruppe einer Einzelpartitionsdatenbank in derselben Datenbankpartition befinden.

Authentifizierungsaspekte bei partitionierten Datenbanken

In einer partitionierten Datenbank muss jede Partition der Datenbank über dieselbe Menge definierter Benutzer und Gruppen verfügen. Wenn die Definitionen nicht übereinstimmen, kann der Benutzer in verschiedenen Partitionen zu verschiedenen Operationen berechtigt sein.

Konsistenz über alle Partitionen hinweg ist zu empfehlen.

Datenbankpartitionsgruppen

Eine Datenbankpartitionsgruppe ist ein benannter Satz, der aus einer oder mehreren Datenbankpartitionen besteht, die zu einer Datenbank gehören.

Eine Datenbankpartitionsgruppe, die mehr als eine Datenbankpartition enthält, wird als *Datenbankpartitionsgruppe mit mehreren Partitionen* bezeichnet. Datenbankpartitionsgruppen mit mehreren Partitionen können nur mit Datenbankpartitionen definiert werden, die zur selben Instanz gehören.

Abb. 1 zeigt ein Beispiel einer Datenbank mit fünf Datenbankpartitionen.

- Die Datenbankpartitionsgruppe 1 enthält alle Datenbankpartitionen, mit Ausnahme von einer Datenbankpartition.
- Die Datenbankpartitionsgruppe 2 enthält eine Datenbankpartition.
- Die Datenbankpartitionsgruppe 3 enthält zwei Datenbankpartitionen.
- Die Datenbankpartition innerhalb der Gruppe 2 wird gemeinsam mit der Gruppe 1 (und überlappend) benutzt.
- Eine einzelne Datenbankpartition in Gruppe 3 wird gemeinsam mit der Gruppe 1 (und überlappend) benutzt.

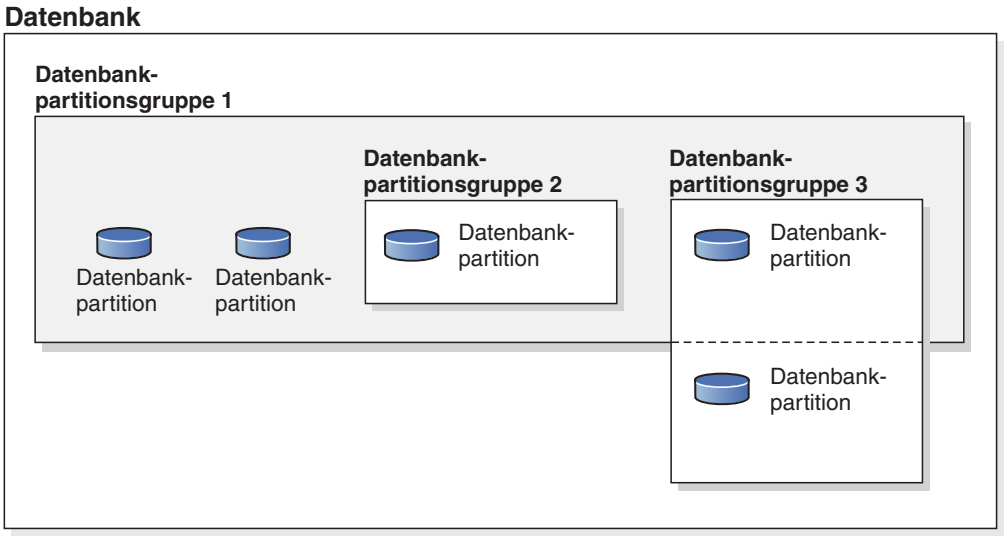


Abbildung 1. Datenbankpartitionsgruppen in einer Datenbank

Wird eine Datenbank erstellt, werden außerdem alle in der *Konfigurationsdatei der Datenbankpartition* mit der Bezeichnung `db2nodes.cfg` angegebenen Datenbankpartitionen erstellt. Andere Datenbankpartitionen können mit dem Befehl **`ADD DBPARTITIONNUM`** oder **`DROP DBPARTITIONNUM VERIFY`** hinzugefügt bzw. entfernt werden. Die Daten werden auf alle Datenbankpartitionen in einer Datenbankpartitionsgruppe verteilt.

Wenn eine Datenbankpartitionsgruppe erstellt wird, wird ihr eine *Verteilungszuordnung* zugeordnet. Die Verteilungszuordnung wird zusammen mit einem *Verteilungsschlüssel* und einem Hashalgorithmus vom Datenbankmanager verwendet, um festzustellen, welche Datenbankpartition in der Datenbankpartitionsgruppe eine bestimmte Zeile mit Daten speichert.

Standarddatenbankpartitionsgruppen

Bei der Datenbankerstellung werden automatisch drei Datenbankpartitionsgruppen definiert:

- IBMCATGROUP ist die Datenbankpartitionsgruppe für den Tabellenbereich SYSCATSPACE, in dem die Systemkatalogtabellen gespeichert werden.
- IBMTEMPGROUP ist die Datenbankpartitionsgruppe für den Tabellenbereich TEMPSPACE1, in dem temporäre Tabellen gespeichert werden, die bei Datenbankoperationen erstellt werden.
- IBMDEFAULTGROUP ist die Datenbankpartitionsgruppe für den Tabellenbereich USERSPACE1, der die Benutzertabellen und Indizes enthält. Ein Tabellenbereich für temporäre Benutzertabellen für eine deklarierte temporäre Tabelle oder eine erstellte temporäre Tabelle kann in IBMDEFAULTGROUP bzw. in einer beliebigen benutzererstellten Datenbankpartitionsgruppe, jedoch nicht in IBMTEMPGROUP erstellt werden.

Tabellenbereiche in Datenbankpartitionsgruppen

Wenn ein Tabellenbereich einer Datenbankpartitionsgruppe mit mehreren Partitionen zugeordnet wird (während der Ausführung der Anweisung CREATE TABLESPACE), werden alle Tabellen innerhalb dieses Tabellenbereichs auf alle Datenbankpartitionen in der Datenbankpartitionsgruppe partitioniert. Ein Tabellenbereich, der einer bestimmten Datenbankpartitionsgruppe zugeordnet wird, kann später keiner anderen Datenbankpartitionsgruppe zugeordnet werden.

Erstellen einer Datenbankpartitionsgruppe

Erstellen Sie eine Datenbankpartitionsgruppe mithilfe der Anweisung CREATE DATABASE PARTITION GROUP. Diese Anweisung gibt eine Gruppe von Datenbankpartitionen an, in der sich die Container eines Tabellenbereichs und die Tabellendaten befinden sollen. Durch diese Anweisung werden außerdem die folgenden Aktionen ausgeführt:

- Erstellen einer Verteilungszuordnung für die Datenbankpartitionsgruppe.
- Generieren einer ID für die Verteilungszuordnung.
- Einfügen von Datensätzen in die folgenden Katalogsichten:
  - SYSCAT.DBPARTITIONGROUPDEF
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS

Ändern einer Datenbankpartitionsgruppe

Mit der Anweisung ALTER DATABASE PARTITION GROUP können Sie Datenbankpartitionen zu einer Datenbankpartitionsgruppe hinzufügen (oder aus ihr löschen). Nach dem Hinzufügen oder Löschen von Datenbankpartitionen verwenden Sie den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP**, um die Daten auf die Menge an Datenbankpartitionen in der Datenbankpartitionsgruppe umzuverteilen.

Aspekte zum Aufbau von Datenbankpartitionsgruppen

Kleine Tabellen sollten in Datenbankpartitionsgruppen mit nur einer einzigen Partition angelegt werden, außer wenn Sie die Vorteile der Kollokation mit einer größeren Tabelle nutzen wollen. Unter *Kollokation* ist die Platzierung von Zeilen aus verschiedenen Tabellen, die zusammengehörige Daten enthalten, in die gleiche Datenbankpartition zu verstehen. Kollokierte (zusammengefasste) Tabellen unterstüt-

zen den Datenbankmanager bei der Verwendung effizienterer Joinstrategien. Solche Tabellen können sich in einer Datenbankpartitionsgruppe mit einer Einzelpartition befinden. Tabellen gelten als kollokiert, wenn sie sich in einer Datenbankpartitionsgruppe mit mehreren Partitionen befinden, dieselbe Anzahl von Spalten im Verteilungsschlüssel haben und die Datentypen der sich entsprechenden Spalten kompatibel sind. Zeilen in kollokierten Tabellen mit dem gleichen Wert im Verteilungsschlüssel werden in derselben Datenbankpartition gespeichert. Tabellen können sich in separaten Tabellenbereichen in derselben Datenbankpartitionsgruppe befinden und trotzdem als durch Kollokation zusammengefasst betrachtet werden.

Mittelgroße Tabellen sollten nicht über zu viele Datenbankpartitionen verteilt werden. Beispielsweise kann eine 100-MB-Tabelle in einer Datenbankpartitionsgruppe mit 16 Partitionen bessere Leistungswerte erzielen als in einer Datenbankpartitionsgruppe mit 32 Partitionen.

Sie können Datenbankpartitionsgruppen dazu verwenden, OLTP-Tabellen (OLTP - Online-Transaktionsverarbeitung) von Entscheidungshilfetabellen (DSS-Tabellen - Decision Support System) zu trennen. Dadurch können Sie sicherzustellen, dass die Leistung von OLTP-Transaktionen nicht beeinträchtigt wird.

Wenn Sie eine Datenbankpartitionsgruppe mit mehreren Partitionen verwenden, beachten Sie die folgenden Punkte:

- In einer Datenbankpartitionsgruppe mit mehreren Partitionen können Sie nur dann einen eindeutigen Index erstellen, wenn der Index eine Obermenge des Verteilungsschlüssels ist.
- Jeder Datenbankpartition muss eine eindeutige Nummer zugewiesen werden, da dieselbe Datenbankpartition in einer oder mehreren Datenbankpartitionsgruppen enthalten sein kann.
- Zur Gewährleistung einer schnellen Recovery einer Datenbankpartition mit den Systemkatalogtabellen sollten Sie keine Benutzertabellen in derselben Datenbankpartition anlegen. Platzieren Sie Benutzertabellen in Datenbankpartitionsgruppen, die nicht die Datenbankpartition der Datenbankpartitionsgruppe IBMCATGROUP enthalten.

Verteilungszuordnungen

In einer Umgebung mit partitionierten Datenbanken muss der Datenbankmanager wissen, wo sich die benötigten Dateien befinden. Zum Auffinden der Daten verwendet der Datenbankmanager eine so genannte *Verteilungszuordnung*.

Eine Verteilungszuordnung ist eine intern generierte Tabelle von entweder Einträgen für Datenbankpartitionsgruppen mit mehreren Partitionen oder einem einzigen Eintrag für Datenbankpartitionsgruppen mit Einzelpartitionen. Für eine Datenbankpartitionsgruppe mit nur einer Partition enthält die Verteilungszuordnung nur einen Eintrag mit der Nummer der Datenbankpartition, in der alle Zeilen einer Datenbanktabelle gespeichert werden. Für Datenbankpartitionsgruppen mit mehreren Partitionen werden die Nummern der Datenbankpartitionsgruppe so angegeben, dass die Datenbankpartitionen eine nach der anderen verwendet werden, um eine gleichmäßige Verteilung über die gesamte Zuordnung sicherzustellen. Vergleichbar mit der Verwendung einer Stadtkarte, die durch Gitterlinien in Sektoren unterteilt ist, verwendet der Datenbankmanager einen *Verteilungsschlüssel*, um die Speicherposition (die Datenbankpartition) zu bestimmen, in der die Daten gespeichert werden.

Angenommen, Sie haben eine Datenbank auf vier Datenbankpartitionen (mit den Nummern 0-3) verteilt. Die Verteilungszuordnung für die Datenbankpartitionsgruppe IBMDEFAULTGROUP dieser Datenbank sähe wie folgt aus:

```
0 1 2 3 0 1 2 ...
```

Wenn eine Datenbankpartitionsgruppe in der Datenbank mit den Datenbankpartitionen 1 und 2 erstellt würde, sähe die Verteilungszuordnung für diese Datenbankpartitionsgruppe wie folgt aus:

```
1 2 1 2 1 2 1 ...
```

Wenn der Verteilungsschlüssel für eine in die Datenbank zu ladende Tabelle eine ganze Zahl (Integer) mit möglichen Werten zwischen 1 und 500.000 ist, wird der Verteilungsschlüssel mit einem Hashverfahren auf eine Nummer zwischen 0 und 32.767 abgebildet. Diese Nummer wird als Index auf die Verteilungszuordnung verwendet, um die Datenbankpartition für die betreffende Zeile auszuwählen.

Abb. 2 zeigt, wie der Zeile mit dem Verteilungsschlüsselwert (c1, c2, c3) die Nummer 2 zugeordnet wird, die ihrerseits auf Datenbankpartition n5 verweist.

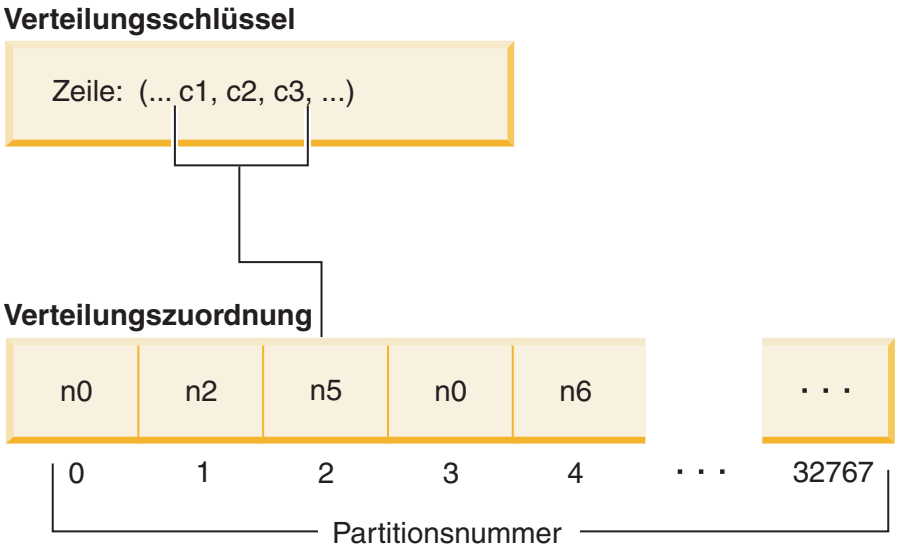


Abbildung 2. Datenverteilung mit einer Verteilungszuordnung

Eine solche Verteilungszuordnung ist eine flexible Steuermethode für die Speicherung von Daten in einer Mehrpartitionsdatenbank. Wenn die Notwendigkeit eintritt, die Datenverteilung auf die Datenbankpartitionen zu ändern, können Sie dazu das Dienstprogramm zur Datenumverteilung verwenden. Dieses Dienstprogramm ermöglicht Ihnen, die Datenverteilung neu auszugleichen oder bewusst ungleichmäßig zu gestalten.

Mithilfe der API db2GetDistMap können Sie eine Kopie der Verteilungszuordnung abrufen, die Sie prüfen können. Wenn Sie die API sqlugtpi weiterhin verwenden, um Verteilungsinformationen abzurufen, gibt die API möglicherweise eine Fehlernachricht SQL2768N zurück, da nur Verteilungszuordnungen abgerufen werden können, die 4096 Einträge enthalten.

Verteilungsschlüssel

Ein *Verteilungsschlüssel* ist eine Spalte (oder Spaltengruppe), die dazu verwendet wird, die Datenbankpartition zu bestimmen, in der eine bestimmte Datenzeile gespeichert wird.

Ein Verteilungsschlüssel wird für eine Tabelle mithilfe der Anweisung CREATE TABLE definiert. Wenn kein Verteilungsschlüssel für eine Tabelle in einem Tabellenbereich, der über mehr als eine Datenbankpartition in einer Datenbankpartitionsgruppe verteilt ist, definiert ist, wird standardmäßig ein Verteilungsschlüssel aus der ersten Spalte des Primärschlüssels erstellt.

Wenn kein Primärschlüssel angegeben ist, wird standardmäßig die erste für die Tabelle definierte Spalte, die keine langen Daten enthält, als Verteilungsschlüssel angenommen. (*Lang* bezieht sich hier auf alle langen Datentypen (LONG) und alle LOB-Datentypen). Wenn Sie eine Tabelle in einem Tabellenbereich erstellen, der zu einer Datenbankpartitionsgruppe mit nur einer Partition gehört, und einen Verteilungsschlüssel erstellen wollen, müssen Sie den Verteilungsschlüssel explizit definieren. In diesem Fall wird kein Standardschlüssel erstellt.

Wenn keine Spalte die Voraussetzungen für einen standardmäßig erstellten Verteilungsschlüssel erfüllt, wird die Tabelle ohne Verteilungsschlüssel erstellt. Tabellen ohne Verteilungsschlüssel sind nur in Datenbankpartitionsgruppen mit einer Partition zulässig. Sie können Verteilungsschlüssel auch später noch mit der Anweisung ALTER TABLE hinzufügen oder entfernen. Das Ändern des Verteilungsschlüssels ist nur bei einer Tabelle möglich, deren Tabellenbereich zu einer Datenbankpartitionsgruppe mit einer Einzelpartition gehört.

Die Auswahl eines guten Verteilungsschlüssels ist von großer Bedeutung. Berücksichtigen Sie die folgenden Aspekte:

- Wie der Zugriff auf Tabellen erfolgen soll
- Die Art der Abfrageauslastung
- Die vom Datenbanksystem angewendeten Joinstrategien

Wenn Kollokation kein Hauptgesichtspunkt ist, dann zeichnet sich ein guter Verteilungsschlüssel für eine Tabelle dadurch aus, dass er die Daten gleichmäßig über alle Datenbankpartitionen in der Datenbankpartitionsgruppe verteilt. Der Verteilungsschlüssel jeder Tabelle in einem Tabellenbereich, der einer Datenbankpartitionsgruppe zugeordnet ist, bestimmt, ob die Tabellen durch Kollokation zusammengefasst werden. Tabellen werden als zusammengefasst betrachtet, wenn folgende Bedingungen gelten:

- Die Tabellen liegen in Tabellenbereichen, die in derselben Datenbankpartitionsgruppe sind.
- Die Verteilungsschlüssel in jeder Tabelle haben dieselbe Anzahl von Spalten.
- Die Datentypen der entsprechenden Spalten sind partitionskompatibel.

Diese Merkmale stellen sicher, dass Zeilen zusammengefasster Tabellen mit denselben Werten für ihre Verteilungsschlüssel in derselben Datenbankpartition gespeichert werden.

Ein ungeeigneter Verteilungsschlüssel kann zu einer ungleichmäßigen Datenverteilung führen. Wählen Sie keine Spalten mit ungleichmäßig verteilten Daten und keine Spalten mit einer kleinen Anzahl unterschiedlicher Werte als Verteilungsschlüssel aus. Die Anzahl der unterschiedlichen Werte muss ausreichend groß sein, um eine gleichmäßige Verteilung der Zeilen auf alle Datenbankpartitionen in der Da-

tenbankpartitionsgruppe sicherzustellen. Der Aufwand für die Anwendung des Hashalgorithmus zur Verteilung ist proportional zur Größe des Verteilungsschlüssels. Der Verteilungsschlüssel kann nicht mehr als 16 Spalten enthalten. Jedoch wird bei weniger Spalten eine bessere Leistung erzielt. Nicht benötigte Spalten sollten nicht in den Verteilungsschlüssel aufgenommen werden.

Beim Definieren eines Verteilungsschlüssels sind folgende Faktoren zu beachten:

- Die Erstellung einer Mehrpartitionstabelle, die ausschließlich BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC, XML oder strukturierte Datentypen enthält, wird nicht unterstützt.
- Die Definition des Verteilungsschlüssels kann nicht geändert werden.
- Der Verteilungsschlüssel sollte die am häufigsten an Joins beteiligten Spalten enthalten.
- Der Verteilungsschlüssel sollte außerdem aus Spalten bestehen, die häufig an GROUP BY-Kauseln beteiligt sind.
- Jeder eindeutige Schlüssel oder Primärschlüssel muss alle Spalten des Verteilungsschlüssels enthalten.
- Stellen Sie in einer OLTP-Umgebung (Umgebung für Online-Transaktionsverarbeitung) sicher, dass alle Spalten des Verteilungsschlüssels an einer Transaktion durch Gleichheitsvergleichselemente beteiligt sind. Angenommen, Sie haben eine Spalte für Personalnummern (EMP_NO), die häufig in Transaktionen wie der folgenden verwendet werden:

  ```
  UPDATE emp_table SET ... WHERE
  emp_no = hostvariable
  ```

  In diesem Fall wäre die Spalte EMP_NO ein guter einspaltiger Verteilungsschlüssel für EMP_TABLE.

Als *Datenbankpartitionierung* wird die Methode bezeichnet, mit der die Speicherposition jeder Zeile in der Tabelle bestimmt wird. Diese Methode funktioniert folgendermaßen:

1. Ein Hashalgorithmus wird auf den Wert des Verteilungsschlüssels angewendet und generiert eine Nummer zwischen 0 und 32.767.
2. Die Verteilungszuordnung wird bei der Erstellung einer Datenbankpartitionsgruppe erstellt. Die Nummern werden immer wieder in derselben Reihenfolge nacheinander wiederholt, bis die Verteilungszuordnung gefüllt ist.
3. Die Nummer wird als Index für die Position in der Verteilungszuordnung verwendet. Die Nummer an dieser Position in der Verteilungszuordnung ist die Nummer der Datenbankpartition, in der die Zeile gespeichert wird.

Tabellenkollokation

Wenn mindestens zwei Tabellen häufig Daten als Antwort auf bestimmte Abfragen liefern, kann es sinnvoll sein, zusammengehörige Daten aus diesen Tabellen möglichst nah beieinander zu speichern. In einer Umgebung mit partitionierten Datenbanken wird dieser Prozess als *Tabellenkollokation* bezeichnet.

Tabellen sind in einer Kollokation zusammengefasst, wenn sie in derselben Datenbankpartitionsgruppe gespeichert sind und ihre Verteilungsschlüssel kompatibel sind. Durch Speichern beider Tabellen in derselben Datenbankpartitionsgruppe wird sichergestellt, dass sie eine gemeinsame Verteilungszuordnung haben. Die Tabellen können sich in verschiedenen Tabellenbereichen befinden, jedoch müssen die Tabellenbereiche derselben Datenbankpartitionsgruppe zugeordnet sein. Die Datentypen der entsprechenden Spalten in den jeweiligen Verteilungsschlüsseln müssen *partitionskompatibel* sein.

Beim Zugriff auf mehr als eine Tabelle bei einem Join oder einer Unterabfrage kann der Datenbankmanager feststellen, ob sich die zu verknüpfenden Daten in derselben Datenbankpartition befinden. Wenn dies der Fall ist, kann die Joinverarbeitung oder die Unterabfrage in der Datenbankpartition ausgeführt werden, in der die Daten gespeichert sind, und die Daten müssen nicht zwischen Datenbankpartitionen verschoben werden. Diese Möglichkeit bietet beträchtliche Leistungsvorteile.

Partitionskompatibilität

Die Basisdatentypen entsprechender Spalten von Verteilungsschlüsseln werden verglichen und können als *partitionskompatibel* deklariert werden. Partitionskompatible Datentypen haben die Eigenschaft, dass ein bestimmter Partitionierungsalgorithmus zwei Variablen mit jeweils einem dieser Datentypen dieselbe Nummer zuordnet, wenn sie denselben Wert haben.

Eine Partitionskompatibilität liegt vor, wenn die folgenden Merkmale gelten:

- Ein Wert des Basisdatentyps ist mit einem anderen desselben Basistyps kompatibel.
- Interne Formate werden für die Datentypen DATE (Datum), TIME (Uhrzeit) und TIMESTAMP (Zeitmarke) verwendet. Sie sind untereinander nicht kompatibel, und keiner dieser Typen ist mit Zeichen- oder Grafikdatentypen kompatibel.
- Die Partitionskompatibilität wird nicht von der Optionalität der Dateneingabe für eine Spalte beeinflusst.
- Die Partitionskompatibilität wird von der Sortierfolge beeinflusst. Die localeabhängigen, UCA-basierten Sortierfolgen benötigen eine exakte Übereinstimmung in der Sortierfolge, das Attribut 'Strength (S)' der Sortierfolge wird hierbei ignoriert. Alle anderen Sortierfolgen werden im Hinblick auf das Feststellen der Partitionskompatibilität als identisch betrachtet.
- Zeichenspalten, die mit FOR BIT DATA definiert wurden, sind nur dann mit Zeichenspalten ohne FOR BIT DATA kompatibel, wenn eine andere Sortierfolge als eine localeabhängige, UCA-basierte Sortierfolge verwendet wird.
- Nullwerte (NULL) kompatibler Datentypen werden auf identische Weise behandelt, Nullwerte nicht kompatibler Datentypen möglicherweise nicht.
- Die Basisdatentypen eines benutzerdefinierten Datentyps werden zur Analyse der Partitionskompatibilität verwendet.
- Dezimalzahlen desselben Werts im Verteilungsschlüssel werden gleich behandelt, auch wenn ihre Anzahl an Kommastellen und ihre Genauigkeit unterschiedlich sind.
- Folgende Leerzeichen in Zeichenfolgen (CHAR, VARCHAR, GRAPHIC oder VARGRAPHIC) werden vom Hashalgorithmus ignoriert.
- BIGINT, SMALLINT und INTEGER sind kompatible Datentypen.
- Wird eine localeabhängige, UCA-basierte Sortierfolge verwendet, sind CHAR, VARCHAR, GRAPHIC und VARGRAPHIC kompatible Datentypen. Wird eine andere Sortierfolge verwendet, sind CHAR und VARCHAR mit unterschiedlichen Längen kompatible Typen und GRAPHIC und VARGRAPHIC sind kompatible Typen, aber CHAR und VARCHAR sind keine kompatiblen Typen für GRAPHIC und VARGRAPHIC.
- Die Partitionskompatibilität gilt nicht für die Datentypen LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB und BLOB, da sie in Verteilungsschlüsseln nicht unterstützt werden.

Partitionierte Tabellen

Partitionierte Tabellen arbeiten mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als *Datenpartitionen* oder *Datenbereiche* bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden.

Eine Datenpartition oder Datenbereich (Range) ist ein Teil einer Tabelle, der eine Teilmenge der Zeilen einer Tabelle enthält und separat von anderen Gruppen von Zeilen gespeichert wird. Daten aus einer gegebenen Tabelle werden auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen in mehrere Datenpartitionen oder Bereiche partitioniert. Diese Datenpartitionen oder Datenbereiche können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden. Wenn eine Tabelle unter Verwendung der Klausel PARTITION BY erstellt wird, wird sie partitioniert.

Alle angegebenen Tabellenbereiche müssen dieselbe Seitengröße und Bereichsgröße, denselben Speichermechanismus (DMS oder SMS) und -typ (REGULAR oder LARGE) aufweisen, und sie müssen sich alle in derselben Datenbankpartitionsgruppe befinden.

Eine partitionierte Tabelle vereinfacht das Rollin und Rollout von Tabellendaten (d. h. die Ein- und Auslagerung ganzer Datenbereiche). Zudem kann eine partitionierte Tabelle wesentlich mehr Daten enthalten als eine gewöhnliche Tabelle. Sie können eine partitionierte Tabelle mit maximal 32.767 Datenpartitionen erstellen. Datenpartitionen können einer partitionierten Tabelle hinzugefügt (ADD PARTITION) und zugeordnet (ATTACH PARTITION) werden und die Zuordnung von Datenpartitionen kann aufgehoben werden (DETACH PARTITION). Darüber hinaus können mehrere Datenpartitionsbereiche einer Tabelle in einem einzigen Tabellenbereich gespeichert werden.

Indizes für eine partitionierte Tabelle können partitioniert oder nicht partitioniert sein. Eine partitionierte Tabelle kann gleichzeitig partitionierte und nicht partitionierte Indizes haben.

Einschränkungen

Partitionierte hierarchische oder temporäre Tabellen, Bereichsclustertabellen und partitionierte Sichten werden zur Verwendung in partitionierten Tabellen nicht unterstützt.

Tabellenpartitionierung

Die Tabellenpartitionierung ist ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als *Datenpartitionen* bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle verteilt werden. Jede Datenpartition wird separat gespeichert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Speicherobjekte verhalten sich ähnlich wie einzelne Tabellen. Daher lassen sich schnelle Dateneinlagerungen (Rollins) problemlos ausführen, indem eine vorhandene Tabelle mithilfe der Anweisung ALTER TABLE ...ATTACH in eine partitionierte Tabelle integriert wird. Analog lässt sich eine Datenauslagerung (Rollout) problemlos mithilfe der Anweisung ALTER TABLE ...DETACH durchführen. Auch die Ab-

frageverarbeitung kann von der Trennung von Daten profitieren, da sie dadurch in der Lage ist, ein Durchsuchen irrelevanter Daten zu vermeiden. Dies führt bei vielen Abfragen, wie sie für Data-Warehouses verwendet werden, zu einer besseren Abfrageleistung.

Tabellendaten werden gemäß den Angaben in der Klausel PARTITION BY der Anweisung CREATE TABLE partitioniert. Die Spalten, die in dieser Definition verwendet werden, bilden den so genannten Tabellenpartitionierungsschlüssel.

Dieses Organisationsschema kann isoliert oder in Kombination mit anderen Organisationsschemata verwendet werden. Durch die Kombination der Klauseln DISTRIBUTE BY und PARTITION BY der Anweisung CREATE TABLE können Daten über Datenbankpartitionen verteilt werden, die sich über mehrere Tabellenbereiche erstrecken. Die folgenden Organisationsschemata sind verfügbar:

- DISTRIBUTE BY HASH
- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

Die Tabellenpartitionierung wird mit DB2 Version 9.1 Enterprise Server Edition für Linux, UNIX und Windows (und späteren Versionen) zur Verfügung gestellt.

Vorteile der Tabellenpartitionierung

Wenn eine der folgenden Bedingungen auf Sie und Ihr Unternehmen zutrifft, sollten Sie die zahlreichen Vorteile der Tabellenpartitionierung in Betracht ziehen:

- Sie haben ein Data-Warehouse, das von einfacheren Rollin- und Rolloutoperationen für Tabellendaten profitieren würde.
- Sie haben ein Data-Warehouse, das sehr große Tabellen enthält.
- Sie erwägen eine Migration von einem vorherigen Release oder von einem konkurrierenden Datenbankprodukt auf eine Datenbank der Version 9.1.
- Sie wünschen eine effektivere Nutzung von HSM-Lösungen (HSM = Hierarchical Storage Management).

Die Tabellenpartitionierung ermöglicht eine einfachere Ein- und Auslagerung (Rollin und Rollout) von Tabellendaten, eine einfachere Verwaltung, eine flexible Positionierung von Indizes sowie eine bessere Abfrageverarbeitung.

Effizientes Rollin und Rollout von Daten

Die Partitionierung von Tabellen ermöglicht eine effiziente Ein- und Auslagerung von Tabellendaten (engl. Rollin und Rollout). Diese Effizienz können Sie durch die Klauseln ATTACH PARTITION und DETACH PARTITION in der Anweisung ALTER TABLE erreichen. Durch ein Rollin partitionierter Tabellendaten lässt sich ein neuer Datenbereich auf einfache Weise in eine partitionierte Tabelle in Form einer zusätzlichen Datenpartition integrieren. Durch ein Rollout partitionierter Tabellendaten lassen sich Datenbereiche auf einfache Weise zur nachfolgenden Löschung oder Archivierung von einer partitionierten Tabelle abtrennen.

Wenn bei DB2 Version 9.7 Fixpack 1 und späteren Releases die Zuordnung einer Datenpartition zu einer partitionierten Tabelle unter Verwendung der Anweisung ALTER TABLE mit der Klausel DETACH PARTITION aufgehoben wird, können dynamische Abfragen, die mit der Isolationsstufe 'Lesestabilität', 'Cursorstabilität' oder 'Nicht festgeschriebener Lesevorgang' ausgeführt werden, weiterhin auf die partitionierte Quellentabelle zugreifen. Ähnlich können bei der Zuordnung einer Datenpartition zu einer partitio-

nierten Tabelle unter Verwendung der Anweisung ALTER TABLE mit der Klausel ATTACH PARTITION dynamische Abfragen, die mit der Isolationsstufe 'Lesestabilität', 'Cursorstabilität' oder 'Nicht festgeschriebener Lesevorgang' ausgeführt werden, weiterhin auf die partitionierte Zieltabelle zugreifen.

Einfachere Verwaltung großer Tabellen
Die Verwaltung auf Tabellenebene ist flexibler, weil Sie Verwaltungstasks an einzelnen Datenpartitionen ausführen können. Solche Verwaltungstasks sind zum Beispiel das Aufheben der Zuordnung und das erneute Zuordnen einer Datenpartition, das Durchführen von Backup- und Restoreoperationen für einzelne Datenpartitionen sowie das Reorganisieren einzelner Indizes. Zeitaufwändige Wartungsoperationen können abgekürzt werden, indem sie in eine Reihe kleinerer Operationen zerlegt werden. Zum Beispiel können Backupoperationen eine Datenpartition nach der anderen abarbeiten, wenn sich die Datenpartitionen in separaten Tabellenbereichen befinden. Auf diese Weise ist es möglich, das Backup für jeweils nur eine Datenpartition einer partitionierten Tabelle durchzuführen.

Flexible Platzierung von Indizes
Indizes können jetzt in anderen Tabellenbereichen gespeichert werden, sodass eine differenziertere Steuerung der Indexplatzierung möglich ist. Dies bietet zum Beispiel folgende Vorteile:

- Verbesserte Leistung beim Löschen von Indizes und während der Onlineindexerstellung.
- Es besteht die Möglichkeit, verschiedene Werte für die Tabellenbereichsmerkmale für jeden Index 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.

Sowohl DMS- als auch SMS-Tabellenbereiche unterstützen die Verwendung von Indizes, die sich an einer anderen Position befinden als die Tabelle.

Verbesserte Leistung für Abfragen, wie sie für Business-Intelligence-Anwendungen verwendet werden
Die Abfrageverarbeitung wurde erweitert, sodass Datenpartitionen auf der Basis von Vergleichselementen der Abfrage automatisch ausgeschlossen werden. Diese Form der Abfrageverarbeitung wird als *Ausschluss von Datenpartitionen* (Data Partition Elemination) bezeichnet und wirkt sich vorteilhaft auf viele Entscheidungsunterstützungsabfragen aus.

Im folgenden Beispiel wird eine Tabelle CUSTOMER erstellt, bei der Zeilen mit l_shipdate >= '01/01/2006' und l_shipdate <= '03/31/2006' im Tabellenbereich TS1, Zeilen mit l_shipdate >= '04/01/2006' und l_shipdate <= '06/30/2006' im Tabellenbereich TS2 usw. gespeichert werden.

```
CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30))
IN ts1, ts2, ts3, ts4, ts5
PARTITION BY RANGE(l_shipdate) (STARTING FROM ('01/01/2006')
ENDING AT ('12/31/2006') EVERY (3 MONTHS))
```

Datenpartitionen und Datenbereiche

Partitionierte Tabellen arbeiten mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als *Datenpartitionen* bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden. Die für die einzelnen Datenpartitionen angegebenen Bereiche können bei der Erstellung einer Tabelle automatisch generiert oder manuell angegeben werden.

Auf Datenpartitionen wird innerhalb der DB2-Bibliothek auf verschiedene Weise Bezug genommen. Die folgende Liste gibt die am häufigsten verwendeten Verweise an:

- DATAPARTITIONNAME ist der permanente Name, der einer Datenpartition für eine bestimmte Tabelle bei der Erstellung zugeordnet wird. Dieser Spaltenwert wird in der Katalogsicht SYSCAT.DATAPARTITIONS gespeichert. Dieser Name bleibt bei einer ATTACH- oder DETACH-Operation nicht erhalten.
- DATAPARTITIONID ist die permanente ID, die einer Datenpartition für eine bestimmte Tabelle bei der Erstellung zugeordnet wird. Sie dient zur eindeutigen Kennzeichnung einer bestimmten Datenpartition in einer gegebenen Tabelle. Diese ID bleibt bei einer ATTACH- oder DETACH-Operation nicht erhalten. Dieser Wert wird vom System generiert und kann in den Ausgaben verschiedener Dienstprogramme angezeigt werden.
- SEQNO gibt die Reihenfolge eines bestimmten Datenpartitionsbereichs in Bezug auf andere Datenpartitionsbereiche in der Tabelle an, wobei Datenpartitionen, deren Zuordnung aufgehoben wurde, bei einer Sortierung hinter allen sichtbaren und zugeordneten Datenpartitionen angeordnet werden.

Datenorganisationsschemata

Mit der Einführung der Tabellenpartitionierung bietet eine DB2-Datenbank ein drei Ebenen umfassendes Datenorganisationsschema. Es gibt drei Klauseln der Anweisung CREATE TABLE, die einen Algorithmus enthalten, der angibt, wie die Daten zu organisieren sind.

Die folgenden drei Klauseln zeigen die Ebenen der Datenorganisation, die in beliebiger Kombination verwendet werden können:

- Die Klausel DISTRIBUTE BY dient zur gleichmäßigen Verteilung der Daten auf Datenbankpartitionen, um die abfrageinterne Parallelität zu aktivieren und zum Lastausgleich zwischen den Datenbankpartitionen (Datenbankpartitionierung)
- Die Klausel PARTITION BY dient zur Gruppierung von Zeilen mit ähnlichen Werten einer einzigen Dimension in der gleichen Datenpartition (Tabellenpartitionierung).
- Die Klausel ORGANIZE BY dient zur Gruppierung von Zeilen mit ähnlichen Werten in mehreren Dimensionen im gleichen Tabellenspeicherbereich (mehrdimensionales Clustering) oder zur Gruppierung von Zeilen auf der Basis der Uhrzeit der Einfügeoperation (ITC = Insert Time Clustering; Clustering anhand der Einfügungszeit).

Diese Syntax ermöglicht Konsistenz zwischen den Klauseln und lässt zukünftige Algorithmen der Datenorganisation zu. Jede dieser Klauseln kann isoliert oder in Kombination mit den anderen verwendet werden. Durch die Kombination der Klauseln DISTRIBUTE BY und PARTITION BY der Anweisung CREATE TABLE können Daten über Datenbankpartitionen verteilt werden, die sich über mehrere

Tabellenbereiche erstrecken. Diese Lösung ermöglicht ein ähnliches Verhalten wie die Hybridfunktionalität von Informix Dynamic Server und Informix Extended Parallel Server.

Sie können die Klauseln in einer Tabelle kombinieren, um durch die verschiedenen Datenorganisationsschemata ausgereiftere Partitionierungsschemata zu erstellen. Zum Beispiel sind Umgebungen mit partitionierten Datenbanken nicht nur kompatibel mit der Tabellenpartitionierung, sondern auch eine geeignete Ergänzung.

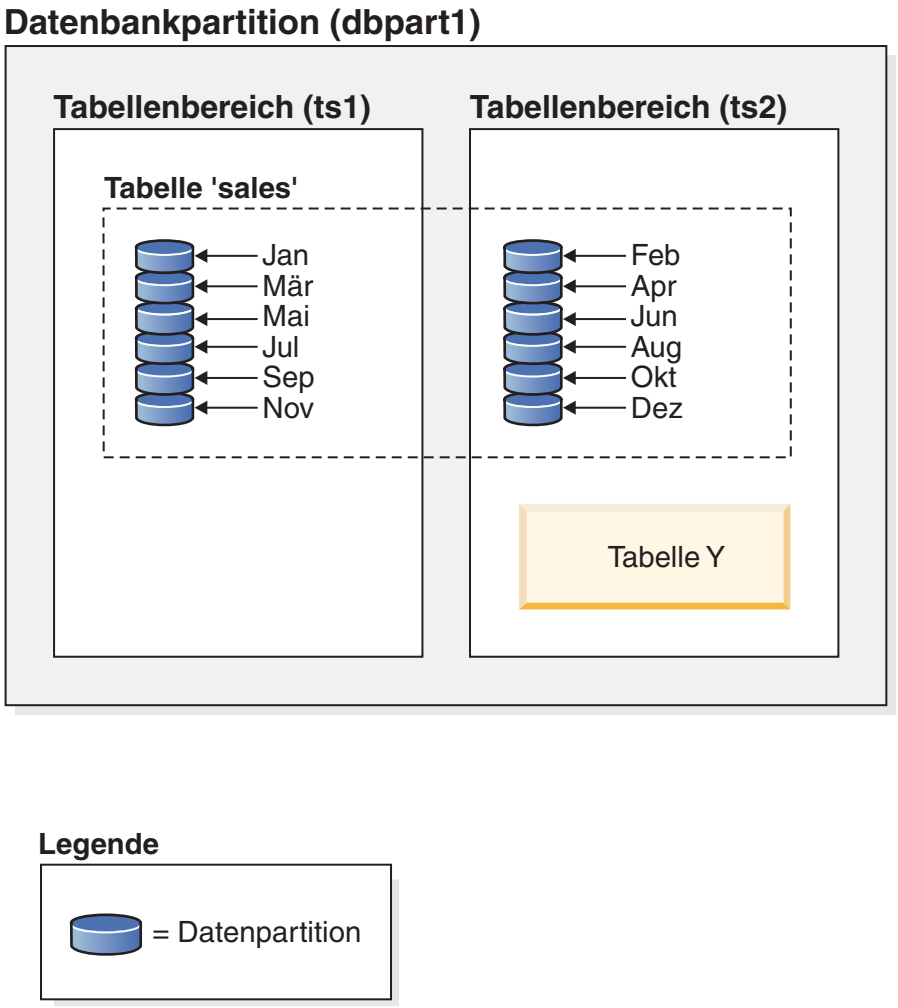


Abbildung 3. Veranschaulichung des Organisationsschemas der Tabellenpartitionierung, bei dem eine Tabelle mit monatlichen Vertriebsdaten in mehrere Datenpartitionen unterteilt ist. Die Tabelle erstreckt sich außerdem über zwei Tabellenbereiche ('ts1' und 'ts2').

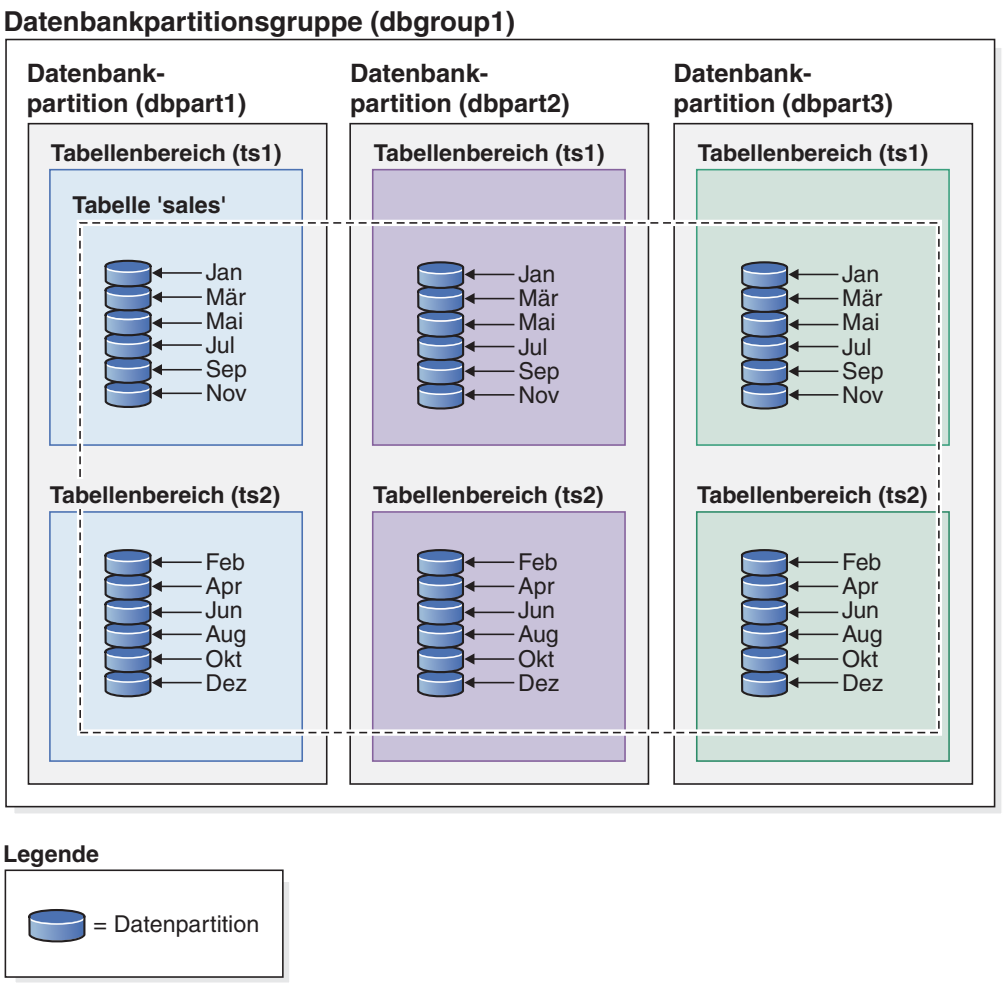


Abbildung 4. Veranschaulichung der ergänzenden Organisationsschemata der Datenbankpartitionierung und Tabellenpartitionierung. Eine Tabelle mit monatlichen Vertriebsdaten ist in mehrere Datenpartitionen partitioniert, die sich über zwei Tabellenbereiche ('ts1' und 'ts2') erstrecken, die wiederum über mehrere Datenbankpartitionen ('dbpart1', 'dbpart2', 'dbpart3') einer Datenbankpartitionsgruppe ('dbgroup1') verteilt sind.

Der auffälligste Unterschied zwischen dem Mehrdimensionalen Clustering (MDC) und der Tabellenpartitionierung besteht im Vorhandensein mehrerer Dimensionen im Gegensatz zu einer einzigen Dimension. MDC eignet sich für Datenwürfel (d. h. Tabellen mit mehreren Dimensionen) und die Tabellenpartitionierung bietet sich an, wenn eine einzige Dimension vorhanden ist, die für den Aufbau der Datenbank von zentraler Bedeutung ist (z. B. eine Datumsspalte). MDC und Tabellenpartitionierung können sich ergänzen, wenn beide dieser Bedingungen erfüllt sind. Dies wird in Abb. 5 auf Seite 19 veranschaulicht.

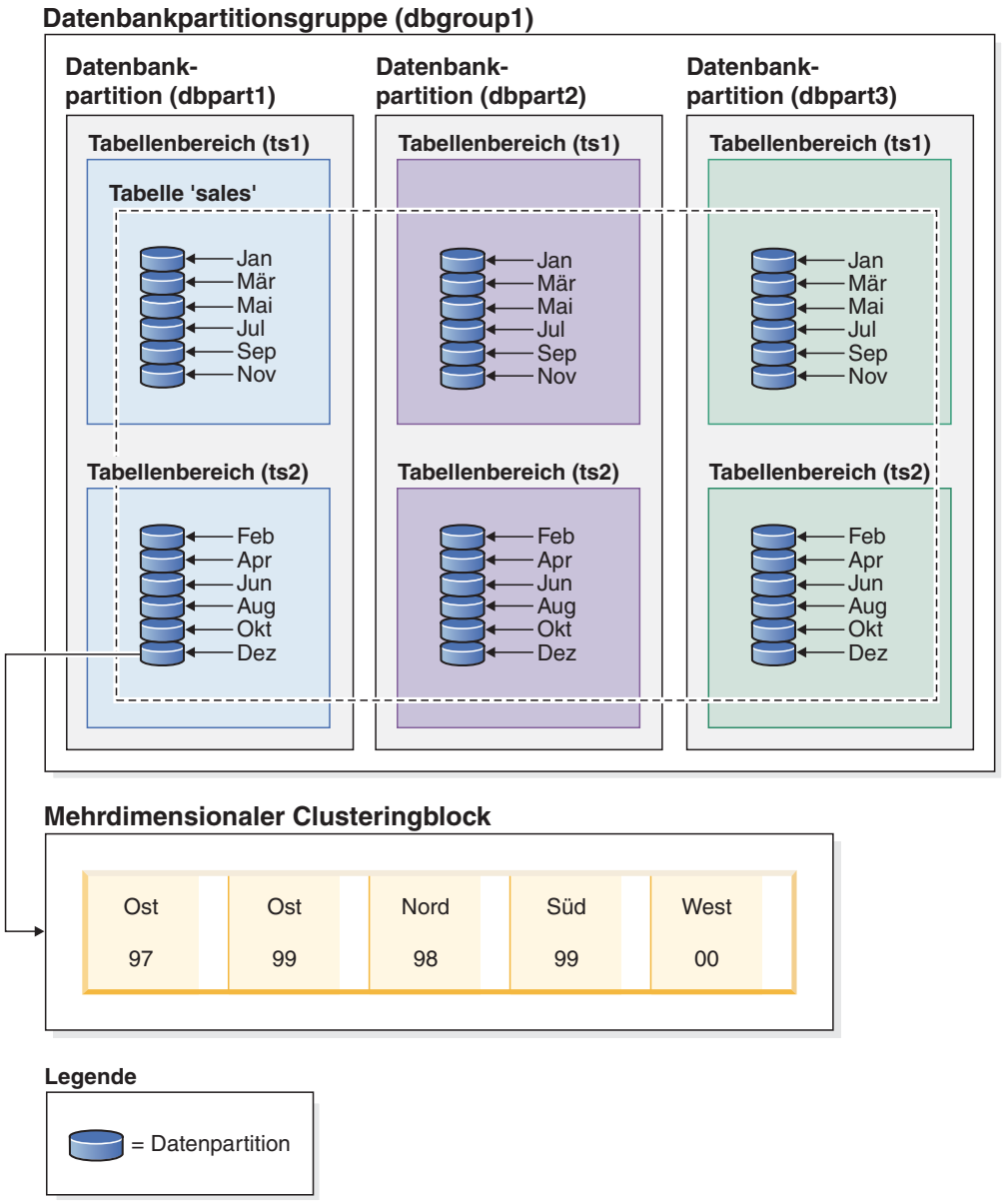


Abbildung 5. Eine Darstellung der Organisationsschemata der Datenbankpartitionierung, der Tabellenpartitionierung und des Mehrdimensionalen Clustering, in denen Daten aus der Tabelle 'sales' nicht nur über mehrere Datenbankpartitionen verteilt und über die Tabellenbereiche 'ts1' und 'ts2' partitioniert sind, sondern auch Zeilen mit ähnlichen Werten in den Dimensionen für Datum und Region gruppiert werden.

Es gibt ein weiteres Datenorganisationsschema, das nicht in Verbindung mit einem anderen der gezeigten Schemata verwendet werden kann. Dieses Schema wird durch die Klausel ORGANIZE BY KEY SEQUENCE angeben. Es dient zur Einfügung der einzelnen Datensätze jeweils in eine Zeile, die bei der Erstellung der Tabelle für diesen Datensatz reserviert wurde (Bereichsclustertabelle).

Terminologie der Datenorganisation

Datenbankpartitionierung

Ein Datenorganisationsschema, bei dem Tabellendaten über mehrere Datenbankpartitionen auf der Basis der Hashwerte in einer oder mehreren Verteilungsschlüsselspalten der Tabelle sowie mithilfe einer Verteilungszuordnung der Datenbankpartitionierung verteilt werden. Daten aus einer

gegebenen Tabelle werden gemäß den Spezifikationen verteilt, die in der Klausel DISTRIBUTE BY HASH der Anweisung CREATE TABLE angegeben wurden.

Datenbankpartition
Ein Teil einer Datenbank auf einem Datenbankpartitionsserver, der eigene Benutzerdaten und Indizes, eine Konfigurationsdatei und Transaktionsprotokolle enthält. Datenbankpartitionen können logisch oder physisch sein.

Tabellenpartitionierung
Ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Datenpartitionen entsprechend den Werten einer oder mehrerer Partitionierungsspalten der Tabelle 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 befinden.

Datenpartition
Eine Gruppe von Tabellenzeilen, die separat von anderen Gruppen von Zeilen gespeichert und durch die Spezifikationen zu Gruppen zusammengefasst werden, die in der Klausel PARTITION BY RANGE der Anweisung CREATE TABLE angegeben wurden.

MDC-Tabelle (MDC - Mehrdimensionales Clustering)
Eine Tabelle, deren Daten physisch in Blöcken in einer oder mehreren Dimensionen bzw. Clusteringschlüsseln organisiert werden, die in der Klausel ORGANIZE BY DIMENSIONS angegeben werden.

Clustering anhand der Einfügungszeit (Insert Time Clustering, ITC)
Eine Tabelle, deren Daten auf der Basis der Einfügezeit für Zeilen physisch in Gruppen zusammengefasst werden, die von der Klausel ORGANIZE BY INSERT TIME angegeben werden.

Vorteile der einzelnen Datenorganisationsschemata

Die Kenntnis der Vorteile der einzelnen Datenorganisationsschemata kann Ihnen helfen, bei der Planung, beim Entwurf oder bei der erneuten Beurteilung der Anforderungen Ihres Datenbanksystems die beste Strategie zu wählen. Tabelle 2 bietet eine Übersicht über häufige Anforderungen von Kunden und zeigt, wie die verschiedenen Datenorganisationsschemata bei der Erfüllung dieser Anforderungen behilflich sein können.

Tabelle 2. Verwendung der Tabellenpartitionierung mit dem Database Partitioning Feature (DPF)

Problemstellung	Empfohlenes Schema	Erläuterung
Datenrollout	Tabellenpartitionierung	Nutzt die Aufhebung der Zuordnung (DETACH PARTITION) zur Durchführung eines Rollouts großer Datenmengen unter minimaler Beeinträchtigung.
Parallele Abfrageausführung (Abfrageleistung)	Database Partitioning Feature	Ermöglicht Abfrageparallelität zur Realisierung einer besseren Abfrageleistung.

Tabelle 2. Verwendung der Tabellenpartitionierung mit dem Database Partitioning Feature (DPF) (Forts.)

Problemstellung	Empfohlenes Schema	Erläuterung
Ausschluss von Datenpartitionen (Abfrageleistung)	Tabellenpartitionierung	Ermöglicht den Ausschluss von Datenpartitionen zur Realisierung einer besseren Abfrageleistung.
Maximierung der Abfrageleistung	Beide	Maximale Abfrageleistung bei kombinierter Verwendung: Abfrageparallelität und Ausschluss von Datenpartitionen ergänzen sich.
Hohe Belastung von Administratoren	Database Partitioning Feature	Ausführung zahlreicher Tasks für die einzelnen Datenbankpartitionen.

Tabelle 3. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Erläuterung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Beide	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.

Anmerkung: Die Tabellenpartitionierung wird jetzt vor UNION ALL-Sichten empfohlen.

Datenorganisationsschemata in DB2- und Informix-Datenbanken

Die Tabellenpartitionierung ist ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als *Datenpartitionen* bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle verteilt werden. Jede Datenpartition wird separat gespeichert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Tabellendaten werden gemäß den Angaben in der Klausel PARTITION BY der Anweisung CREATE TABLE partitioniert. Die Spalten, die in dieser Definition verwendet werden, bilden den so genannten Tabellenpartitionierungsschlüssel. Die DB2-Tabellenpartitionierung lässt sich mit der Datenfragmentierungsstrategie für die Datenorganisation vergleichen, die von Informix Dynamic Server und Informix Extended Parallel Server angeboten wird.

Das Informix-Konzept

Informix unterstützt eine Reihe von Datenorganisationsschemata, die in den Informix-Produkten als *Fragmentierung* bezeichnet werden. Einer der häufiger genutzten Typen von Fragmentierung ist FRAGMENT BY EXPRESSION. Dieser Fragmentierungstyp funktioniert ähnlich einer CASE-Anweisung, wobei jedem Fragment der Tabelle ein Ausdruck zugeordnet ist. Diese Ausdrücke werden ausgewertet, um zu bestimmen, wo die Zeile abzulegen ist.

Vergleich zwischen dem Informix- und dem DB2-Datenbanksystem

Das DB2-Datenbanksystem stellt ein reich ausgestattetes Sortiment an ergänzenden Funktionen bereit, die sich direkt den Datenorganisationsschemata von Informix zuordnen lassen und es Kunden relativ leicht machen, sich von der Informix-Syntax auf die DB2-Syntax umzustellen. Der DB2-Datenbankmanager realisiert komplizierte Informix-Schemata durch eine Kombination aus generierten Spalten und der Klausel PARTITION BY RANGE der Anweisung CREATE TABLE. Tabelle 4 stellt die von Informix- und DB2-Datenbankprodukten verwendeten Datenorganisationsschemata einander gegenüber.

Tabelle 4. Übersicht über die Informix-Datenorganisationsschemata und ihre DB2-Entsprechungen

Datenorganisationsschema	Informix-Syntax	Syntax in DB2 Version 9.1
• Informix: ausdrucksbasiert • DB2: Tabellenpartitionierung	FRAGMENT BY EXPRESSION	PARTITION BY RANGE
• Informix: Umlaufmodus • DB2: Standard	FRAGMENT BY ROUND ROBIN	Keine Syntax: Der DB2-Datenbankmanager verteilt Daten automatisch auf Container.
• Informix: Bereichsverteilung • DB2: Tabellenpartitionierung	FRAGMENT BY RANGE	PARTITION BY RANGE
• Informix: systemdefiniertes Hashverfahren • DB2: Datenbankpartitionierung	FRAGMENT BY HASH	DISTRIBUTE BY HASH
• Informix: HYBRID • DB2: Datenbankpartitionierung mit Tabellenpartitionierung	FRAGMENT BY HYBRID	DISTRIBUTE BY HASH, PARTITION BY RANGE
• Informix: n/v • DB2: Mehrdimensionales Clustering (MDC)	n/v	ORGANIZE BY DIMENSIONS

Beispiele

Die folgenden Beispiele zeigen Details dazu, wie äquivalente DB2-Ergebnisse für die einzelnen FRAGMENT BY EXPRESSION-Schemata von Informix realisiert werden.

Beispiel 1: Die folgende einfache Anweisung CREATE TABLE zeigt die Informix-Fragmentierung und die äquivalente Tabellenpartitionierungssyntax für ein DB2-Datenbanksystem:

Informix-Syntax:

```
CREATE TABLE demo(a INT) FRAGMENT BY EXPRESSION
 a = 1 IN db1,
 a = 2 IN db2,
 a = 3 IN db3;
```

DB2-Syntax:

```
CREATE TABLE demo(a INT) PARTITION BY RANGE(a)
 (STARTING(1) IN db1,
 STARTING(2) IN db2,
 STARTING(3) ENDING(3) IN db3);
```

Informix XPS unterstützt ein zweistufiges Fragmentierungsschema, das als Hybridschema bezeichnet wird. Bei diesem Schema werden Daten über Co-Server mithilfe eines Ausdrucks und innerhalb des Co-Servers mithilfe eines zweiten Ausdrucks verteilt. Dadurch können alle Co-Server an einer Abfrage aktiv arbeiten (d. h. auf allen Co-Servern befinden sich Daten), und die Abfrage kann den Ausschluss von Datenpartitionen vorteilhaft nutzen.

Das DB2-Datenbanksystem realisiert ein zum Informix-Hybridschema äquivalentes Organisationsschema durch eine Kombination aus den Klauseln DISTRIBUTE BY und PARTITION BY der Anweisung CREATE TABLE.

Beispiel 2: Das folgende Beispiel zeigt die Syntax für die kombinierten Klauseln:

Informix-Syntax

```
CREATE TABLE demo(a INT, b INT) FRAGMENT BY HYBRID HASH(a)
 EXPRESSION b = 1 IN dbsl1,
   b = 2 IN dbsl2;
```

DB2-Syntax

```
CREATE TABLE demo(a INT, b INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1);
```

Darüber hinaus können Sie MDC-Tabellen verwenden, um eine zusätzliche Ebene der Datenorganisation zu gewinnen:

```
CREATE TABLE demo(a INT, b INT, c INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1)
 ORGANIZE BY DIMENSIONS(c);
```

In diesem Beispiel werden alle Zeilen mit dem gleichen Wert in der Spalte **a** in derselben Datenbankpartition abgelegt. Alle Zeilen mit dem gleichen Wert in Spalte **b** werden im selben Tabellenbereich gespeichert. Für einen gegebenen Wert in Spalte **a** und Spalte **b** werden alle Zeilen mit gleichen Wert in Spalte **c** im selben Clusteringbereich auf der Platte zusammengefasst. Diese Lösung eignet sich ideal für OLAP-Operationen, die detailliertere Informationen abrufen, weil nur ein oder mehrere Speicherbereiche (Extents) bzw. Blöcke in einem einzigen Tabellenbereich in einer einzigen Datenbankpartition durchsucht werden müssen, um eine Abfrage dieses Typs zu erfüllen.

Anwendung der Tabellenpartitionierung auf häufig anzutreffende Anwendungsprobleme

In den folgenden Abschnitten wird die Anwendung der verschiedenen Funktionsmerkmale der DB2-Tabellenpartitionierung auf Anwendungsprobleme erläutert, die häufiger vorkommen. In jedem Abschnitt werden dabei bewährte Verfahrensweisen zur Umsetzung verschiedener Fragmentierungsschemata von Informix in äquivalente DB2-Tabellenpartitionierungsschemata behandelt.

Zu beachtende Aspekte für die Erstellung einfacher Datenpartitionsbereiche

Eine der gängigsten Anwendungen der Tabellenpartitionierung ist die Partitionierung einer großen Fakttabelle nach einem Datumsschlüssel. Wenn Sie gleichmäßig große Bereiche von Daten erzeugen müssen, ziehen Sie die Verwendung einer Form der CREATE TABLE-Syntax zur automatischen Generierung in Betracht.

Beispiele

Beispiel 1: Das folgende Beispiel zeigt die Form der Syntax zur automatischen Generierung:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber  INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice  DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag  CHAR(1),
 l_linestatus  CHAR(1),
 l_shipdate  DATE,
 l_commitdate  DATE,
 l_receiptdate  DATE,
 l_shipinstruct  CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44))
  PARTITION BY RANGE(l_shipdate)
  (STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH);
```

Durch diese Anweisung werden 24 Bereiche, d. h. einer für jeden Monat im Zeitraum 1992-1993, erstellt. Ein Versuch, eine Zeile mit einem Wert für l_shipdate einzufügen, der außerhalb dieses Bereichs liegt, führt zu einem Fehler.

Beispiel 2: Vergleichen Sie das vorausgehende Beispiel mit der folgenden Informix-Syntax:

```
create table orders
(
 l_orderkey  decimal(10,0) not null,
 l_partkey  integer,
 l_suppkey  integer,
 l_linenumber integer,
 l_quantity  decimal(12,2),
 l_extendedprice decimal(12,2),
 l_discount  decimal(12,2),
 l_tax   decimal(12,2),
 l_returnflag char(1),
 l_linestatus char(1),
 l_shipdate  date,
```

```
 l_commitdate date,
 l_receiptdate date,
 l_shipinstruct char(25),
 l_shipmode  char(10),
 l_comment  varchar(44)
) fragment by expression
l_shipdate < '1992-02-01' in ldbs1,
l_shipdate >= '1992-02-01' and l_shipdate < '1992-03-01' in ldbs2,
l_shipdate >= '1992-03-01' and l_shipdate < '1992-04-01' in ldbs3,
l_shipdate >= '1992-04-01' and l_shipdate < '1992-05-01' in ldbs4,
l_shipdate >= '1992-05-01' and l_shipdate < '1992-06-01' in ldbs5,
l_shipdate >= '1992-06-01' and l_shipdate < '1992-07-01' in ldbs6,
l_shipdate >= '1992-07-01' and l_shipdate < '1992-08-01' in ldbs7,
l_shipdate >= '1992-08-01' and l_shipdate < '1992-09-01' in ldbs8,
l_shipdate >= '1992-09-01' and l_shipdate < '1992-10-01' in ldbs9,
l_shipdate >= '1992-10-01' and l_shipdate < '1992-11-01' in ldbs10,
l_shipdate >= '1992-11-01' and l_shipdate < '1992-12-01' in ldbs11,
l_shipdate >= '1992-12-01' and l_shipdate < '1993-01-01' in ldbs12,
l_shipdate >= '1993-01-01' and l_shipdate < '1993-02-01' in ldbs13,
l_shipdate >= '1993-02-01' and l_shipdate < '1993-03-01' in ldbs14,
l_shipdate >= '1993-03-01' and l_shipdate < '1993-04-01' in ldbs15,
l_shipdate >= '1993-04-01' and l_shipdate < '1993-05-01' in ldbs16,
l_shipdate >= '1993-05-01' and l_shipdate < '1993-06-01' in ldbs17,
l_shipdate >= '1993-06-01' and l_shipdate < '1993-07-01' in ldbs18,
l_shipdate >= '1993-07-01' and l_shipdate < '1993-08-01' in ldbs19,
l_shipdate >= '1993-08-01' and l_shipdate < '1993-09-01' in ldbs20,
l_shipdate >= '1993-09-01' and l_shipdate < '1993-10-01' in ldbs21,
l_shipdate >= '1993-10-01' and l_shipdate < '1993-11-01' in ldbs22,
l_shipdate >= '1993-11-01' and l_shipdate < '1993-12-01' in ldbs23,
l_shipdate >= '1993-12-01' and l_shipdate < '1994-01-01' in ldbs24,
l_shipdate >= '1994-01-01' in ldbs25;
```

Beachten Sie, dass die Informix-Syntax einen Bereich mit offenem Anfang und offenem Ende vorsieht, um Datumseingaben aufzufangen, die nicht innerhalb des erwarteten Bereichs liegen. Durch das Hinzufügen von Bereichen mit MINVALUE und MAXVALUE lässt sich die DB2-Syntax der Informix-Syntax entsprechend anpassen.

Beispiel 3: Im folgenden Beispiel wird Beispiel 1 zur Angleichung an die Informix-Syntax abgewandelt:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag CHAR(1),
 l_linestatus CHAR(1),
 l_shipdate  DATE,
 l_commitdate DATE,
 l_receiptdate DATE,
 l_shipinstruct CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44)
) PARTITION BY RANGE(l_shipdate)
 (STARTING MINVALUE,
  STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH,
  ENDING MAXVALUE);
```

Durch dieses Verfahren kann nun jedes beliebige Datum in die Tabelle eingefügt werden.

Partitionierung durch Ausdruck mit generierten Spalten

Obwohl das DB2-Datenbanksystem keine direkte Unterstützung für eine Partitionierung durch einen Ausdruck bietet, wird die Partitionierung über eine generierte Spalte unterstützt, sodass sich dasselbe Ergebnis erzielen lässt.

Beachten Sie die folgenden Verwendungshinweise, bevor Sie entscheiden, ob diese Lösung in Betracht kommt:

- Die generierte Spalte ist eine reale Spalte, die physischen Plattenspeicher belegt. Tabellen, die eine generierte Spalte enthalten, können daher etwas größer sein.
- Eine Änderung des Ausdrucks für die generierte Spalte, über die eine partitionierte Tabelle partitioniert wird, wird nicht unterstützt. Jeder Versuch, dies zu tun, führt zu einer Nachricht SQL0190. Das Hinzufügen einer neuen Datenpartition zu einer Tabelle, die generierte Spalten in der im folgenden Abschnitt beschriebenen Weise verwendet, erfordert in der Regel eine Änderung des Ausdrucks, der die generierte Spalte definiert. Das Ändern des Ausdrucks, der eine generierte Spalte definiert, wird gegenwärtig nicht unterstützt.
- Die Möglichkeiten zur Anwendung des Ausschlusses von Datenpartitionen unterliegen Einschränkungen, wenn eine Tabelle generierte Spalten verwendet.

Beispiele

Beispiel 1: Im folgenden Beispiel wird die Informix-Syntax anstellen verwendet, an denen sich die Verwendung generierter Spalten anbietet. In diesem Beispiel enthält die Spalte, über die die Partitionierung erfolgt, kanadische Provinzen und Territorien. Da es unwahrscheinlich ist, dass sich die Liste der Provinzen ändert, ist es auch unwahrscheinlich, dass sich der Ausdruck für die generierte Spalte ändert.

```
CREATE TABLE customer (
 cust_id  INT,
 cust_prov CHAR(2))
 FRAGMENT BY EXPRESSION
  cust_prov = "AB" IN dbspace_ab
  cust_prov = "BC" IN dbspace_bc
  cust_prov = "MB" IN dbspace_mb
  ...
  cust_prov = "YT" IN dbspace_yt
  REMAINDER IN dbspace_remainder;
```

Beispiel 2: In diesem Beispiel wird die DB2-Tabelle über eine generierte Spalte partitioniert:

```
CREATE TABLE customer (
 cust_id     INT,
 cust_prov   CHAR(2),
 cust_prov_gen GENERATED ALWAYS AS (CASE
  WHEN cust_prov = 'AB' THEN 1
  WHEN cust_prov = 'BC' THEN 2
  WHEN cust_prov = 'MB' THEN 3
  ...
  WHEN cust_prov = 'YT' THEN 13
  ELSE 14 END))
 IN tbspace_ab, tbspace_bc, tbspace_mb, .... tbspace_remainder
 PARTITION BY RANGE (cust_prov_gen)
  (STARTING 1 ENDING 14 EVERY 1);
```

Hier entsprechen die Ausdrücke in der CASE-Anweisung den zuvor gezeigten Ausdrücken in der Klausel FRAGMENT BY EXPRESSION. Die CASE-Anweisung ordnet jedem ursprünglichen Ausdruck eine Nummer zu, die in der generierten Spalte (hier: cust_prov_gen) gespeichert wird. Diese Spalte ist eine reale Spalte, die auf der Platte gespeichert wird. Daher kann die Tabelle in dieser Form geringfügig mehr Speicher beanspruchen als wenn DB2 die Partitionierung durch einen Ausdruck direkt unterstützen würde. In diesem Beispiel wird die Kurzform der Syntax verwendet. Daher müssen die Tabellenbereiche, in denen die Datenpartitionen gespeichert werden sollen, in der IN-Klausel der Anweisung CREATE TABLE aufgelistet werden. Bei der Langform der Syntax ist für jede Datenpartition eine separate IN-Klausel erforderlich.

Anmerkung: Dieses Verfahren lässt sich auf jede beliebige Klausel FRAGMENT BY EXPRESSION anwenden.

Tabellenpartitionierungsschlüssel

Ein Tabellenpartitionierungsschlüssel ist eine geordnete Gruppe aus mindestens einer Spalte in einer Tabelle. Mithilfe der Werte in den Spalten des Tabellenpartitionierungsschlüssels wird bestimmt, zu welcher Datenpartition die einzelnen Tabellenzeilen gehören.

Zur Definition des Tabellenpartitionierungsschlüssels für eine Tabelle verwenden Sie die Anweisung CREATE TABLE mit der Klausel PARTITION BY.

Die Auswahl einer geeigneten Spalte für den Tabellenpartitionierungsschlüssel spielt eine wesentliche Rolle, wenn Sie die Vorteile der Tabellenpartitionierung in vollem Umfang ausschöpfen möchten. Die folgenden Richtlinien können Ihnen bei der Auswahl der am besten geeigneten Spalten für den Partitionierungsschlüssel Ihrer partitionierten Tabelle helfen.

- Definieren Sie die Bereichsunterteilung, die dem Datenrollout entsprechen soll. Dazu werden sehr häufig Wochen, Monate oder Quartale verwendet.
- Definieren Sie Bereiche, die der Rollin-Größe von Daten entsprechen. Daten werden zum Beispiel häufig über eine Datums- oder Zeitspalte partitioniert.
- Partitionieren Sie über eine Spalte, die Vorteile beim Ausschluss von Partitionen bietet.

Unterstützte Datentypen

In Tabelle 5 werden die Datentypen (einschließlich Synonymen) aufgeführt, die zur Verwendung in einer Spalte eines Tabellenpartitionierungsschlüssels unterstützt werden:

Tabelle 5. Unterstützte Datentypen

Datentypspalte 1	Datentypspalte 2
SMALLINT	INTEGER
INT	BIGINT
FLOAT	REAL
DOUBLE	DECIMAL
DEC	DECFLOAT
NUMERIC	NUM
CHARACTER	CHAR
VARCHAR	DATE

Tabelle 5. Unterstützte Datentypen (Forts.)

Datentypspalte 1	Datentypspalte 2
TIME	GRAPHIC
VARGRAPHIC	CHARACTER VARYING
TIMESTAMP	CHAR VARYING
CHARACTER FOR BIT DATA	CHAR FOR BIT DATA
VARCHAR FOR BIT DATA	CHARACTER VARYING FOR BIT DATA
CHAR VARYING FOR BIT DATA	Benutzerdefinierte Datentypen (einzigartige Datentypen)

Nicht unterstützte Datentypen

Die folgenden Datentypen können in einer partitionierten Tabelle enthalten sein, werden jedoch nicht zur Verwendung in einer Spalte des Tabellenpartitionierungsschlüssels unterstützt:

- Benutzerdefinierte Datentypen (strukturierte Datentypen)
- LONG VARCHAR
- LONG VARCHAR FOR BIT DATA
- BLOB
- BINARY LARGE OBJECT
- CLOB
- CHARACTER LARGE OBJECT
- DBCLOB
- LONG VARGRAPHIC
- REF
- Zeichenfolge mit variabler Länge für C
- Zeichenfolge mit variabler Länge für Pascal
- XML

Wenn Sie eine automatische Generierung von Datenpartitionen mithilfe der Klausel EVERY der Anweisung CREATE TABLE anfordern, kann nur eine Spalte als Tabellenpartitionierungsschlüssel verwendet werden. Wenn Sie Datenpartitionen manuell generieren, indem Sie die einzelnen Bereiche in der Klausel PARTITION BY der Anweisung CREATE TABLE angeben, können mehrere Spalten als Tabellenpartitionierungsschlüssel verwendet werden, wie im folgenden Beispiel gezeigt:

```
CREATE TABLE sales (year INT, month INT)
   PARTITION BY RANGE(year, month)
   (STARTING FROM (2001, 1) ENDING (2001,3) IN tbsp1,
   ENDING (2001,6) IN tbsp2, ENDING (2001,9)
   IN tbsp3, ENDING (2001,12) IN tbsp4,
   ENDING (2002,3) IN tbsp5, ENDING (2002,6)
   IN tbsp6, ENDING (2002,9) IN tbsp7,
   ENDING (2002,12) IN tbsp8)
```

Mit dieser Anweisung werden acht Datenpartitionen erstellt, d. h. eine für jedes Quartal der Jahre 2001 und 2002.

Anmerkung:

1. Wenn mehrere Spalten als Tabellenpartitionierungsschlüssel verwendet werden, werden sie insofern wie ein zusammengesetzter Schlüssel behandelt (ähnlich

den zusammengesetzten Schlüsseln in einem Index), als dass nachfolgende Spalten von den führenden Spalten abhängig sind. Jeder Anfangs- oder Endwert (aller Spalten zusammen) muss in höchstens 512 Zeichen angegeben werden. Diese Begrenzung entspricht der Größe der Spalten LOWVALUE und HIGHVALUE der Katalogsicht SYSCAT.DATAPARTITIONS. Ein Anfangs- oder Endwert, der mit mehr als 512 Zeichen angegeben wird, führt zu einem Fehler SQL0636N mit Ursachencode 9.

2. Die Tabellenpartitionierung erstreckt sich über mehrere Spalten, nicht Dimensionen. Bei der Tabellenpartitionierung gehören alle Spalten nur zu einer Dimension.

Generierte Spalten

Generierte Spalten können als Tabellenpartitionierungsschlüssel verwendet werden. Im folgenden Beispiel wird eine Tabelle mit zwölf Datenpartitionen, d. h. eine für jeden Monat, erstellt. Alle Zeilen für Januar eines beliebigen Jahres werden in der ersten Datenpartition, die Zeilen für Februar in der zweiten usw. abgelegt.

Beispiel 1

```
CREATE TABLE monthly_sales (sales_date date,
 sales_month int GENERATED ALWAYS AS (month(sales_date)))
     PARTITION BY RANGE (sales_month)
     (STARTING FROM 1 ENDING AT 12 EVERY 1);
```

Anmerkung:

1. Sie können den Ausdruck einer generierten Spalte, die im Tabellenpartitionierungsschlüssel verwendet wird, nicht ändern oder löschen. Auch das Hinzufügen eines Ausdrucks zur Generierung einer Spalte in einer Spalte, die im Tabellenpartitionierungsschlüssel verwendet wird, ist nicht zulässig. Der Versuch, einen Ausdruck zur Generierung einer Spalte in einer Spalte, die im Tabellenpartitionierungsschlüssel verwendet wird, hinzuzufügen, zu löschen oder zu ändern führt zu einem Fehler (SQL0270N RC=52).
2. Der Ausschluss von Datenpartitionen wird für Bereichsvergleichselemente nicht verwendet, wenn die generierte Spalte nicht monoton ist oder das Optimierungsprogramm nicht erkennen kann, dass sie monoton ist. Wenn nicht monotone Ausdrücke vorhanden sind, kann der Ausschluss von Datenpartitionen nur für Gleichheitsvergleichselemente oder IN-Vergleichselemente ausgeführt werden. Eine detaillierte Beschreibung und Beispiele zur Monotonie finden Sie in „Aspekte der Erstellung von MDC- oder ITC-Tabellen" auf Seite 56.

Ladeaspekte für partitionierte Tabellen

Alle vorhandenen Ladefunktionen werden unterstützt, wenn die Zieltabelle partitioniert ist. Hierbei gelten jedoch die folgenden allgemeinen Einschränkungen:

- Konsistenzpunkte werden nicht unterstützt, wenn die Anzahl der Partitionierungsagenten größer als 1 ist.
- Das Laden von Daten in eine Untergruppe von Datenpartitionen, während die verbleibenden Datenpartitionen vollständig online bleiben, wird nicht unterstützt.
- Die von einer Ladeoperation verwendete Ausnahmetabelle kann nicht partitioniert sein.
- Es kann keine Ausnahmetabelle angegeben werden, wenn die Zieltabelle eine XML-Spalte enthält.

- Ein eindeutiger Index kann nicht erneut erstellt werden, wenn das Dienstprogramm LOAD im Einfügemodus (INSERT) oder Neustartmodus (RESTART) ausgeführt wird und freigegebene Objekte von der Zieltabelle der Ladeoperation abhängen.
- Ähnlich wie beim Laden von MDC-Tabellen bleibt die genaue Reihenfolge der Eingabedatensätze beim Laden von partitionierten Tabellen nicht erhalten. Sortierungen werden lediglich innerhalb der Zelle oder Datenpartition beibehalten.
- Ladeoperationen, die mehrere Formatierungsprogramme auf jeder Datenbankpartition verwenden, erhalten lediglich eine ungefähre Reihenfolge der Eingabedatensätze. Beim Ausführen eines einzelnen Formatierungsprogramms auf jeder Datenbankpartition werden die Eingabedatensätze nach Zellen- oder Tabellenpartitionierungsschlüssel gruppiert. Um auf jeder Datenbankpartition ein einziges Formatierungsprogramm auszuführen, muss für `CPU_PARALLELISM` explizit der Wert 1 angefordert werden.

Allgemeines Verhalten des Dienstprogramms LOAD

Das Dienstprogramm LOAD fügt Datensätze in die richtige Datenpartition ein. Es ist nicht erforderlich, ein externes Dienstprogramm (wie beispielsweise einen Verteilerprozess) zu verwenden, um die Eingabedaten vor dem Laden zu partitionieren.

Das Dienstprogramm LOAD greift nicht auf freigegebene oder zugeordnete Datenpartitionen zu. Daten werden lediglich in sichtbare Datenpartitionen eingefügt. Sichtbare Datenpartitionen sind weder zugeordnet noch freigegeben. Darüber hinaus werden freigegebene oder zugeordnete Datenpartitionen von LOAD REPLACE-Operationen nicht abgeschnitten. Da das Dienstprogramm LOAD Sperren für die Systemkatalogtabellen anfordert, wartet das Dienstprogramm LOAD auf alle ALTER TABLE-Transaktionen, für die noch kein Commit durchgeführt wurde. Solche Transaktionen fordern eine exklusive Sperre für die relevanten Zeilen in den Katalogtabellen an, und die exklusive Sperre muss beendet werden, bevor die Ladeoperation fortgesetzt werden kann. Dies bedeutet, dass nicht festgeschriebene Transaktionen ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION nicht zulässig sind, während die Ladeoperation ausgeführt wird. Alle Eingabequellensätze, die für eine zugeordnete (ATTACHED) oder freigegebene (DETACHED) Datenpartition bestimmt sind, werden zurückgewiesen und können aus der Ausnahmetabelle abgerufen werden, sofern eine angegeben wurde. Eine Informationsnachricht wird in die Nachrichtendatei geschrieben, um anzugeben, dass einige Datenpartitionen der Zieltabelle im Status zugeordnet (ATTACHED) oder freigegeben (DETACHED) waren. Sperren für die relevanten Zeilen der Katalogtabelle, die der Zieltabelle entsprechen, verhindern, dass Benutzer die Partitionierung der Zieltabelle durch Absetzen einer Operation ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION während der Ausführung des Dienstprogramms LOAD ändern.

Verarbeitung ungültiger Zeilen

Wenn das Dienstprogramm LOAD auf einen Datensatz trifft, der zu keiner der sichtbaren Datenpartitionen gehört, wird der betreffende Datensatz zurückgewiesen, und das Dienstprogramm LOAD setzt die Verarbeitung fort. Die Anzahl der Datensätze, die aufgrund einer Verletzung der Bereichsvorgabe zurückgewiesen wurden, wird nicht explizit angegeben, ist aber in der Gesamtanzahl der zurückgewiesenen Datensätze enthalten. Durch das Zurückweisen eines Datensatzes aufgrund einer Verletzung der Bereichsvorgabe wird die Anzahl der Warnungen für Zeilen nicht erhöht. Es wird

eine einzige Nachricht (SQL0327N) in die Nachrichtendatei des Dienstprogramms LOAD geschrieben, die angibt, dass Bereichsverletzungen festgestellt wurden. Es wird jedoch nicht für jeden Datensatz eine eigene Nachricht protokolliert. Zusätzlich zu allen Spalten der Zieltabelle enthält die Ausnahmetabelle auch Spalten, die den Typ des Verstoßes beschreibt, der für eine bestimmte Zeile aufgetreten ist. Zeilen, die ungültige Daten enthalten (einschließlich Daten, die nicht partitioniert werden können), werden in die Speicherauszugsdatei geschrieben.

Da Einfügungen (INSERT-Operationen) in die Ausnahmetabelle ressourcenintensiv sind, können Sie steuern, welche Verstöße gegen Integritätsbedingungen in die Ausnahmetabelle eingefügt werden sollen. Das Standardverhalten des Dienstprogramms LOAD ist beispielsweise, Zeilen in die Ausnahmetabelle einzufügen, die aufgrund einer Verletzung der Bereichsvorgabe oder der eindeutigen Integritätsbedingung zurückgewiesen wurden, ansonsten jedoch gültig sind. Sie können dieses Verhalten inaktivieren, indem Sie jeweils NORANGEEXC oder NOUNIQUEEXC mit der Ausnahmebedingung FOR EXCEPTION angeben. Wenn Sie angeben, dass diese Verletzungen der Bereichsvorgabe nicht in die Ausnahmetabelle eingefügt werden sollen, oder wenn Sie keine Ausnahmetabelle angeben, gehen alle Informationen zu Zeilen, die eine Bereichsvorgabe oder eine eindeutige Integritätsbedingung verletzen, verloren.

Protokolldatei

Bei partitionierten Zieltabellen enthält der entsprechende Eintrag in der Protokolldatei keine Liste der Tabellenbereiche, die die Zieltabelle umfasst. Eine andere Kennung für die Granularität von Operationen ('R' anstelle von 'T') gibt an, dass eine Ladeoperation für eine partitionierte Tabelle ausgeführt wurde.

Beenden einer Ladeoperation

Beim Beenden einer LOAD REPLACE-Operation werden alle sichtbaren Datenpartitionen vollständig abgeschnitten; beim Beenden einer LOAD INSERT-Operation werden alle sichtbaren Datenpartitionen auf ihre jeweilige Länge vor der Ladeoperation abgeschnitten. Beim Beenden einer ALLOW READ ACCESS-Ladeoperation, die in der LOAD COPY-Phase fehlschlug, werden die entsprechenden Indizes ungültig gemacht. Ein Index wird ebenfalls ungültig gemacht, wenn eine Ladeoperation mit der Option ALLOW NO ACCESS beendet wird, die den Index geändert hat. (Der Index wird ungültig gemacht, weil der Indexierungsmodus REBUILD (erneut erstellen) ist oder weil ein Schlüssel während der inkrementellen Verwaltung eingefügt wurde und den Index in einem inkonsistenten Status belassen hat). Das Laden von Daten in mehrere Ziele hat keinen Einfluss auf Recoveryoperationen, mit der Ausnahme, dass die Ladeoperation von einem Konsistenzpunkt, der während der LOAD-Phase erstellt wurde, nicht erneut gestartet werden kann. In diesem Fall wird die LOAD-Option `SAVECOUNT` ignoriert, wenn die Zieltabelle partitioniert ist. Dieses Verhalten entspricht dem Verhalten beim Laden von Daten in einer MDC-Zieltabelle.

Generierte Spalten

Befindet sich eine generierte Spalte in einem der Partitionierungs-, Dimensions- oder Verteilungsschlüssel, wird der Änderungswert `generatedoverride` für den Dateityp ignoriert, und das Dienstprogramm LOAD generiert Werte, als wäre der Änderungswert `generatedignore` für den Dateityp angegeben. Das Laden eines falschen Wertes einer generierten Spalte kann in diesem Fall dazu führen, dass der Datensatz in die falsche

physische Position (wie beispielsweise in die falsche Datenpartition, den falschen MDC-Block oder die falsche Datenbankpartition) gestellt wird. Beispiel: Befindet sich ein Datensatz in einer falschen Datenpartition, muss die Operation SET INTEGRITY zum Festlegen der Integrität den Satz in eine andere physische Position versetzen, was im Verlauf von SET INTEGRITY-Operationen, die online ausgeführt werden, nicht bewerkstelligt werden kann.

Datenverfügbarkeit

Der aktuelle Algorithmus für ALLOW READ ACCESS-Ladeoperationen erstreckt sich auf partitionierte Tabellen. Eine ALLOW READ ACCESS-Ladeoperation ermöglicht den gleichzeitigen Lesezugriff auf die gesamte Tabelle. Dies schließt sowohl Ladedatenpartitionen als auch Nicht-Ladedatenpartitionen ein.

Wichtig: Ab Version 10.1 Fixpack 1 gilt der Parameter ALLOW READ ACCESS als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Weitere Informationen hierzu finden Sie im Abschnitt „Parameter ALLOW READ ACCESS im Befehl LOAD gilt als veraltet" in .

Das Dienstprogramm INGEST unterstützt auch partitionierte Tabellen und ist besser geeignet, den gleichzeitigen Zugriff auf Daten und die Verfügbarkeit der Daten zu ermöglichen, als der Befehl LOAD mit dem Parameter ALLOW READ ACCESS. Es kann zum Versetzen großer Datenvolumen aus Dateien und Pipes ohne Sperren der Zieltabelle verwendet werden. Darüber hinaus werden hierbei die Daten sofort nach dem Commit auf der Basis der verstrichenen Zeit bzw. der Anzahl der Zeilen zugänglich.

Datenpartitionsstatus

Nach einer erfolgreichen Ladeoperation können sichtbare Datenpartitionen entweder in den Tabellenstatus Set Integrity Pending oder Read Access Only oder beide wechseln. Dies hängt von bestimmten Bedingungen ab. So können Datenpartitionen in diese Status versetzt werden, wenn für die Tabelle Integritätsbedingungen bestehen, die von der Ladeoperation nicht verwaltet werden können. Dabei kann es sich z. B. um Integritätsbedingungen handeln, die Prüfungen auf Integritätsbedingungen und freigegebene MQTs (Materialized Query Table) beinhalten. Schlägt eine Ladeoperation fehl, verbleiben alle sichtbaren Datenpartitionen im Tabellenstatus Load Pending ('Laden anstehend').

Fehlerisolation

Eine Fehlerisolation auf Datenpartitionsebene wird nicht unterstützt. Das Isolieren von Fehlern bedeutet, Ladeoperationen für Datenpartitionen ohne Fehler fortzusetzen und Ladeoperationen für Datenpartitionen mit Fehlern zu stoppen. Fehler können in verschiedenen Datenbankpartitionen übergreifend isoliert werden. Das Dienstprogramm LOAD kann jedoch keine Transaktionen in einer Untergruppe von sichtbaren Datenpartitionen festschreiben und in den verbleibenden sichtbaren Datenpartitionen rückgängig machen.

Sonstige Aspekte

- Eine inkrementelle Indexierung wird nicht unterstützt, wenn einer der Indizes als ungültig markiert ist. Ein Index gilt als ungültig, wenn er erneut erstellt werden muss oder wenn freigegebene abhängige Objekte mit der Anweisung SET INTEGRITY überprüft werden müssen.

- Das Laden von Daten in Tabellen, deren Partitionierung anhand einer beliebigen Kombination aus Algorithmen für das Partitionieren nach Bereich, das Verteilen nach Hash oder das Organisieren nach Dimension erfolgte, wird ebenfalls unterstützt.
- Die Größe von Protokollsätzen, die eine Liste der von der Ladeoperation betroffenen Objekt- und Tabellenbereichs-IDs umfassen (Protokollsätze LOAD START und COMMIT (PENDING LIST)) kann stark anwachsen und so den für andere Anwendungen zur Verfügung stehenden aktiven Protokollspeicher reduzieren.
- Wenn eine Tabelle sowohl partitioniert als auch verteilt ist, kann es sein, dass sich eine Ladeoperation für partitionierte Datenbanken nicht auf alle Datenbankpartitionen auswirkt. Nur die Objekte in den Ausgabedatenbankpartitionen werden geändert.
- Während einer Ladeoperation nimmt die Speicherbelegung für partitionierte Tabellen mit der Anzahl der Tabellen zu. Bitte beachten Sie, dass der Gesamtanstieg nicht linear ist, da nur ein geringer Prozentsatz des Gesamtspeicherbedarfs proportional zur Anzahl der Datenpartitionen ist.

Replizierte MQTs

Eine *MQT (Materialized Query Table, gespeicherte Abfragetabelle)* wird durch eine Abfrage definiert, mit der auch die Daten für die Tabelle festgelegt werden. Durch MQTs kann die Leistungsfähigkeit von Abfragen erhöht werden. Wenn der Datenbankmanager erkennt, dass ein Teil einer Abfrage durch eine MQT aufgelöst werden kann, kann die Abfrage so abgewandelt werden, dass die entsprechende MQT verwendet wird.

In einer Umgebung mit partitionierten Datenbanken können Sie MQTs replizieren und verwenden, um die Abfrageleistung zu steigern. Eine *replizierte MQT* basiert auf einer Tabelle, die möglicherweise in einer Datenbankpartitionsgruppe mit einer Einzelpartition erstellt wurde, die Sie jedoch über alle Datenbankpartitionen in einer anderen Datenbankpartitionsgruppe replizieren möchten. Die replizierte MQT wird durch Ausführen der Anweisung CREATE TABLE mit der Option REPLICATED erstellt.

Mithilfe von replizierten MQTs können Sie Kollokationen zwischen Tabellen herstellen, die normalerweise nicht kollokiert sind. Replizierte MQTs sind besonders nützlich für Joins großer Fakttabellen mit kleinen Dimensionstabellen. Zur Minimierung des erforderlichen zusätzlichen Speicherbedarfs sowie des Aufwands zum Aktualisieren aller Replikate, sollten zu replizierende Tabellen klein sein und nicht häufig aktualisiert werden.

Anmerkung: Sie sollten auch in Erwägung ziehen, umfangreiche Tabellen, die selten aktualisiert werden, zu replizieren: Der einmalige Zusatzaufwand für eine Replikation wird durch die mithilfe der Kollokation erzielten Leistungsvorteile wettgemacht.

Durch Angeben eines geeigneten Vergleichselements in der Subselectklausel, die zum Definieren der replizierten Tabelle verwendet wird, können Sie ausgewählte Spalten und/oder ausgewählte Zeilen replizieren.

DELETE- oder UPDATE-Anweisungen, die nicht deterministische Operationen enthalten, werden bei replizierten MQTs nicht unterstützt.

Tabellenbereiche in Datenbankpartitionsgruppen

Wenn ein Tabellenbereich in einer Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen erstellt wird, werden alle Tabellen innerhalb des Tabellenbereichs auf alle Datenbankpartitionen in der Datenbankpartitionsgruppe verteilt oder partitioniert.

Der Tabellenbereich wird in einer Datenbankpartitionsgruppe erstellt. Wenn der Tabellenbereich in einer Datenbankpartitionsgruppe erstellt wurde, muss er dort verbleiben und kann nicht in eine andere Datenbankpartitionsgruppe versetzt werden. Für die Zuordnung eines Tabellenbereichs zu einer Datenbankpartitionsgruppe wird die Anweisung CREATE TABLESPACE verwendet.

Tabellenpartitionierung und MDC-Tabellen

In einer Tabelle, die sowohl MDC- als auch Datenpartitionstabelle ist, können Spalten in der Bereichspartitionsspezifikation (Range-Partition-Spec) und auch im MDC-Schlüssel für die Tabellenpartitionierung verwendet werden. Eine Tabelle, die diese beiden Merkmale aufweist, ermöglicht eine exaktere Differenzierung des Datenpartitions- und Blockausschlusses, als dies bei Verwendung nur einer dieser Funktionalitäten der Fall wäre.

Es gibt auch zahlreiche Anwendungen, in denen es sinnvoll ist, für den MDC-Schlüssel andere Spalten als die für die Tabellenpartitionierung genutzten Spalten anzugeben. Hierbei ist zu beachten, dass die Tabellenpartitionierung mit mehreren Spalten arbeitet, während beim MDC mit mehreren Dimensionen gearbeitet wird.

Merkmale eines normalen DB2 Data Warehouse

Die folgenden Empfehlungen beziehen sich auf typische, normale Data Warehouses, die in DB2 Version 9.1 neu implementiert wurden. Die folgenden Merkmale werden angenommen:

- Die Datenbank arbeitet auf mehreren Systemen oder in mehreren logischen AIX-Partitionen.
- Umgebungen mit partitionierten Datenbanken werden verwendet (Tabellen werden mithilfe der Klausel DISTRIBUTE BY HASH erstellt).
- Es wurden zwischen vier und 50 Datenpartitionen definiert.
- Die Tabelle, für die die Möglichkeit zur Verwendung des MDC und der Tabellenpartitionierung geprüft wird, ist eine der zentralen Fakttabellen.
- Die Tabelle umfasst zwischen 100.000.000 und 100.000.000.000 Zeilen.
- Neue Daten werden in unterschiedlichen Zeitrahmen geladen: Jeweils über Nacht, wöchentlich, monatlich.
- Das tägliche Aufnahmevolumen liegt zwischen 10.000 und 10 Millionen Datensätzen.
- Die Datenvolumen schwanken: Hierbei liegt das Volumen des Spitzenmonats um das Fünffache höher als das des Monats mit dem geringsten Datenaufkommen. Entsprechend beläuft sich auch der Umfang der größten Dimensionen (Produktlinie, Bereich) auf das Fünffache des Umfangs der kleinsten Dimensionen.
- Die Datenbank enthält detaillierte Datenbestände der letzten 1 - 5 Jahre.
- Abgelaufene Daten werden in monatlichen oder vierteljährlichen Abständen mit einer Rollout-Operation ausgelagert.
- Tabellen verwenden eine Vielzahl von Abfragetypen. Die Workload besteht jedoch größtenteils aus analytischen Abfragen, die in Bezug auf OLTP-Workloads die folgenden Merkmale aufweisen:

  - Es gibt umfangreichere Ergebnismengen mit bis zu 2 Millionen Zeilen.
  - Die Mehrzahl oder alle Abfragen beziehen sich auf Sichten und nicht auf Basistabellen.
- SQL-Klauseln zur Auswahl von Daten nach Bereichen (Klausel BETWEEN), Elementen in Listen etc.

Merkmale einer Fakttabelle eines normalen DB2 Data Warehouse der Version 9.1

Eine normale Data Warehouse-Fakttabelle kann z. B. das folgende Design verwenden:

- Erstellung von Datenpartitionen über die Spalte 'Month'.
- Definition einer Datenpartition für jeden Rollout-Zeitraum, z. B. 1 Monat, 3 Monate.
- Erstellung von MDC-Dimensionen für 'Day' und für 1 - 4 weitere Dimensionen. Typische Dimensionen sind: Produktlinie und Bereich.
- Alle Datenpartitionen und MDC-Cluster sind über alle Datenpartitionen verteilt.

Das MDC und die Tabellenpartitionierung bieten teilweise dieselben Vorteile. Die folgende Tabelle enthält mögliche Anforderungen innerhalb Ihres Unternehmens und Empfehlungen zu einem Organisationsschema, die auf der Basis der zuvor festgestellten Merkmale gegeben werden.

Tabelle 6. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Empfehlung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Tabellenpartitionierung und MDC	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines traditionellen Offlinezeitfensters	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines Mikro-Offlinezeitfensters (weniger als 1 Minute)	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.

Tabelle 6. Verwendung der Tabellenpartitionierung mit MDC-Tabellen (Forts.)

Problemstellung	Empfohlenes Schema	Empfehlung
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums bei gleichzeitiger Erhaltung der Tabellenverfügbarkeit für Geschäftsbenutzer, die Abfragen ohne Serviceverluste weiterhin übergeben können	MDC	MDC kann nicht alle in diesem Zusammenhang geltenden Anforderungen voll erfüllen. Die Tabellenpartitionierung eignet sich hier nicht, da die Tabelle nur für kurze Zeit in den Offlinemodus versetzt wird.
Tägliches Laden von Daten (Befehl **LOAD** oder **INGEST**)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Kontinuierliches Laden von Daten (Befehl **LOAD** mit ALLOW READ ACCESS oder Befehl **INGEST**)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Leistung bei der Ausführung von Abfragen für traditionelle BI-Abfragen	Tabellenpartitionierung und MDC	MDC eignet sich besonders gut für Abfragen in Datenkuben und mehreren Dimensionen. Die Tabellenpartitionierung bietet Unterstützung über den Partitionsausschluss.
Minimierung des Reorganisationsaufwandes durch Vermeidung des Reorganisationsbedarfs oder Reduzierung des Aufwands für die Ausführung der Task	MDC	Beim MDC wird das Clustering beibehalten, sodass sich die Notwendigkeit von Reorganisationen verringert. Wenn das MDC verwendet wird, dann bietet die Datenpartitionierung auch in Teilbereichen keine Vorteile. Allerdings ermöglicht die Tabellenpartitionierung bei Nichtverwendung des MDC die Reduzierung des Reorganisationsbedarfs, indem auf Partitionsebene ein etwas groberes Clustering beibehalten wird.

Beispiel 1:

Sie arbeiten mit einer Tabelle mit den Schlüsselspalten 'YearAndMonth' und 'Province'. Bei der Planung dieser Tabelle wäre eine Partitionierung nach Datum mit einer Zeitspanne von zwei Monaten pro Datenpartition sinnvoll. Darüber hinaus können Sie die Daten auch nach der Spalte 'Province' organisieren, sodass alle Zeilen für ein bestimmtes Gebiet innerhalb eines Datumsbereichs von zwei Monaten in einer Gruppe zusammengefasst werden. Diese Vorgehensweise ist in Abb. 6 auf Seite 37 dargestellt.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

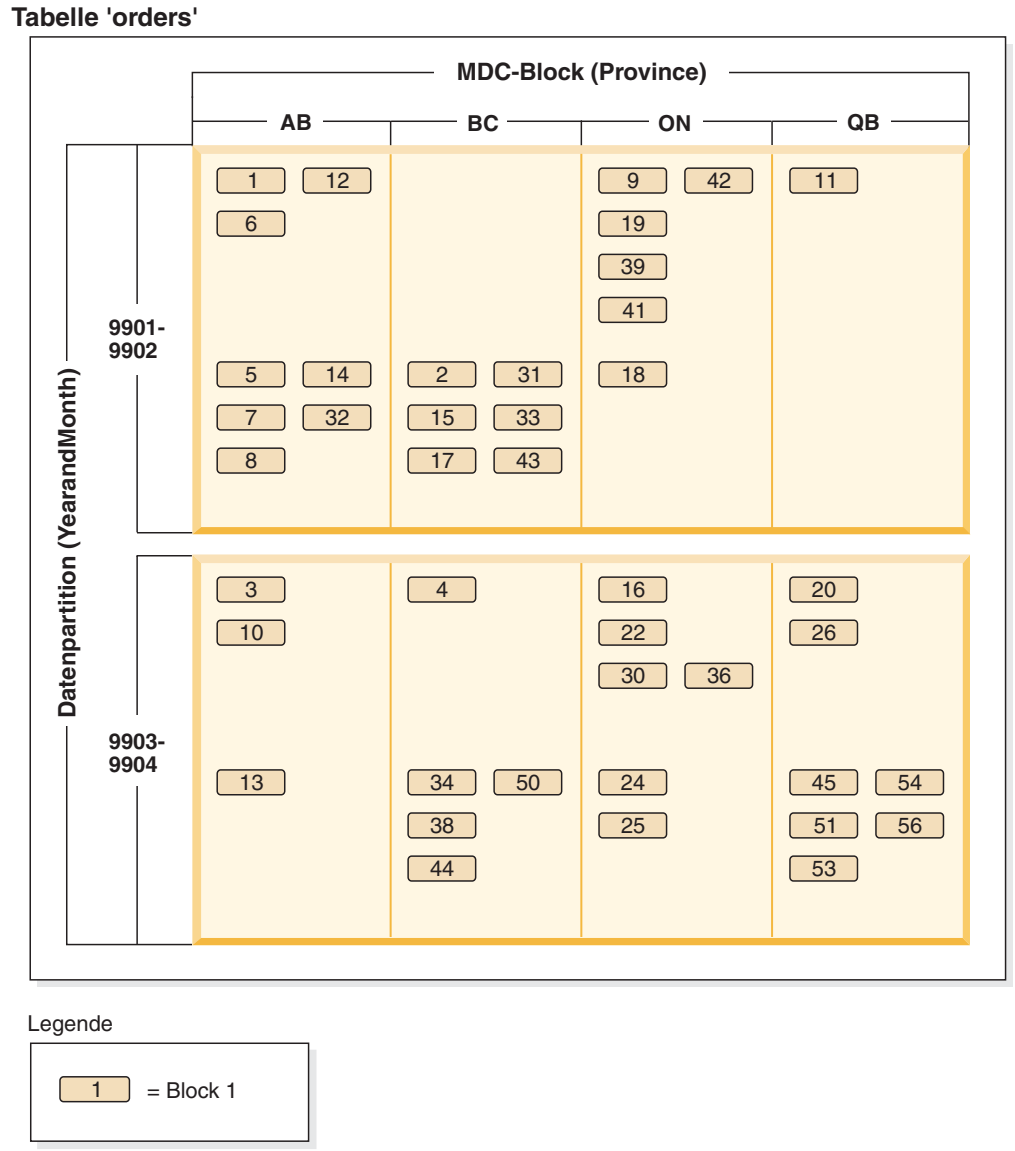


Abbildung 6. Eine nach 'YearAndMonth' partitionierte und nach 'Province' organisierte Tabelle

Beispiel 2:

Eine exaktere Differenzierung kann durch Hinzufügen von 'YearAndMonth' zur Klausel ORGANIZE BY DIMENSIONS (siehe hierzu Abb. 7 auf Seite 38) erzielt werden.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

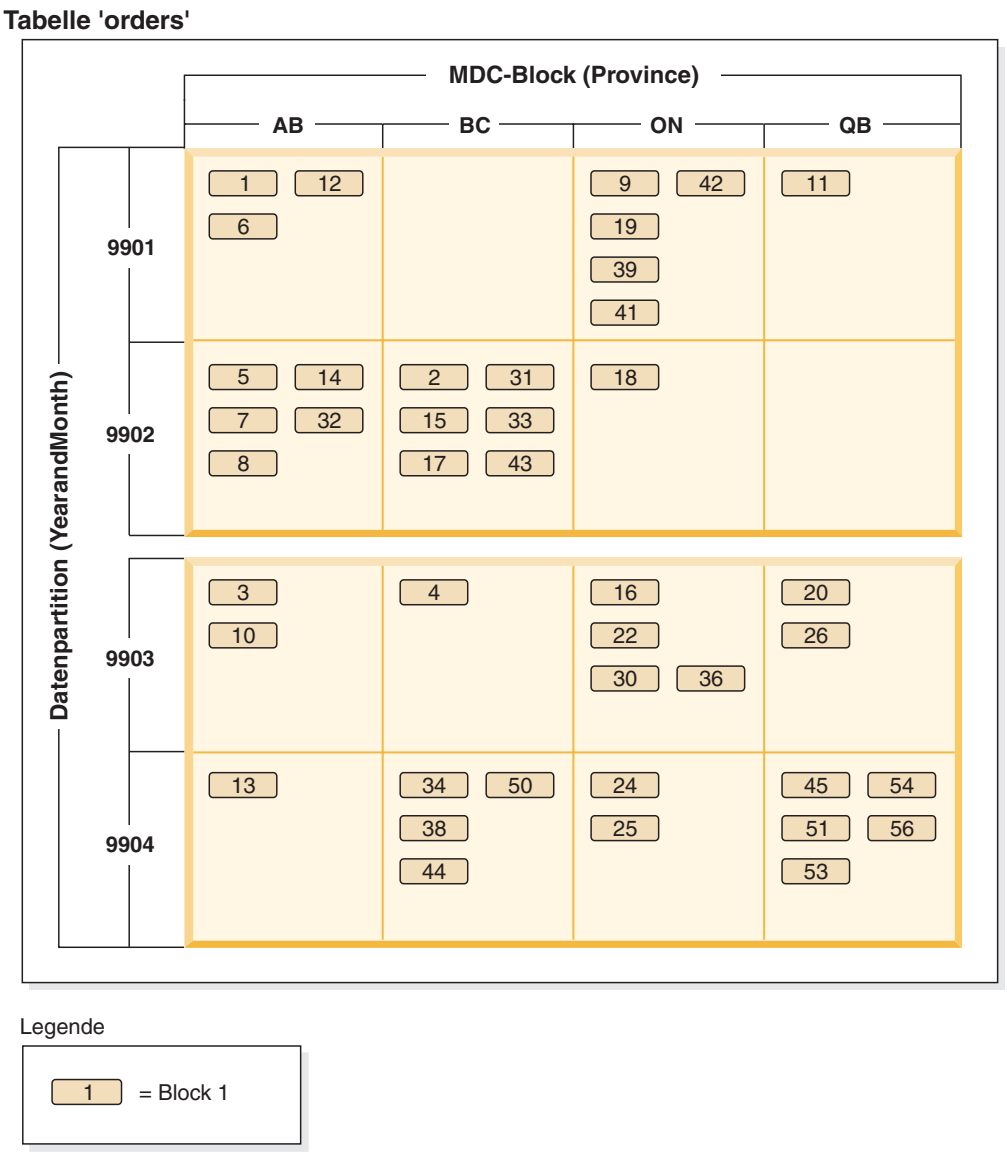


Abbildung 7. Eine nach 'YearAndMonth' partitionierte und nach 'Province' und 'YearAndMonth' organisierte Tabelle

Wenn die Partitionierung so festgelegt wurde, dass jeder Bereich nur einen einzigen Wert enthält, ergeben sich keine Vorteile, wenn die Tabellenpartitionierungsspalte in den MDC-Schlüssel aufgenommen wird.

Wichtige Hinweise

- Im Vergleich mit einer Basistabelle benötigen sowohl MDC-Tabellen als auch partitionierte Tabellen mehr Speicherplatz. Diese Speicherplatzanforderungen gelten zusätzlich zu den sonstigen Anforderungen, sind jedoch unter Berücksichtigung der sich daraus ergebenden Vorteile sinnvoll.
- Wenn Sie die Tabellenpartitionierung und die MDC-Funktionalität in Ihrer partitionierten Datenbankumgebung nicht zusammen einsetzen wollen, dann sollten Sie die Tabellenpartitionierung in den Fällen einsetzen, in denen die Datenverteilung verlässlich vorausgesagt werden kann. Dies ist normalerweise bei den hier erläuterten Systemtypen der Fall. Andernfalls sollten Sie die Verwendung von MDC in Betracht ziehen.

- Bei einer MDC-Datenpartitionstabelle, die mit DB2 Version 9.7 Fixpack 1 (oder einem neueren Release) erstellt wurde, sind die MDC-Blockindizes für die Tabelle partitioniert. Bei einer MDC-Datenpartitionstabelle, die mit DB2 V9.7 (oder einem früheren Release) erstellt wurde, sind die MDC-Blockindizes für die Tabelle nicht partitioniert.

Tabellenpartitionierung in einer DB2 pureScale-Umgebung

Sie können die Tabellenpartionierung in DB2 pureScale verwenden, um für eine bessere Leistung große Tabellenobjekte auf mehrere Partitionen aufzuteilen.

Sie können die Tabellenpartitionierung in DB2 pureScale-Tabellen verwenden; dazu gehören Tabellen, die die Klausel PARTITION BY RANGE verwenden. Darüber hinaus können die der Tabellenpartitionierung zugeordneten Befehle in einer DB2 pureScale-Umgebung verwendet werden.

Dies bedeutet beispielsweise, dass alle der folgenden Operationen unterstützt werden:

- Die Operationen für die Durchführung von Rollout und Rollin für Partitionen, die über die Anweisung ALTER TABLE zur Verfügung stehen
- Die Klauseln PARTITIONED und NOT PARTITIONED für die Anweisung CREATE INDEX
- Für partitionierte Indizes die Klausel ON DATA PARTITION der Anweisungen REORG TABLE und REORG INDEXES ALL

Darüber hinaus wurde für die Tabellenfunktion MON_GET_PAGE_ACCESS_INFO ein Update durchgeführt, sodass sie mit partitionierten Tabellen eingesetzt werden kann. Alle vorhandenen Überwachungsfunktionen, die für Datenpartitionen funktionieren, können für DB2 pureScale-Tabellen ausgeführt werden.

Wenn Sie bereits mit DB2 pureScale Feature arbeiten, können Sie die Tabellenpartitionierung zum Lösen von Konkurrenzsituationen beim gemeinsamen Zugriff auf Seiten verwenden. Indem Sie die gemeinsamen Zugriffe über einen breiteren Bereich verteilen, können Sie die Konkurrenz beim Zugriff auf Datenseiten verringern. Analog können Sie auch die Konkurrenz beim Zugriff auf Indexseiten durch die Verwendung partitionierter Indizes verringern.

Anmerkung: Hinsichtlich der Leistung von DB2 pureScale hängt die verwendete Speichermenge von der Anzahl der Tabellenpartitionen und der Anzahl der Indizes ab. Der Umfang der für die Partionierung auf dem Member verwendeten Speicherressourcen hängt von dem Konfigurationsparameter **dbheap** ab. Für die CF werden die Speicherressourcen durch den Konfigurationsparameter **cf_sca_sz** definiert.

Kapitel 2. Bereichsclustertabellen

Eine Bereichsclustertabelle (Range-Clustered Table, RCT) verfügt über ein Tabellenlayoutschema, in dem jeder Datensatz in der Tabelle eine vorbestimmte Satz-ID (RID) besitzt. Die RID ist eine interne Kennung zum Lokalisieren eines Datensatzes in der Tabelle.

Für die Zuordnung eines Datensatzschlüsselwerts zur Speicherposition einer bestimmten Tabellenzeile wird ein Algorithmus verwendet. Diese Methode bietet ungewöhnlich schnellen Zugriff auf bestimmte Tabellenzeilen. Der Algorithmus verwendet kein Hashing, da bei einem Hashverfahren die Schlüssel/Wert-Abfolge nicht erhalten bleibt. Die Beibehaltung dieser Abfolge ist jedoch von großer Bedeutung, da durch sie die Notwendigkeit zur Reorganisation der Tabelle entfällt.

Für jeden Datensatzschlüsselwert in der Tabelle muss Folgendes gelten:

- Eindeutigkeit
- Wert ungleich null
- Integerwert (SMALLINT, INTEGER oder BIGINT)
- Monoton wachsend
- Wert innerhalb einer vordefinierten Bereichsgruppe auf der Basis der einzelnen Spalten im Schlüssel. (Falls dies erforderlich sein sollte, verwenden Sie die Option ALLOW OVERFLOW in der Anweisung CREATE TABLE, um Zeilen mit Schlüsselwerten zuzulassen, die sich außerhalb des definierten Wertebereichs befinden.)

Zusätzlich zu direktem Zugriff auf bestimmte Tabellenzeilen hat die Verwendung von Bereichsclustertabellen weitere Vorteile.

- Es ist weniger Pflege erforderlich. Eine Sekundärstruktur wie z. B. ein B+-Baumstrukturindex, der nach jeder Einfüge-, Aktualisierungs- oder Löschoperation aktualisiert werden muss, ist nicht vorhanden.
- Es ist weniger Protokollieren für Bereichsclustertabellen erforderlich, wenn ein Vergleich mit regulären Tabellen mit B+-Baumstrukturindizes von ähnlicher Größe vorgenommen wird.
- Es ist weniger Pufferpoolspeicher erforderlich. Es wird kein zusätzlicher Speicher zum Speichern von Sekundärstrukturen, z. B. B+-Baumstrukturindex, benötigt.

Speicherbereich für eine Bereichsclustertabelle wird für die Verwendung durch die Tabelle auch dann vorab zugeordnet und reserviert wird, wenn noch keine Datensätze vorhanden sind. Daher benötigen Bereichsclustertabellen keine Steuersätze für freie Speicherbereiche. Bei der Tabellenerstellung sind in der Tabelle keine Datensätze vorhanden. Der gesamte Seitenbereich wird allerdings vorab zugeordnet. Das vorab ausgeführte Zuordnen basiert auf der Datensatzlänge und der maximalen Anzahl der zu speichernden Datensätze. Wenn ein Feld variabler Länge (z. B. VARCHAR) definiert wird, wird die maximale Feldlänge verwendet und die Gesamtdatensatzlänge ist fest. Dies kann zu einer nicht optimalen Speichernutzung führen. Wenn zwischen Schlüsselwerten große Lücken liegen, hat nicht genutzter Speicherplatz negative Auswirkungen auf die Leistung von Bereichssuchen. Bereichssuchen müssen alle möglichen Zeilen innerhalb eines Bereichs aufsuchen, selbst Zeilen, die noch keine Daten enthalten.

Falls eine Schemamodifikation für eine Bereichsclustertabelle erforderlich ist, muss die Tabelle mit einem neuen Schemanamen erneut erstellt und anschließend mit den Daten aus der alten Tabelle gefüllt werden. Beispiel: Wenn die Bereiche einer Tabelle geändert werden müssen, müssen Sie eine Tabelle mit neuen Bereichen erstellen und sie mit Daten aus der alten Tabelle füllen.

Wenn eine Bereichsclustertabelle Überlaufdatensätze zulässt und ein neuer Datensatz einen Schlüsselwert aufweist, der außerhalb des definierten Wertebereichs liegt, wird der betreffende Datensatz dann in einen Überlaufbereich gestellt, der dynamisch zugeordnet wird. Mit wachsender Menge zu diesem Überlaufbereich hinzugefügter Datensätze ist für Operationen, die für diese Tabelle ausgeführt werden und die den Überlaufbereich umfassen, mehr Verarbeitungszeit erforderlich. Je größer der Überlaufbereich ist, desto länger dauert der Zugriff auf diesen Bereich. Wenn dies zu Problemen führt, sollten Sie die Größe des Überlaufbereichs durch Exportieren der Daten in eine neue Bereichsclustertabelle mit größeren Bereichen reduzieren.

Einschränkungen für Bereichsclustertabellen

Es gibt bestimmte Kontexte, in denen Bereichsclustertabellen nicht verwendet werden können, und es gibt bestimmte Dienstprogramme, die nicht mit Bereichsclustertabellen arbeiten können.

Die folgenden Einschränkungen gelten für Bereichsclustertabellen:

- Bereichsclustertabellen dürfen in einer DB2 pureScale-Umgebung nicht angegeben werden (SQLSTATE 42997).
- Partitionstabellen können keine Bereichsclustertabellen sein.
- Deklarierte temporäre Tabellen und erstellte temporäre Tabellen können keine Bereichsclustertabellen sein.
- Automatisch aktualisierte Übersichtstabellen (AST) können keine Bereichsclustertabellen sein.
- Das Dienstprogramm LOAD wird nicht unterstützt. Die Daten können in eine Bereichsclustertabelle über das Dienstprogramm IMPORT oder über eine parallele Einfügeanwendung eingefügt werden.
- Das Dienstprogramm **REORG** wird nicht unterstützt. Bereichsclustertabellen, die mit der Option DISALLOW OVERFLOW definiert werden, müssen nicht reorganisiert zu werden. Für Bereichsclustertabellen, die mit der Option ALLOW OVERFLOW definiert werden, können die Daten im Überlaufbereich nicht reorganisiert werden.
- Die Klausel DISALLOW OVERFLOW in der Tabellenanweisung CREATE TABLE kann nicht angegeben werden, wenn die Tabelle eine Tabelle des Typs Bereichscluster-MQT (Materialized Query Table) ist.
- Der Designadvisor empfiehlt keine Bereichsclustertabellen.
- Mehrdimensionale Cluster und Clusterindizes sind mit Bereichsclustertabellen inkompatibel.
- Wert- und Standardwertkomprimierung wird nicht unterstützt.
- Rückwärtssuchläufe werden für Bereichsclustertabellen nicht unterstützt.
- Der Parameter **REPLACE** im Befehl **IMPORT** wird nicht unterstützt.
- Die Option WITH EMPTY TABLE in der Anweisung ALTER TABLE...ACTIVATE NOT LOGGED INITIALLY wird nicht unterstützt.

Kapitel 3. Tabellen mit mehrdimensionalem Clustering (MDC)

Tabellen mit mehrdimensionalem Clustering

Das Verfahren des mehrdimensionalen Clustering (MDC - Multidimensional Clustering) stellt eine elegante Methode zur flexiblen, kontinuierlichen und automatischen Anordnung von Daten in Clustern in Bezug auf mehrere Dimensionen bereit. MDC trägt bedeutend zur Verbesserung des Abfrageleistung bei.

Außerdem kann MDC den Aufwand der Datenpflege, wie beispielsweise die Neuorganisation und Indexpflegeoperationen bei INSERT-, UPDATE- und DELETE-Operationen, bedeutend verringern. MDC ist in erster Linie für Data-Warehouse-Umgebungen und große Datenbankumgebungen gedacht und kann darüber hinaus in OLTP-Umgebungen (OLTP - Onlinetransaktionsverarbeitung) genutzt werden.

Vergleich zwischen regulären Tabellen und MDC-Tabellen

Reguläre Tabellen haben datensatzbasierte Indizes. Jedes Clustering der Indizes ist auf eine Dimension beschränkt. Vor Version 8 wurde vom Datenbankmanager lediglich ein eindimensionales Clustering von Daten über Clusterindizes unterstützt. Der Datenbankmanager versucht mithilfe eines Clusterindex die physische Reihenfolge von Daten auf den Seiten in der Schlüsselreihenfolge des Index beizubehalten, wenn Datensätze in die Tabelle eingefügt und aktualisiert werden.

Clusterindizes verbessern die Leistung von Datenbereichsabfragen immens, die Vergleichselemente haben, die den Schlüssel (oder die Schlüssel) des Clusterindex enthalten. Die Leistung wird durch einen guten Clusterindex erhöht, weil nur auf einen Teil der Tabelle zugegriffen werden muss und ein effizienterer Vorablesezugriff ausgeführt werden kann.

Das Datenclustering über einen Clusterindex hat jedoch auch einige Nachteile. Erstens kann das Clustering nicht gewährleistet werden, da sich der Speicherbereich auf Datenseiten mit der Zeit füllt. Eine Einfügeoperation versucht, einen Datensatz auf einer Seite in der Nähe von solchen Datensätzen einzufügen, die die gleichen oder ähnliche Clusterschlüsselwerte haben. Wenn jedoch kein Speicherplatz an der Idealposition frei ist, wird der Datensatz an einer anderen Stelle in die Tabelle eingefügt. Daher können zur Wiederherstellung des Clustering der Tabelle sowie zur Einrichtung von Seiten mit zusätzlichem freien Speicher für zukünftige Clustereinfügeanforderungen in regelmäßigen Abständen Tabellenreorganisationen erforderlich sein.

Zweitens kann nur ein Index als „Clusterindex” definiert werden. Alle anderen Indizes sind ohne Clustering, weil die Daten physisch nur in einer Dimension zusammengefasst ('geclustert') werden können. Diese Einschränkung ergibt sich aus der Tatsache, dass der Clusterindex datensatzbasiert ist, so wie dies bei allen Indizes vor Version 8.1 der Fall war.

Drittens können datensatzbasierte Indizes sehr groß sein, weil sie einen Zeiger für jeden einzelnen Datensatz in der Tabelle enthalten.

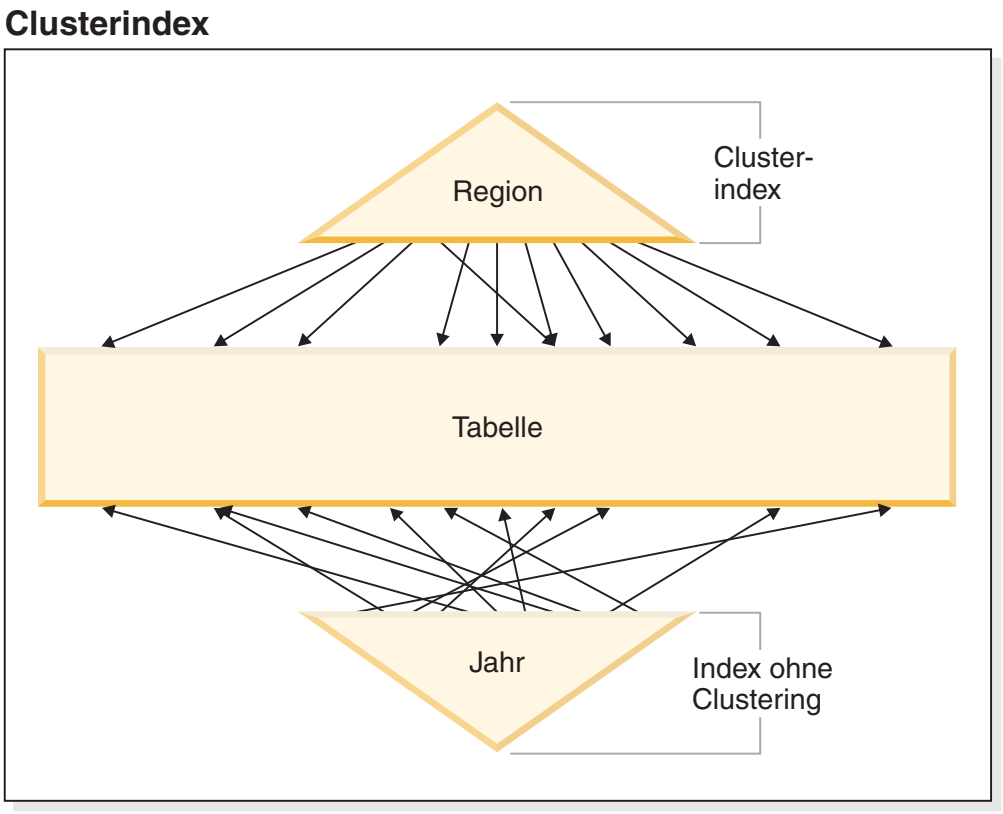


Abbildung 8. Eine reguläre Tabelle mit einem Clusterindex

Für die Tabelle in Abb. 8 sind zwei datensatzbasierte Indizes definiert:

- Ein Clusterindex für „Region"
- Ein anderer Index für „Jahr"

Der Index „Region" ist ein Clusterindex. Dies bedeutet, dass bei der Suche nach Schlüsseln im Index die entsprechenden Datensätze meistenteils auf der gleichen bzw. auf benachbarten Seiten in der Tabelle zu finden sein sollten. Im Unterschied dazu ist der Index „Jahr" ohne Clustering. Das heißt, dass beim Suchen von Schlüsseln in diesem Index die entsprechenden Datensätze wahrscheinlich auf verschiedenen Seiten in der gesamten Tabelle zu finden sind. Suchen über den Clusterindex zeigen eine bessere E/A-Leistung und profitieren umso mehr von sequenziellen Vorablesezugriffen, je mehr die Daten entsprechend diesem Index angeordnet (geclustert) sind.

Mit dem mehrdimensionalen Clustering (MDC) werden Indizes eingeführt, die blockbasiert sind. „Blockindizes" verweisen nicht auf einzelne Datensätze, sondern auf Blöcke bzw. Gruppen von Datensätzen. Durch die physische Anordnung von Daten in einer MDC-Tabelle in Blöcken, die bestimmten Clusteringwerten entsprechen, sowie durch die Verwendung von Blockindizes für den Zugriff auf diese Blöcke kann mehrdimensionales Clustering (MDC) nicht nur alle Nachteile von Clusterindizes wettmachen, sondern bietet darüber hinaus auch erhebliche zusätzliche Leistungsvorteile.

Erstens ermöglicht mehrdimensionales Clustering, die Daten einer Tabelle physisch auf der Grundlage mehrerer Schlüssel bzw. Dimensionen gleichzeitig in Clustern anzuordnen. Mit MDC werden daher die Vorteile eines eindimensionalen Clustering auf mehrere Dimensionen oder Clusterschlüssel ausgeweitet.

Die Abfrageleistung wird überall dort verbessert, wo es ein Clustering einer oder mehrerer angegebener Dimensionen einer Tabelle gibt. Diese Abfragen greifen nicht nur lediglich auf die Seiten zu, die Datensätze mit den richtigen Dimensionswerten enthalten, sondern diese qualifizierten Seiten sind außerdem in Blöcke bzw. EXTENTSIZE große Speicherbereiche gruppiert.

Zweitens kann eine MDC-Tabelle im Gegensatz zu einer Tabelle mit einem Clusterindex, deren Clustering mit der Zeit verloren gehen kann, in den meisten Fällen das Clustering über alle Dimensionen automatisch und kontinuierlich beibehalten und garantieren. Dadurch brauchen MDC-Tabellen zur Wiederherstellung der physischen Anordnung der Daten nicht häufig reorganisiert zu werden. Während die Reihenfolge der Datensätze innerhalb der Blöcke immer beibehalten wird, bleibt die physische Reihenfolge von Blöcken (d. h. von einem Block zum anderen bei einer Blockindexsuche) bei Einfügungen (oder manchmal nicht einmal beim anfänglichen Laden) erhalten.

Dritten sind bei MDC die Clusterindizes blockbasiert. Diese Indizes sind beträchtlich kleiner als die regulären datensatzbasierten Indizes, sodass sie wesentlich weniger Plattenspeicherplatz belegen und sich schneller durchsuchen lassen.

Auswählen von MDC-Tabellendimensionen

Wenn Sie sich zur Arbeit mit MDC-Tabellen entschlossen haben, hängen die auszuwählenden Dimensionen nicht nur vom Typ der Abfragen ab, die auf die Tabellen angewendet werden und von einem Clustering auf Blockebene profitieren, sondern, was wesentlich wichtiger ist, von der Menge und Verteilung Ihrer tatsächlichen Daten.

Abfragen, die von MDC profitieren

Die erste Überlegung bei der Auswahl von Clusteringdimensionen für Ihre Tabelle besteht in der Bestimmung der Abfragen, die vom Clustering auf Blockebene profitieren sollen. In der Regel werden bei der Auswahl von Dimensionen auf der Grundlage der Abfragen, die die zu erledigende Arbeit an den Daten darstellen, mehrere Kandidaten in Betracht kommen. Die Rangfolge dieser Kandidaten spielt eine wichtige Rolle. Spalten, die in Abfragen mit Gleichheits- oder Bereichsvergleichselementen verwendet werden (insbesondere Spalten mit niedrigen Kardinalitäten), bieten den größten Nutzen aus Clusterdimensionen. Ziehen Sie auch die Erstellung von Dimensionen für Fremdschlüssel in einer MDC-Fakttabelle in Betracht, die an Sternjoins mit Dimensionstabellen beteiligt sind. Denken Sie dabei auch an die Leistungsvorteile durch automatisches und kontinuierliches Clustering in mehreren Dimensionen und durch Clustering auf Speicherbereichs- oder Blockebene.

Viele Abfragen können von einem mehrdimensionalen Clustering profitieren. Im Folgenden werden Beispiele solcher Abfragen erläutert. Stellen Sie sich für einige der Beispiele vor, dass eine MDC-Tabelle t1 mit den Dimensionen c1, c2 und c3 vorhanden ist. In anderen Beispielen wird eine MDC-Tabelle 'mdctable' mit den Dimensionen 'Farbe' und 'Land' angenommen.

Beispiel 1:

```
SELECT .... FROM t1 WHERE c3 < 5000
```

Diese Abfrage enthält ein Bereichsvergleichselement für eine einzelne Dimension, sodass sie intern umgeschrieben werden kann, um den Zugriff auf die Tabelle über den Dimensionsblockindex für Spalte c3 durchzuführen. Der Index wird nach Block-IDs (BIDs) von Schlüsseln durchsucht, die Werte kleiner 5000 haben. Auf die resultierende Gruppe von Blöcken wird eine relationale Minisuche angewendet, um die tatsächlichen Datensätze abzurufen.

Beispiel 2:

```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

Diese Abfrage enthält ein Vergleichselement IN für eine einzelne Dimension und kann Suchen auf der Grundlage eines Blockindex auslösen. Diese Abfrage kann intern so umgeschrieben werden, dass auf die Tabelle über den Dimensionsblockindex für Spalte c2 zugegriffen wird. Der Index wird nach BIDs von Schlüsseln durchsucht, die die Werte 1 und 2037 besitzen. Auf die resultierende Gruppe von Blöcken wird eine relationale Minisuche durchgeführt, um die tatsächlichen Datensätze abzurufen.

Beispiel 3:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' AND LAND='USA'
```

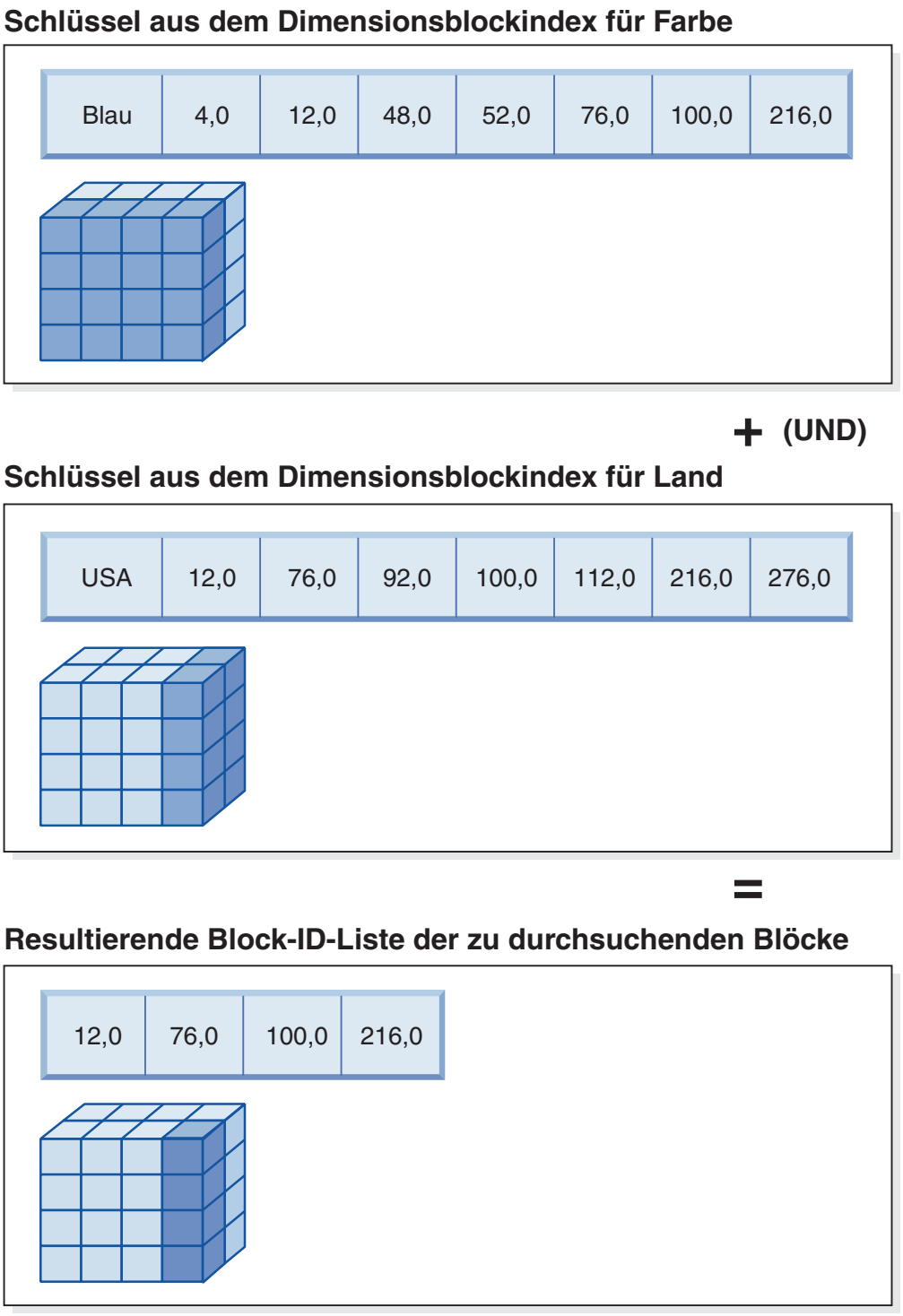


Abbildung 9. Eine Abfrageanforderung mit einer logischen UND-Operation und zwei Blockindizes

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 9):

- Es werden Suchen in den Dimensionsblockindizes ausgeführt: eine für den Sektor 'Blau' und eine weitere für den Sektor 'USA'.
- Eine logische UND-Operation wird auf die Blöcke angewendet, um die Schnittmenge der beiden Sektoren zu bestimmen. Das heißt, die logische UND-Operation bestimmt nur diejenigen Blöcke, die sich in beiden Sektoren befinden.

- In den resultierenden Blöcken der Tabelle wird eine relationale Minisuche ausgeführt.

Beispiel 4:

```
SELECT ... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 AND c3 < 5000
```

Diese Abfrage enthält Bereichsvergleichselemente für c2 und c3, ein Gleichheitsvergleichselement für c1 sowie eine logische UND-Operation. Diese Abfrage kann intern so umgeschrieben werden, dass auf die Tabelle über jeden der Dimensionsblockindizes zugegriffen wird:

- Eine Suche im Blockindex für c2 wird ausgeführt, um BIDs von Schlüsseln mit Werten größer als 100 zu finden.
- Eine Suche im Blockindex für c3 wird ausgeführt, um BIDs von Schlüsseln mit Werten zwischen 1000 und 5000 zu finden.
- Eine Suche im Blockindex für c1 wird ausgeführt, um BIDs von Schlüsseln mit dem Wert '16/03/1999' zu finden.

Anschließend wird eine logische UND-Operation auf die resultierenden BIDs aus den einzelnen Blocksuchen angewendet, um deren Schnittmenge zu ermitteln. Und schließlich wird eine relationale Minisuche auf die resultierende Gruppe von Blöcken angewendet, um die tatsächlichen Datensätze zu finden.

Beispiel 5:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' OR LAND='USA'
```

Diese Abfrageanforderung wird wie folgt ausgeführt:

- Es werden Suchen in den Dimensionsblockindizes ausgeführt: eine für jeden Sektor.
- Eine logische ODER-Operation wird angewendet, um die Vereinigungsmenge (Union) der beiden Sektoren zu bestimmen.
- In den resultierenden Blöcken der Tabelle wird eine relationale Minisuche ausgeführt.

Beispiel 6:

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

Diese Abfrage enthält ein Bereichsvergleichselement für die Dimension c1, ein Vergleichselement IN für die Dimension c2 sowie eine logische ODER-Operation. Diese Abfrage kann intern so umgeschrieben werden, dass auf die Tabelle über die Dimensionsblockindizes für c1 und c2 zugegriffen wird. Im Dimensionsblockindex für c1 wird eine Suche nach den Werten kleiner 5000 und im Dimensionsblockindex für c2 eine Suche nach den Werten 1, 2 und 3 durchgeführt. Auf die resultierenden BIDs aus den Blockindexsuchen wird eine logische ODER-Operation angewendet. Anschließend wird eine relationale Minisuche auf die resultierende Gruppe von Blöcken angewendet, um die tatsächlichen Datensätze zu ermitteln.

Beispiel 7:

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

Diese Abfrage enthält ein Gleichheitsvergleichselement für die Dimension c1 und ein weiteres Bereichsvergleichselement für eine Spalte, die keine Dimension ist, sowie eine logische UND-Operation.

Diese Abfrage kann intern so umgeschrieben werden, dass auf den Dimensionsblockindex für c1 zugegriffen wird, um die Liste der Blöcke aus dem Sektor der Tabelle abzurufen, der für c1 den Wert 15 aufweist. Wenn ein Satz-ID-Index für Spalte c4 vorhanden ist, kann eine Indexsuche durchgeführt werden, um die Satz-IDs von Datensätzen abzurufen, in denen der Wert von c4 kleiner 12 ist. Anschließend kann die resultierende Gruppe von Blöcken durch eine logische UND-Operation mit dieser Datensatzliste verknüpft werden. Durch die Ermittlung dieser Schnittmenge werden die Satz-IDs eliminiert, die nicht in den Blöcken mit c1 gleich 15 vorhanden sind, und nur die aufgelisteten Satz-IDs, die in den entsprechenden Blöcken gefunden werden, werden aus der Tabelle abgerufen.

Wenn kein Satz-ID-Index für c4 vorhanden ist, kann der Blockindex nach den entsprechenden Blöcken durchsucht und während der relationalen Minisuche in den einzelnen Blöcken das Vergleichselement c4 < 12 auf jeden gefundenen Datensatz angewendet werden.

Beispiel 8:

In einem Szenario mit den Dimensionen Farbe, Jahr und Land sowie mit einem Satz-ID-Index (RID-Index) für die Teilenummer wäre die folgende Abfrage möglich:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' AND TEILENUMMER < 1000
```

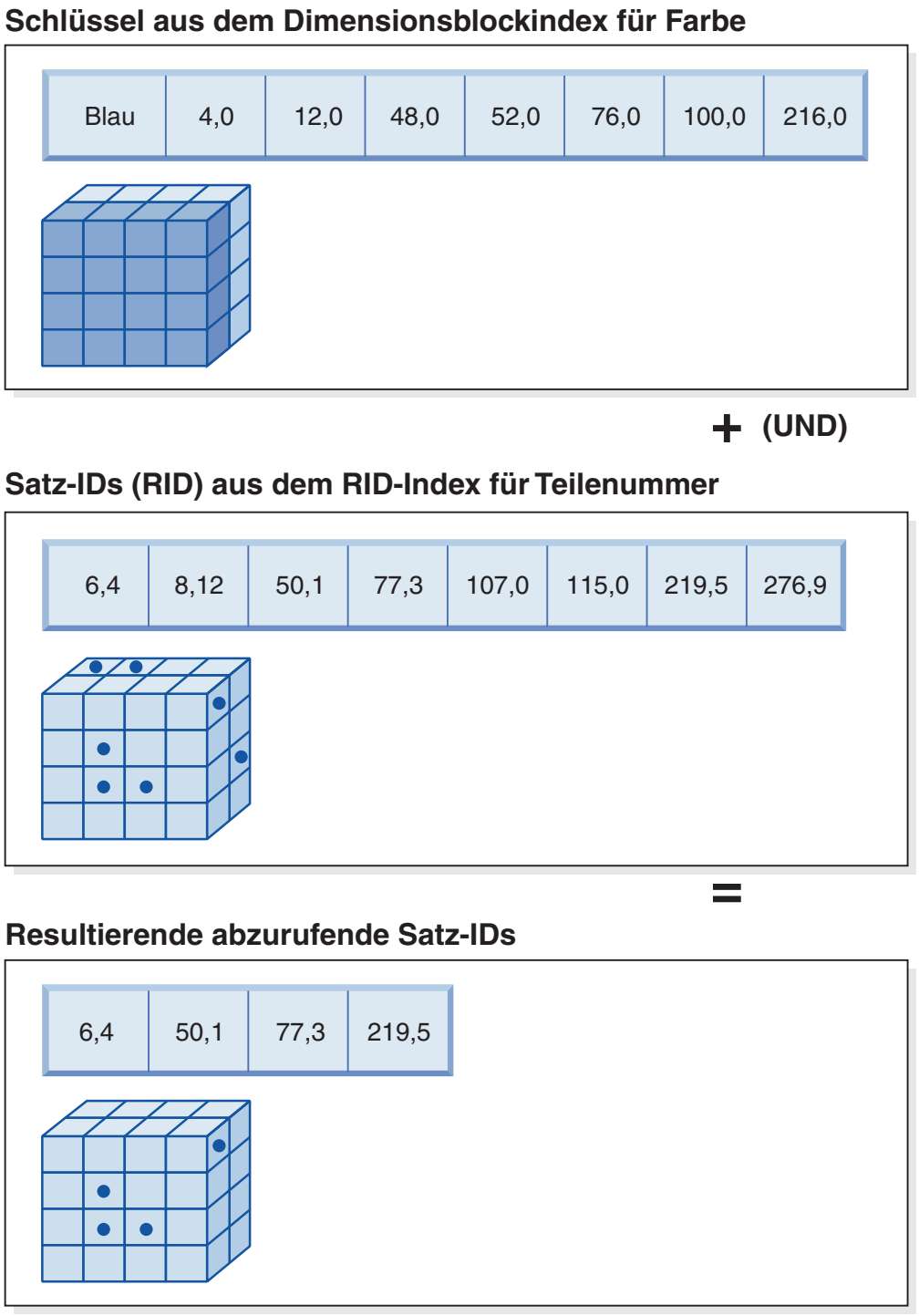


Abbildung 10. Eine Abfrageanforderung mit einer logischen UND-Operation sowie einem Blockindex und einem Satz-ID-Index

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 10 auf Seite 50):

- Es wird eine Suche im Dimensionsblockindex und eine Suche im RID-Index ausgeführt.
- Eine logische UND-Operation wird auf die Blöcke und Satz-IDs angewendet, um die Schnittmenge des Sektors und der Datensätze zu ermitteln, die der Bedingung des Vergleichselements entsprechen.
- Das Ergebnis sind nur die Satz-IDs (RIDs), die zu den qualifizierten Blöcken gehören.

Beispiel 9:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' OR TEILENUMMER < 1000
```

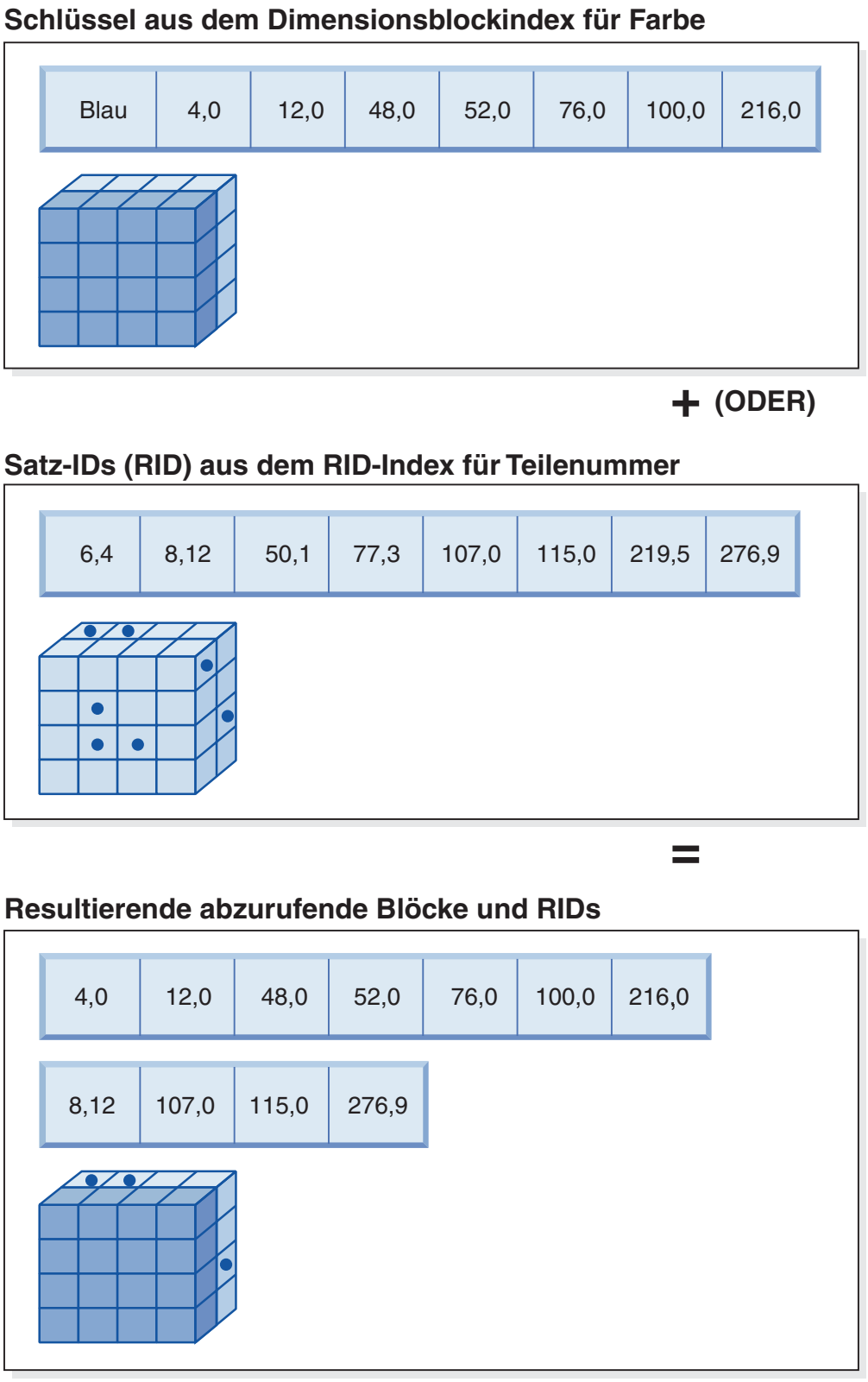


Abbildung 11. Funktionsweise eines Blockindex und eines Satz-ID-Index mit einer logischen ODER-Operation

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 11):

- Es wird eine Suche im Dimensionsblockindex und eine Suche im RID-Index ausgeführt.
- Eine logische UND-Operation wird auf die Blöcke und Satz-IDs angewendet, um die Vereinigungsmenge (Union) des Sektors und der Datensätze zu ermitteln, die der Bedingung des Vergleichselements entsprechen.
- Das Ergebnis umfasst alle Datensätze in den qualifizierten Blöcken und die zusätzlichen Satz-IDs, die außerhalb der qualifizierten Blöcke liegen, jedoch die Bedingung des Vergleichselements erfüllen. In den Blöcken wird eine relationale Minisuche ausgeführt, um deren Datensätze abzurufen, während die zusätzlichen Datensätze außerhalb dieser Blöcke einzeln abgerufen werden.

Beispiel 10:

```
SELECT ... FROM t1 WHERE c1 < 5 OR c4 = 100
```

Diese Abfrage enthält ein Bereichsvergleichselement für Dimension c1, ein Gleichheitsvergleichselement für Spalte c4, die keine Dimension ist, sowie eine logische ODER-Operation. Wenn ein Satz-ID-Index für Spalte c4 vorhanden ist, kann diese Abfrage intern so umgeschrieben werden, dass eine logische ODER-Operation mit dem Dimensionsblockindex für c1 und dem Satz-ID-Index für c4 durchgeführt wird. Wenn kein Index für c4 vorhanden ist, kann stattdessen eine Tabellensuche gewählt werden, da alle Datensätze überprüft werden müssen. Bei der logischen ODER-Operation wird eine Blockindexsuche für c1 nach Werten kleiner 4 sowie eine Satz-ID-Indexsuche für c4 nach Werten gleich 100 durchgeführt. Anschließend wird eine relationale Minisuche in allen qualifizierten Blöcken durchgeführt, da alle Datensätze in diesen Blöcken der Suchbedingung entsprechen, und alle weiteren Satz-IDs für Datensätze außerhalb dieser Blöcke werden ebenfalls abgerufen.

Beispiel 11:

```
SELECT .... FROM t1,d1,d2,d3
      WHERE t1.c1 = d1.c1 and d1.region = 'NY'
      AND t2.c2 = d2.c3 and d2.jahr='1994'
      AND t3.c3 = d3.c3 and d3.produkt='basketball'
```

Diese Abfrage enthält einen Sternjoin. In diesem Beispiel ist t1 die Fakttabelle und hat die Fremdschlüssel c1, c2 und c3, die den Primärschlüsseln der Dimensionstabellen d1, d2 und d3 entsprechen. Die Dimensionstabellen müssen keine MDC-Tabellen sein. Region, Jahr und Produkt sind Spalten der jeweiligen Dimensionstabellen, die mithilfe von normalen Indizes oder Blockindizes indexiert werden können (wenn die Dimensionstabellen MDC-Tabellen sind). Wenn auf eine Fakttabelle über die Werte der Spalten c1, c2 und c3 zugegriffen wird, können Blockindexsuchen in den Dimensionsblockindizes für diese Spalten durchgeführt und anschließend eine logische UND-Operation auf die resultierenden BIDs angewendet werden. Wenn eine Liste von Blöcken vorhanden ist, kann eine relationale Minisuche in jedem Block durchgeführt werden, um die Datensätze abzurufen.

Zelldichte

Die Entscheidungen, die für die geeigneten Dimensionen sowie für den Wert von EXTENTSIZE getroffen werden, sind von **kritischer** Bedeutung für den MDC-Entwurf. Diese Faktoren bestimmen die erwartete Zelldichte der Tabelle. Sie spielen eine wichtige Rolle, da für jede vorhandene Zelle unabhängig von der Anzahl der Datensätze in der Zelle ein EXTENTSIZE großer Speicherbereich zugeordnet wird. Die richtigen Entscheidungen ermöglichen eine vorteilhafte Nutzung der blockbasierten Indexierung und des mehrdimensionalen Clustering, sodass sich Leistungsgewinne realisieren lassen. Das Ziel besteht darin, dicht gefüllte Blöcke zu haben,

um den größtmöglichen Vorteil aus dem mehrdimensionalen Clustering zu ziehen und eine optimale Speicherplatznutzung zu erhalten.

Aus diesem Grund stellt die Dichte der Zellen in der Tabelle, die aufgrund der aktuellen und für die Zukunft absehbaren Daten zu erwarten ist, einen wichtigen Aspekt beim Entwurf einer mehrdimensionalen Tabelle dar. Sie können eine Gruppe von Dimensionen auf der Grundlage der Abfrageleistung auswählen, die bewirken, dass die Anzahl von Zellen in der Tabelle je nach der Anzahl möglicher Werte für die einzelnen Dimensionen potenziell sehr groß wird. Die Anzahl der möglichen Zellen in der Tabelle entspricht dem kartesischen Produkt der Kardinalitäten jeder der gewählten Dimensionen. Wenn Sie zum Beispiel die Daten der Tabelle nach den Dimensionen Tag, Region und Produkt in Clustern zusammenfassen und die Daten fünf Jahre umfassen, kann Ihre Tabelle `1821 Tage * 12 Regionen * 5 Produkte = 109.260` verschiedene mögliche Zellen enthalten. Jede Zelle, die nur einige wenige Datensätze enthält, benötigt trotzdem einen ganzen Block von Seiten, um ihre Datensätze zu speichern. Wenn die Blockgröße groß ist, kann die Tabelle schließlich wesentlich größer werden, als sie eigentlich sein müsste.

Verschiedene Entwurfsfaktoren haben Einfluss auf die optimale Zelldichte:

- Variieren der Anzahl von Dimensionen
- Variieren der Granularität einer oder mehrerer Dimensionen
- Variieren der Blockgröße (EXTENTSIZE) und Seitengröße des Tabellenbereichs

Führen Sie die folgenden Schritte aus, um das optimale Entwurfsergebnis zu erzielen:

1. Ermitteln Sie in Frage kommende Kandidatendimensionen.

   Stellen Sie fest, welche Abfragen von einem Clustering auf Blockebene profitieren würden. Untersuchen Sie die potenzielle Auslastung für Spalten, die einige oder alle der folgenden Merkmale besitzen:

   - Bereich oder Gleichheit in Vergleichselementen mit IN-Listen
   - Ein- oder Auslagerung von Daten
   - GROUP BY- oder ORDER BY-Klauseln
   - Joinklauseln (insbesondere in Sternschema-Umgebungen)

2. Schätzen Sie die Anzahl von Zellen.

   Ermitteln Sie, wie viele potenzielle Zellen in einer Tabelle vorhanden sein können, die nach einer Gruppe von Kandidatendimensionen organisiert ist. Bestimmen Sie die Anzahl der eindeutigen Kombinationen der Dimensionswerte, die in den Daten auftreten. Wenn die Tabelle bereits vorhanden ist, kann die exakte Anzahl für die aktuellen Daten ermittelt werden, indem die Anzahl der unterschiedlichen Werte in jeder der Spalten, die eine Dimension der Tabelle bilden sollen, ausgewählt wird. Alternativ können Sie einen Näherungswert bestimmen, wenn nur die Statistikdaten für eine Tabelle vorliegen, indem Sie die Kardinalitäten der Kandidatenspalten für die geplanten Dimensionen miteinander multiplizieren.

   Anmerkung: Wenn Ihre Tabelle in einer Umgebung mit partitionierten Datenbanken angelegt ist und der Verteilungsschlüssel keinen Bezug zu einer der in Betracht gezogenen Dimensionen hat, ermitteln Sie eine Durchschnittsmenge von Daten pro Zelle, indem Sie die gesamten Daten durch die Anzahl der Datenbankpartitionen dividieren.

3. Schätzen Sie die Speicherbelegung oder Dichte ab.

   Im Durchschnitt gehen Sie von einer Zelle aus, die einen zum Teil gefüllten Block besitzt, in dem nur wenige Datensätze gespeichert sind. Die Anzahl teil-

weise gefüllter Blöcke wird mit abnehmender Anzahl von Datensätzen pro Zelle größer. Beachten Sie außerdem, dass im Durchschnitt (unter der Annahme nur geringer oder keiner Datenabweichung) die Anzahl von Datensätzen pro Zelle durch Dividieren der Anzahl der Datensätze in der Tabelle durch die Anzahl von Zellen ermittelt werden kann. Wenn sich Ihre Tabelle jedoch in einer Umgebung mit partitionierten Datenbanken befindet, sollten Sie berücksichtigen, wie viele Datensätze pro Zelle in jeder Datenbankpartition vorhanden sind, da Blöcke für Daten auf Datenbankpartitionsebene zugeordnet werden. Berücksichtigen Sie bei der Schätzung der Speicherplatzbelegung und der Dichte in einer Umgebung mit partitionierten Datenbanken die durchschnittliche Anzahl von Datensätzen pro Zelle in jeder Datenbankpartition und nicht für die gesamte Tabelle.

Es gibt verschiedene Möglichkeiten zur Verbesserung der Dichte:

- Verringern Sie die Blockgröße, sodass teilweise gefüllte Blöcke weniger Speicherplatz belegen.

  Verringern Sie die Größe der einzelnen Blöcke, indem Sie einen möglichst kleinen geeigneten Wert für EXTENTSIZE wählen. Jede Zelle, die einen teilweise gefüllten Block enthält oder die nur einen Block mit wenigen Datensätzen enthält, verschwendet Speicherplatz. Andererseits benötigen nun Zellen, die viele Datensätze enthalten, mehr Blöcke, um diese aufzunehmen. Infolgedessen erhöht sich die Anzahl von Block-IDs (BIDs) für diese Zellen in den Blockindizes, sodass diese Indizes größer werden und potenziell mehr Einfüge- und Löschoperationen an diesen Indizes verursachen, da Blöcke schneller geleert oder gefüllt werden. Darüber hinaus führt dies zu einer größeren Zahl kleiner Datencluster in der Tabelle für diese volleren Zellwerte im Gegensatz zu einer kleineren Anzahl größerer Datencluster.
- Verringern Sie die Anzahl von Zellen, indem Sie die Anzahl der Dimensionen verkleinern oder die Granularität der Zellen mit einer generierten Spalte erhöhen.

  Sie können eine oder mehrere Dimensionen auf eine geringere Granularität hochstufen, um ihnen eine geringere Kardinalität zu verleihen. Zum Beispiel können Sie die Daten im vorigen Beispiel weiterhin in den Dimensionen 'Region' und 'Produkt' in Clustern zusammenfassen, die Dimension 'Tag' jedoch durch eine Dimension 'JahrUndMonat' ersetzen. Dies ergäbe Kardinalitäten von 60 (12 Monate mal 5 Jahre) 12 und 5 für 'JahrUndMonat', 'Region' und 'Produkt' mit einer Anzahl möglicher Zellen von 3600. Jede Zelle umfasst dann einen größeren Bereich von Werten und enthält mit geringerer Wahrscheinlichkeit nur wenige Datensätze.

  Berücksichtigen Sie auch Vergleichselemente, die am häufigsten auf die beteiligten Spalten angewendet werden (z. B. ob sich viele von ihnen auf Spalten mit Monatsdatums-, Quartals- oder Tagesangaben beziehen). Dies hat Einfluss auf die Entscheidung, ob die Granularität der Dimension geändert werden sollte. Wenn beispielsweise die meisten Vergleichselemente bestimmte Tage betreffen und Sie die Tabelle nach der Spalte 'Monat' zu Clustern zusammengefasst haben, kann DB2 for Linux, UNIX and Windows mithilfe des Blockindexes für 'JahrUndMonat' rasch die Monate eingrenzen, in denen die gewünschten Tage enthalten sind, und nur auf die ihnen zugeordneten Blöcke zugreifen. Beim Durchsuchen der Blöcke muss das Tagesvergleichselement jedoch angewendet werden, um die gesuchten Tage zu bestimmen. Wenn Sie allerdings nach 'Tag' in Clustern zusammenfassen (und dies bedeutet eine hohe Kardinalität), kann der Blockindex für 'Tag' dazu verwendet werden, die zu durchsuchenden Blöcke zu ermitteln, und das Vergleichselement für den Tag muss nur jeweils auf den ersten Datensatz in jeder gefundenen Zelle angewendet werden. In diesem Fall kann es vorteilhafter

sein, eine der anderen Dimensionen mit einer geringeren Granularität zu definieren, um die Dichte der Zellen zu erhöhen. Zum Beispiel kann die Spalte 'Region', die zwölf verschiedene Werte enthält, mithilfe einer benutzerdefinierten Funktion auf die Regionen 'West', 'Nord', 'Süd' und 'Ost' hochgestuft werden.

Aspekte der Erstellung von MDC- oder ITC-Tabellen

Bei der Erstellung von MDC- oder ITC-Tabellen sind zahlreiche Faktoren zu beachten. Die Entscheidungen zur Erstellung, Platzierung und Verwendung Ihrer MDC- oder ITC-Tabellen können durch Ihre aktuelle Datenbankumgebung (z. B. ob es sich um eine partitionierte Datenbank handelt oder nicht) und durch Ihre Auswahl in Bezug auf Dimensionen beeinflusst werden.

Versetzen von Daten aus vorhandenen Tabellen in MDC-Tabellen

Zur Verbesserung der Abfrageleistung und zur Verringerung der Anforderungen für Datenpflegeoperationen in einem Data-Warehouse oder einer großen Datenbankumgebung können Sie Daten aus regulären Tabellen in Tabellen mit mehrdimensionalem Clustering (MDC) versetzen. Gehen Sie wie folgt vor, um Daten aus einer vorhandenen Tabelle in eine MDC-Tabelle zu versetzen:

1. Exportieren Sie Ihre Daten.
2. Löschen Sie die ursprüngliche Tabelle (optional).
3. Erstellen Sie eine MDC-Tabelle (mithilfe der Anweisung CREATE TABLE mit der Klausel ORGANIZE BY DIMENSIONS).
4. Laden Sie Ihre Daten in die MDC-Tabelle.

Eine ALTER TABLE-Prozedur mit dem Namen SYSPROC.ALTOBJ kann zur Ausführung der Umsetzung von Daten aus einer vorhandenen Tabelle in eine MDC-Tabelle verwendet werden. Die Prozedur wird über den DB2-Designadvisor aufgerufen. Die Zeit, die zur Umsetzung der Daten zwischen den Tabellen erforderlich ist, kann enorm sein und hängt von der Größe der Tabelle und der Menge von Daten ab, die umgesetzt werden müssen.

Die Prozedur ALTOBJ führt bei der Änderung einer Tabelle die folgenden Schritte aus:

1. Löschen aller abhängigen Objekte der Tabelle.
2. Umbenennen der Tabelle.
3. Erstellen der Tabelle mit der neuen Definition.
4. Erneutes Erstellen aller abhängigen Objekte der Tabelle.
5. Umsetzen vorhandener Daten in der Tabelle in die Daten, die für die neue Tabelle erforderlich sind. Das heißt, es werden die Daten aus der alten Tabelle ausgewählt und in die neue Tabelle geladen, wobei Spaltenfunktionen zur Umsetzung eines alten Datentyps in einen neuen Datentyp eingesetzt werden können.

Versetzen von Daten aus vorhandenen Tabellen in ITC-Tabellen

Zur Verringerung der Anforderungen für Datenpflegeoperationen können Sie Daten aus regulären Tabellen in ITC-Tabellen (ITC = Insert Time Clustering; Clustering anhand der Einfügungszeit) versetzen. Verwenden Sie die folgende gespeicherte Prozedur zur Onlinetabellenversetzung, um Daten aus einer vorhandenen Tabelle in eine ITC-Tabelle zu versetzen:

Im Szenario "ExampleBank" wird gezeigt, wie Daten aus einer vorhandenen Tabelle in eine ITC-Tabelle versetzt werden. Das Szenario zeigt außerdem, welche Vorteile die Freigabe von Speicherplatz bei der Verwendung von ITC-Tabellen hat. Weitere Informationen finden Sie in den Links zu den zugehörigen Konzepten.

MDC-Advisorfunktion im DB2-Designadvisor

Der DB2-Designadvisor (**db2advis**) verfügt über eine MDC-Funktion. Diese Funktion empfiehlt Clusteringdimensionen zur Verwendung in einer MDC-Tabelle, einschließlich Vergröberungen (d. h. Verringerungen der Granularität) von Basisspalten, um die Auslastungsleistung zu verbessern. Mit der Bezeichnung *Vergröberung* ist ein mathematischer Ausdruck zur Verringerung der Kardinalität (Anzahl unterschiedlicher Werte) in einer Clusterdimension gemeint. Ein allgemeines Beispiel ist die Vergröberung nach Datum, Woche des Datums, Monat des Datums oder Quartal des Jahres.

Eine Voraussetzung für die Verwendung der MDC-Funktion des DB2-Designadvisors ist das Vorhandensein mindestens mehrerer EXTENTSIZE großer Speicherbereiche mit Daten innerhalb der Datenbank. Der DB2-Designadvisor verwendet die Daten zur Modellierung der Datendichte und Kardinalität.

Wenn die Datenbank keine Daten in den Tabellen enthält, empfiehlt der DB2-Designadvisor kein MDC, selbst wenn die Datenbank zwar leere Tabellen enthält, jedoch über nachgebildete Statistikdaten verfügt, die eine mit Daten gefüllte Datenbank simulieren.

Die Empfehlung schließt die Ermittlung potenzieller generierter Spalten mit ein, die eine Vergröberung von Dimensionen definieren. Die Empfehlung bezieht sich nicht auf mögliche Blockgrößen. Bei der Generierung von Empfehlungen für MDC-Tabellen wird der Wert für EXTENTSIZE des Tabellenbereichs zugrunde gelegt. Die Empfehlung geht von der Annahme aus, dass die empfohlene MDC-Tabelle im gleichen Tabellenbereich wie die vorhandene Tabelle erstellt wird, sodass sie die gleiche Speicherbereichsgröße hat. Die Empfehlungen für MDC-Dimensionen variieren entsprechend dem Wert für EXTENTSIZE des Tabellenbereichs, da die Speicherbereichsgröße Auswirkungen darauf hat, wie viele Datensätze in einen Block oder eine Zelle passen. Die Speicherbereichsgröße hat unmittelbaren Einfluss auf die Dichte der Zellen.

Lediglich Dimensionen, die nur aus einer Spalte und nicht aus mehreren Spalten zusammengesetzt sind, werden in Betracht gezogen, obwohl einzelne oder mehrere Dimensionen für die Tabelle empfohlen werden können. Die MDC-Funktion empfiehlt Vergröberungen für die meisten unterstützten Datentypen, um die Kardinalität von Zellen in der resultierenden MDC-Lösung zu reduzieren. Ausnahmen bilden die Datentypen CHAR, VARCHAR, GRAPHIC und VARGRAPHIC. Alle unterstützten Datentypen werden in INTEGER umgesetzt und ihre Kardinalität durch einen generierten Ausdruck verringert.

Das Ziel der MDC-Funktion des DB2-Designadvisors besteht darin, MDC-Lösungen auszuwählen, die zu einer Leistungssteigerung führen. Ein sekundäres Ziel besteht darin, das Anwachsen des Speicherbedarfs der Datenbank auf einer moderaten Ebene zu halten. Das maximale Speicherwachstum jeder Tabelle wird mithilfe einer statistischen Methode ermittelt.

Die Analyseoperation im Designadvisor berücksichtigt nicht nur die Vorteile des Zugriffs über Blockindizes, sondern auch die Auswirkungen des mehrdimensionalen Clustering (MDC) auf Einfüge-, Aktualisierungs- und Löschoperationen für Di-

mensionen der Tabelle. Diese Aktionen an der Tabelle können potenziell bewirken, dass Datensätze zwischen Zellen versetzt werden. Darüber hinaus modelliert die Analyseoperation die potenziellen Leistungsauswirkungen einer Tabellenerweiterung, die sich aus der Organisation von Daten nach bestimmten MDC-Dimensionen ergibt.

Die MDC-Funktion wird über die Markierung `-m <empfehlungstyp>` im Dienstprogramm 'db2advis' ausgeführt. Der Empfehlungstyp „C" dient zur Angabe von MDC-Tabellen. Folgende Empfehlungstypen sind verfügbar: „I" für Index, „M" für MQTs (Materialized Query Tables, gespeicherte Abfragetabellen), „C" für MDC und „P" für eine Umgebung mit partitionierten Datenbanken. Die Empfehlungstypen können miteinander kombiniert werden.

Anmerkung: Der DB2-Designadvisor untersucht keine Tabellen, die weniger als zwölf EXTENTSIZE-Speicherbereiche groß sind.

Der Designadvisor analysiert sowohl MQTs als auch reguläre Basistabellen, um Empfehlungen vorzuschlagen.

Die Ausgabe aus der MDC-Funktion umfasst folgende Informationen:

- Ausdrücke für generierte Spalten für jede Tabelle zur Vergröberung von Dimensionen, die in der MDC-Lösung auftreten.
- Eine Empfehlung für eine ORGANIZE BY DIMENSIONS-Klausel für jede Tabelle.

Die Empfehlungen werden sowohl auf der Standardausgabeeinheit (stdout) als auch in den ADVISE-Tabellen ausgegeben, die zur EXPLAIN-Einrichtung gehören.

MDC-Tabellen und Umgebungen mit partitionierten Datenbanken

Das mehrdimensionale Clustering (MDC) kann in einer Umgebung mit partitionierten Datenbanken verwendet werden. Tatsächlich kann MDC eine Umgebung mit partitionierten Datenbanken ergänzen. Eine Umgebung mit partitionierten Datenbanken dient zur Verteilung von Daten aus einer Tabelle auf mehrere physische oder logische Datenbankpartitionen zu folgenden Zwecken:

- Nutzung der Vorteile mehrerer Maschinen, um die Parallelverarbeitung von Anforderungen zu verbessern
- Physische Vergrößerung der Tabelle über die Grenzen einer einzelnen Datenbankpartition hinaus
- Verbesserung der Skalierbarkeit der Datenbank

Der Grund zur Verteilung einer Tabelle ist davon unabhängig, ob die Tabelle eine MDC-Tabelle oder eine reguläre Tabelle ist. Zum Beispiel unterscheiden sich die Regeln zur Auswahl von Spalten für den Verteilungsschlüssel nicht. Der Verteilungsschlüssel für eine MDC-Tabelle kann jede beliebige Spalte ohne Rücksicht darauf enthalten, ob die Spalten Teil einer Dimension der Tabelle sind oder nicht.

Wenn der Verteilungsschlüssel mit einer Dimension aus der Tabelle identisch ist, enthält jede Datenbankpartition einen anderen Abschnitt der Tabelle. Beispiel: Wenn die MDC-Tabelle aus dem erläuterten Beispiel nach Farbe auf zwei Datenbankpartitionen verteilt wird, wird die Spalte 'Farbe' zur Unterteilung der Daten verwendet. Infolgedessen können zum Beispiel die Sektoren 'Rot' und 'Blau' in einer Datenbankpartition und der Sektor 'Gelb' in einer anderen Datenbankpartition abgelegt werden. Wenn der Verteilungsschlüssel nicht mit den Dimensionen aus der Tabelle identisch ist, wird in jeder Datenbankpartition eine Untergruppe der

Daten aus allen Sektoren gespeichert. Bei der Auswahl der Dimensionen und der Abschätzung der Zellbelegung ist zu beachten, dass sich die durchschnittliche Menge der Daten pro Zelle durch Dividieren der gesamten Daten durch die Anzahl der Datenbankpartitionen bestimmen lässt.

MDC-Tabellen mit mehreren Dimensionen

Wenn Sie wissen, dass bestimmte Vergleichselemente in Abfragen sehr häufig verwendet werden, können Sie die Tabelle nach den beteiligten Spalten in Clustern organisieren. Hierfür können Sie die Klausel ORGANIZE BY DIMENSIONS verwenden.

Beispiel 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

Die Tabelle in Beispiel 1 wird nach Werten aus drei Spalten in Cluster organisiert, sodass ein logischer Würfel (mit drei Dimensionen) gebildet wird. Die Tabelle kann nun bei der Abfrageverarbeitung unter Verwendung einer oder mehrerer dieser Dimensionen logisch in Sektoren untergliedert werden, sodass nur die Blöcke in den entsprechenden Sektoren oder Zellen durch die angewendeten relationalen Operatoren verarbeitet werden. Die Größe eines Blocks (die Anzahl von Seiten) stimmt mit dem Wert für EXTENTSIZE der Tabelle überein.

MDC-Tabellen mit Dimensionen auf der Basis mehrerer Spalten

Jede Dimension kann aus einer oder mehreren Spalten bestehen. Zum Beispiel können Sie eine Tabelle erstellen, die nach einer Dimension mit zwei Spalten organisiert ist.

Beispiel 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

In Beispiel 2 wird die Tabelle in zwei Dimensionen organisiert: c1 und (c3, c4). Bei der Abfrageverarbeitung kann die Tabelle also logisch in einen Sektor nach der Dimension für c1 oder einen Sektor nach der zusammengesetzten Dimension (c3, c4) untergliedert werden. Die Tabelle enthält die gleiche Anzahl von Blöcken wie die Tabelle in Beispiel 1, jedoch einen Dimensionsblockindex weniger. In Beispiel 1 werden drei Dimensionsblockindizes erstellt: jeweils einer für jede der Spalten c1, c3 und c4. In Beispiel 2 werden zwei Dimensionsblockindizes erstellt: einer für Spalte c1 und der andere für die Spalten c3 und c4. Der Hauptunterschied zwischen diesen beiden Ansätzen besteht darin, dass in Beispiel 1 Abfragen, die die Spalte c4 betreffen, den Dimensionsblockindex für c4 nutzen können, um rasch und direkt auf Blöcke mit relevanten Daten zuzugreifen. In Beispiel 2 ist die Spalte c4 der zweite Teil eines Schlüssels in einem Dimensionsblockindex, sodass Abfragen, die die Spalte c4 betreffen, mehr Verarbeitungsaufwand erfordern. Allerdings ist in Beispiel 2 ein Blockindex weniger zu verwalten und zu speichern.

Der DB2-Designadvisor schlägt keine Empfehlungen für Dimensionen aus mehr als einer Spalte vor.

MDC-Tabellen mit Spaltenausdrücken als Dimensionen

Spaltenausdrücke können ebenfalls zum Clustering von Dimensionen verwendet werden. Die Möglichkeit, Daten nach Spaltenausdrücken in Clustern zu organisie-

ren, ist nützlich, wenn Dimensionen in eine gröbere Granularität hochgestuft werden sollen, zum Beispiel wenn eine Adresse auf eine geographische Position bzw. Region oder ein Datum auf eine Woche, Monat oder Jahr abgebildet wird. Zur Implementierung der Hochstufung von Dimensionen auf diese Weise können Sie generierte Spalten verwenden. Dieser Typ von Spaltendefinition ermöglicht die Erstellung von Spalten durch Ausdrücke, die Dimensionen darstellen können. In Beispiel 3 erstellt die Anweisung eine Tabelle, die nach einer Basisspalte und zwei Spaltenausdrücken in Clustern organisiert wird.

Beispiel 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

In Beispiel 3 ist die Spalte c5 ein Ausdruck, der auf den Spalten c3 und c4 basiert. Von der Spalte c6 wird die Spalte c1 in eine grobere Granularität der Zeit hochgestuft. Die Anweisung organisiert die Tabelle in Clustern nach den Werten in den Spalten c2, c5 und c6.

Bereichsabfragen für Dimensionen auf generierten Spalten

Bereichsabfragen für eine Dimension auf einer generierten Spalte erfordern monotone Spaltenfunktionen. Ausdrücke müssen monoton sein, damit Bereichsvergleichselemente für Dimensionen auf generierten Spalten abgeleitet werden können. Wenn Sie eine Dimension für eine generierte Spalte erstellen, können Abfragen auf die Basisspalte den Blockindex für die generierte Spalte nutzen, um die Leistung zu verbessern, allerdings mit einer Ausnahme. Für Bereichsabfragen auf die Basisspalte (z. B. Datum) muss der Ausdruck, der zur Generierung der Spalte in der Anweisung CREATE TABLE verwendet wird, monoton sein, um eine Bereichssuche im Dimensionsblockindex verwenden zu können. Obwohl ein Spaltenausdruck jeden gültigen Ausdruck (einschließlich benutzerdefinierter Funktionen - UDFs) enthalten kann, können bei nicht monotonen Ausdrücken nur Gleichheitsvergleichselemente und Vergleichselemente mit IN-Listen den Blockindex verwenden, um die Abfrage zu erfüllen, wenn sich diese Vergleichselemente auf die Basisspalte beziehen.

Angenommen, eine MDC-Tabelle mit Dimensionen wird auf der Basis der generierten Spalte für Monat mit `monat = INTEGER (datum)/100` erstellt. Für Abfragen auf die Dimension (monat) können Blockindexsuchen ausgeführt werden. Für Abfragen auf die Basisspalte (datum), können ebenfalls Blockindexsuchen ausgeführt werden, um die zu durchsuchenden Blöcke einzugrenzen. Anschließend können die Vergleichselemente für das Datum nur auf die Datensätze in diesen Blöcken angewendet werden.

Der Compiler generiert zusätzliche Vergleichselemente zur Verwendung bei der Blockindexsuche. Zum Beispiel wird die folgende Abfrage formuliert:

```
SELECT * FROM MDCTABLE WHERE DATE > "1999-03-03" AND DATE < "2000/01/15"
```

Der Compiler generiert die folgenden zusätzlichen Vergleichselemente: „month >= 199903" und „month <= 200001". Diese Vergleichselemente können zur Suche im Dimensionsblockindex verwendet werden. Beim Durchsuchen der resultierenden Blöcke werden die ursprünglichen Vergleichselemente auf die Datensätze in den Blöcken angewendet.

Ein nicht monotoner Ausdruck lässt die Anwendung von Gleichheitsvergleichselementen auf diese Dimension zu. Ein gutes Beispiel für eine nicht monotone Funktion ist die Funktion MONTH(), wie sie in der Definition von Spalte c6 in Beispiel 3 zu sehen ist. Wenn die Spalte c1 ein Datum, eine Zeitmarke oder eine gültige Zeichenfolgendarstellung eines Datums oder einer Zeitmarke ist, liefert die Funktion einen Ganzzahlwert aus dem Bereich von 1 bis 12. Obwohl die Ausgabe der Funktion deterministisch ist, ähnelt sie einer Schrittfunktion (d. h. einem zyklischen Muster):

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

Obwohl das Datum in diesem Beispiel kontinuierlich fortschreitet, tut dies der Wert der Funktion MONTH(datum) nicht. Dies bedeutet zum Beispiel, dass nicht garantiert ist, dass bei zwei Werten datum1 und datum2, wobei datum1 größer ist als datum2, der Wert von MONTH(datum1) größer als oder gleich dem Wert von MONTH(datum2) ist. Eben dies ist die Monotoniebedingung. Diese Nichteinhaltung der Monotoniebedingung ist zwar zulässig, aber sie beschränkt die Dimension in der Weise, dass ein Bereichsvergleichselement, das die Basisspalte betrifft, kein Bereichsvergleichselement für die Dimension generieren kann. Jedoch ist ein Bereichsvergleichselement für den Ausdruck problemlos möglich, wie zum Beispiel `where month(c1) between 4 and 6`. Die entsprechende Verarbeitung kann den Index für die Dimension mit einem Startschlüsselwert 4 und einem Endschlüsselwert 6 ganz normal verwenden.

Um diese Funktion monoton zu machen, schließen Sie das Jahr als übergeordnete Ordnungskomponente des Monats mit ein. Es gibt eine Erweiterung zur integrierten Funktion INTEGER(), um die Definition eines monotonen Ausdrucks für Datum zu unterstützten. INTEGER(datum) liefert eine Ganzzahldarstellung des Datums, die dann dividiert werden kann, um eine Ganzzahldarstellung des Jahres und des Monats zu ermitteln. Zum Beispiel liefert `INTEGER(date('2000/05/24'))` den Wert `20000524`. Somit ist `INTEGER(date('2000/05/24'))/100 = 200005`. Die Funktion INTEGER(datum)/100 ist monoton.

Analog verfügen auch die integrierten Funktionen DECIMAL() und BIGINT() über Erweiterungen, die es Ihnen ermöglichen, monotone Funktionen abzuleiten. Die Funktion DECIMAL(zeitmarke) liefert eine dezimale Darstellung einer Zeitmarke und ihr Wert kann in monotonen Ausdrücken zur Ableitung steigender Werte für Monat, Tag, Stunde, Minute usw. verwendet werden. Die Funktion BIGINT(datum) liefert eine BIGINTEGER-Darstellung des Datums ähnlich wie INTEGER(datum).

Der Datenbankmanager bestimmt die Monotonie eines Ausdrucks, falls möglich, bei der Erstellung der generierten Spalte für die Tabelle oder bei der Erstellung einer Dimension aus einem Ausdruck in der DIMENSIONS-Klausel. Bestimmte Funktionen können als Monotonie bewahrend erkannt werden, wie zum Beispiel DATENUM( ), DAYS( ), YEAR( ). Außerdem sind verschiedene mathematische Ausdrücke wie Division, Multiplikation oder Addition einer Spalte und einer Konstante Monotonie bewahrend. In Fällen, in denen DB2 ermittelt, dass ein Ausdruck die Monotonie nicht bewahrt, oder wenn sich dies nicht feststellen lässt, unterstützt die Dimension nur die Verwendung von Gleichheitsvergleichselementen in der entsprechenden Basisspalte.

Ladeaspekte für MDC- und ITC-Tabellen

Wenn Sie regelmäßig Daten in Ihr Data-Warehouse versetzen, können Sie MDC-Tabellen (Multidimensional Clustering Table - mehrdimensionale Clusteringtabelle) vorteilhaft nutzen. In MDC-Tabellen verwendet das Programm LOAD zunächst geleerte Blöcke in der Tabelle wieder, bevor es die Tabelle erweitert und neue Blöcke für die verbleibenden Daten hinzufügt.

Wenn Sie eine Datengruppe gelöscht haben, zum Beispiel alle alten Daten für einen Monat, können Sie das Dienstprogramm LOAD verwenden, um die Daten des nächsten Monats einzuladen, wobei die Blöcke wiederverwendet werden können, die geleert wurden, nachdem die Löschoperation festgeschrieben wurde. Sie haben auch die Möglichkeit, die MDC-Rolloutfunktion mit verzögerter Bereinigung zu verwenden. Wenn der Rollout, bei dem es sich auch um ein Löschen handelt, festgeschrieben ist, sind die Blöcke nicht frei und können noch nicht wiederverwendet werden. Ein Hintergrundprozess wird aufgerufen, um die Satz-ID-Indizes (RID-Indizes) zu pflegen. Wenn die Pflegeoperation abgeschlossen ist, werden die Blöcke freigegeben und können wiederverwendet werden. Bei ITC-Tabellen (Insert Time Clustering - Clustering anhand der Einfügungszeit) werden Blöcke, die nicht verwendet werden, wenn möglich wiederverwendet, bevor eine Tabellenerweiterung durchgeführt wird. Dies umfasst auch freigegebene Blöcke. Die Durchführung von Rollouts wird in ITC-Tabellen nicht unterstützt.

Beim Laden von Daten in MDC-Tabellen können die Eingabedaten sortiert oder unsortiert sein. Falls die Daten unsortiert sind und die Tabelle mehr als eine Dimension aufweist, sollten Sie Folgendes tun:

- Erhöhen Sie den Wert des Konfigurationsparameters **util_heap_sz**.

  Um eine bessere Leistung des Dienstprogramms LOAD beim Laden von MDC-Tabellen zu erzielen, müssen Sie den Wert des Datenbankkonfigurationsparameters **util_heap_sz** erhöhen. Die Leistung des Algorithmus 'mdc-load' ist besser, wenn dem Dienstprogramm mehr Speicher zur Verfügung steht. Dies verringert die Platten-E/A während des Datenclusterings, das in der LOAD-Phase stattfindet. Falls der Befehl **LOAD** eingesetzt wird, um mehrere MDC-Tabellen gleichzeitig zu laden, muss der Wert von **util_heap_sz** entsprechend erhöht werden.
- Erhöhen Sie den Wert, der mit der Klausel **DATA BUFFER** des Befehls **LOAD** angegeben wird.

  Eine Erhöhung dieses Werts wirkt sich nur auf eine Ladeanforderung aus. Die Zwischenspeichergröße für Dienstprogramme muss ausreichen, um möglicherweise mehrere gleichzeitig zu verarbeitende Ladeanforderungen unterzubringen. Ab Version 9.5 kann der Wert für den Parameter **DATA BUFFER** des Befehls **LOAD** den Wert von **util_heap_sz** temporär überschreiten, falls im System mehr Speicher zur Verfügung steht.
- Stellen Sie sicher, dass die für den Pufferpool verwendete Seitengröße mit der größten Seitengröße für den temporären Tabellenbereich übereinstimmt.

  Während der LOAD-Phase werden zusätzliche Protokollierungen zur Verwaltung der Blockzuordnung ausgeführt. Pro zugeordnetem Speicherbereich werden ca. zwei zusätzliche Protokollsätze erstellt. Um eine zufriedenstellende Leistung zu gewährleisten, muss der Datenbankkonfigurationsparameter **logbufsz** auf einen Wert gesetzt werden, der diesen Umstand berücksichtigt.

Für das Laden von Daten in MDC- und ITC-Tabellen gelten die folgenden Einschränkungen:

- Der Parameter **SAVECOUNT** des Befehls **LOAD** wird nicht unterstützt.

- Der Änderungswert `totalfreespace` für den Dateityp wird nicht unterstützt, da diese Tabellen ihren freien Speicherbereich selbst verwalten.
- Der Änderungswert `anyorder` für den Dateityp ist für MDC- und ITC-Tabellen erforderlich. Werden Daten ohne den Änderungswert `anyorder` in eine MDC- oder ITC-Tabelle geladen, wird dieser Änderungswert vom Dienstprogramm explizit aktiviert.

Bei Verwendung des Befehls **LOAD** mit einer MDC- oder ITC-Tabelle werden Verstöße gegen eindeutige Integritätsbedingungen wie folgt behandelt:

- Wenn die Tabelle vor der Ladeoperation einen eindeutigen Schlüssel enthielt und doppelte Datensätze in die Tabelle geladen werden, verbleibt der ursprüngliche Datensatz in der Tabelle, und die neuen Datensätze werden während der DELETE-Phase gelöscht.
- Enthielt die Tabelle vor der Ladeoperation keinen eindeutigen Schlüssel und werden sowohl ein eindeutiger Schlüssel als auch doppelte Datensätze in die Tabelle geladen, wird nur einer der Datensätze mit dem eindeutigen Schlüssel geladen, und die anderen Datensätze werden während der DELETE-Phase gelöscht.

  Anmerkung: Es gibt keine explizite Methode, mit der bestimmt werden kann, welche Datensätze geladen und welche gelöscht werden.

Das Laden beginnt an einer Blockgrenze, sodass sich Ladeoperationen am besten für Daten, die zu neuen Zellen gehören, zum ersten Füllen einer Tabelle mit Daten oder zum Laden weiterer Daten in ITC-Tabellen eignen.

MDC- und ITC-Ladeoperationen enthalten immer eine BUILD-Phase, da alle MDC- und ITC-Tabellen mit Blockindizes ausgestattet sind.

Protokollierungsaspekte für MDC- und ITC-Tabellen

Die Indexpflege und -protokollierung reduziert sich bei der Verwendung von Dimensionen und damit Blockindizes, im Vergleich zu Fällen, in denen Satzkennungsindizes verwendet werden.

Der Datenbankmanager entfernt die Block-ID aus den Blockindizes erst, wenn der letzte Datensatz in einem gesamten Block gelöscht wurde. Diese Indexoperation wird zurzeit auch protokolliert. Analog dazu fügt der Datenbankmanager erst eine Block-ID in den Blockindex, wenn ein Datensatz in einen neuen Block eingefügt wurde. Bei diesem Datensatz muss es sich um den ersten Datensatz einer logischen Zelle oder eine Einfügung in eine logische Zelle von Blöcken handeln, die derzeit voll sind. Diese Indexoperation wird zurzeit auch protokolliert.

Da Blöcke eine Größe zwischen 2 und 256 Datensatzseiten haben können, ist dieser Pflege- und Protokollierungsaufwand für Blockindizes relativ gering. Einfüge- und Löschoperationen an der Tabelle und den Satzkennungsindizes werden weiterhin protokolliert. Bei Rolloutlöschungen werden die gelöschten Datensätze nicht protokolliert. Stattdessen werden die Seiten mit den Sätzen durch eine Neuformatierung von Teilen der Seiten optisch geleert. Die Änderungen an den neu formatierten Teilen werden protokolliert, die Sätze selbst werden allerdings nicht protokolliert.

Blockindexaspekte für MDC- und ITC-Tabellen

Wenn Sie Dimensionen für eine MDC-Tabelle definieren, werden Dimensionsblockindizes erstellt. Darüber hinaus kann auch ein zusammengesetzter Blockindex erstellt werden, wenn mehrere Dimensionen definiert werden. Wenn Sie hingegen

nur eine Dimension für Ihre MDC-Tabelle definiert haben oder wenn es sich bei Ihrer Tabelle um eine ITC-Tabelle handelt, wird nur ein Blockindex erstellt, der sowohl als Dimensionsblockindex als auch als zusammengesetzter Blockindex verwendet wird. Bei einer MDC- oder ITC-Datenpartitionstabelle wird der MDC- oder ITC-Blockindex der Tabelle partitioniert.

Wenn Sie eine MDC-Tabelle mit Dimensionen in Spalte A und (Spalte A, Spalte B) erstellen, wird ein Dimensionsblockindex für Spalte A und einen Dimensionsblockindex für 'Spalte A, Spalte B' erstellt. Da ein zusammengesetzter Blockindex ein Blockindex aller Dimensionen in der Tabelle ist, dient der Dimensionsblockindex für 'Spalte A, Spalte B' gleichzeitig als zusammengesetzter Blockindex.

Für eine MDC-Tabelle dient der zusammengesetzte Blockindex außerdem zur Verarbeitung von Abfragen, bei denen auf Daten in der Tabelle zugegriffen wird, die bestimmte Dimensionswerte haben. Die Reihenfolge der Schlüsselbestandteile im zusammengesetzten Blockindex wirkt sich möglicherweise auf seine Verwendung oder Anwendbarkeit für die Abfrageverarbeitung aus. Die Reihenfolge der Schlüsselbestandteile wird durch die Reihenfolge der Spalten festgelegt, die in der gesamten zur Erstellung der MDC-Tabelle verwendeten Klausel ORGANIZE BY DIMENSIONS vorgefunden wird. Eine Tabelle wird zum Beispiel mit der folgenden Anweisung erstellt:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

In diesem Fall wird der zusammengesetzte Blockindex für die Spalten (c4, c3, c1, c2) erstellt. Die Spalte c1 ist zwar in der Dimensionsklausel zweimal angegeben, sie wird jedoch nur einmal als Schlüsselbestandteil für den zusammengesetzten Blockindex und in der Reihenfolge ihres ersten Auftretens verwendet. Die Reihenfolge der Schlüsselbestandteile im zusammengesetzten Blockindex macht für die Verarbeitung von Einfügeoperationen keinen Unterschied, kann sich jedoch auf die Abfrageverarbeitung auswirken. Wenn also die Spaltenfolge (c1, c2, c3, c4) im zusammengesetzten Blockindex vorgezogen wird, sollte die Tabelle mit folgender Anweisung erstellt werden:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

Blockindizes für MDC-Tabellen

Dieser Abschnitt erläutert, wie Datensätze in MDC-Tabellen mithilfe von Blockindizes organisiert werden.

Die MDC-Tabelle in Abb. 12 auf Seite 65 ist physisch so organisiert, dass Datensätze, die gleiche Werte für „Region" und „Jahr" aufweisen, in getrennte Blöcke bzw. EXTENTSIZE große Speicherbereiche zusammengruppiert werden. Ein Speicherbereich ist eine Gruppe zusammenhängender Seiten auf der Platte, sodass diese Gruppen von Datensätzen in physisch zusammenhängenden Datenseiten zusammengefasst werden. Jede Tabellenseite gehört genau zu einem Block, und alle Blöcke haben die gleiche Größe (d. h. die gleiche Anzahl von Seiten). Die Größe eines Blocks entspricht der Größe eines EXTENTSIZE großen Speicherbereichs des Tabellenbereichs, sodass sich die Blockgrenzen an den Speicherbereichsgrenzen ausrichten. Im oben gezeigten Fall werden zwei Blockindizes erstellt, einer für die Dimension „Region" und ein weiterer für die Dimension „Jahr". Diese Blockindizes enthalten Zeiger, die nur auf die Blöcke in der Tabelle verweisen. Eine Suche im Blockindex „Region" für alle Datensätze, bei denen „Region" gleich „Ost" ist, findet zwei Blöcke, die den Kriterien entsprechen. Alle Datensätze, bei denen „Regi-

on" gleich „Ost" ist, und nur diese, werden in diesen beiden Blöcken gefunden, und zwar zusammengefasst in diese beiden Gruppen zusammenhängender Seiten bzw. Speicherbereiche. Völlig unabhängig davon findet eine gleichzeitige Suche nach Datensätzen zwischen 1999 und 2000 im Index „Jahr" drei entsprechende Blöcke. Eine Datensuche in jedem dieser drei Blöcke liefert alle Datensätze, und nur diese Datensätze, die zwischen 1999 und 2000 liegen. Außerdem befinden sich diese Datensätze zusammengefasst auf sequenziell angeordneten Seiten innerhalb der einzelnen Blöcke.

Mehrdimensionaler Clusterindex

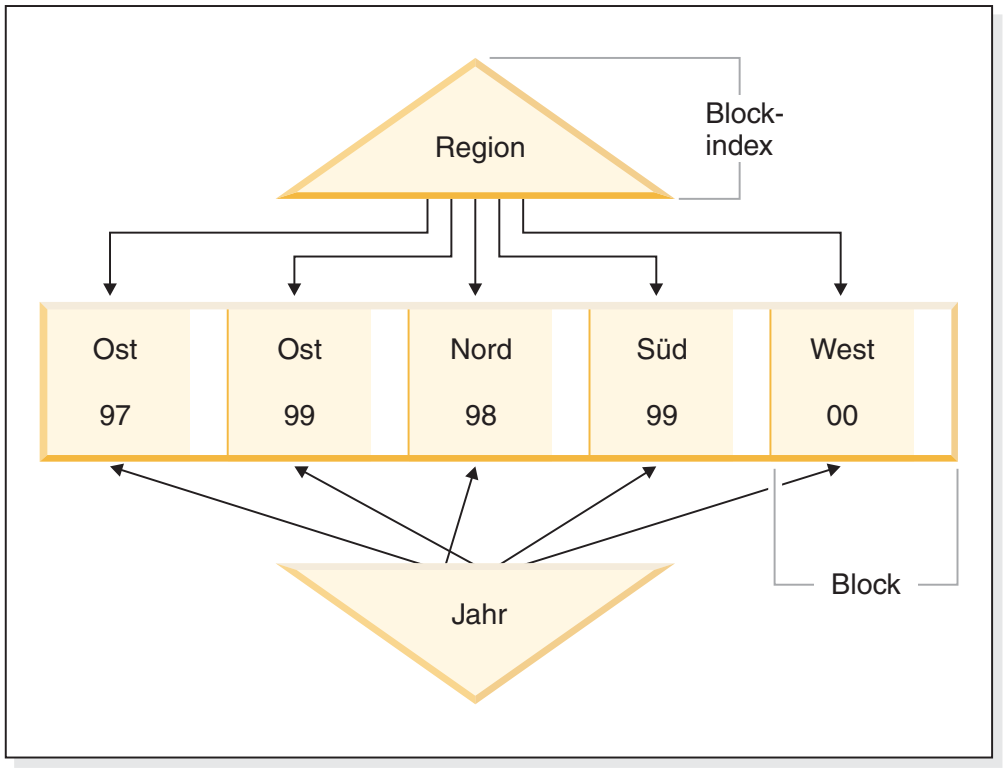


Abbildung 12. Eine Tabelle mit mehrdimensionalem Clustering (MDC)

Über diese Clusteringverbesserungen hinaus bieten MDC-Tabellen zudem die folgenden Vorteile:

- Prüfungen und Suchen in Blockindizes laufen aufgrund ihrer im Vergleich zu datensatzbasierten Indizes unglaublich kleinen Größe wesentlich schneller ab.
- Blockindizes und die entsprechende Anordnung von Daten ermöglichen einen sehr differenzierten „Ausschluss von Datenbankpartitionen" bzw. einen selektiven Tabellenzugriff.
- Abfragen, die Blockindizes nutzen, profitieren von der geringeren Indexgröße, vom optimierten Vorablesezugriff auf Blöcke sowie vom garantierten Clustering der entsprechenden Daten.
- Für einige Abfragen sind weniger Sperren und weniger Aufwand für die Vergleichselementauswertung möglich.
- Blockindizes sind mit wesentlich weniger Protokollier- und Verwaltungsaufwand verbunden, da sie nur aktualisiert werden müssen, wenn der erste Datensatz einem Block hinzugefügt oder der letzte Datensatz aus einem Block gelöscht wird.
- Neu eingefügte Daten können den zusammenhängenden Speicherplatz wiederverwenden, der von Daten freigegeben wurde, die zuvor entfernt wurden.

Anmerkung: Selbst eine MDC-Tabelle, die mit nur einer Dimension definiert wurde, kann von diesen MDC-Attributen profitieren und eine mögliche Alternative zu einer regulären Tabelle mit einem Clusterindex darstellen. Diese Entscheidung sollte auf viele Faktoren gestützt werden, zu denen die Abfragen gehören, die die Auslastung darstellen, sowie die Art und die Verteilung der Daten in der Tabelle. Weitere Informationen finden Sie in „Auswählen von MDC-Tabellendimensionen" auf Seite 45 und „Aspekte der Erstellung von MDC- oder ITC-Tabellen" auf Seite 56.

Wenn Sie eine Tabelle erstellen, können Sie einen oder mehrere Schlüssel als Dimensionen angeben, über die die Daten zu Clustern anzuordnen sind. Jede dieser MDC-Dimensionen kann ähnlich wie bei regulären Indexschlüsseln aus einer oder mehreren Spalten bestehen. Ein *Dimensionsblockindex* wird automatisch für jede angegebene Dimension erstellt und vom Optimierungsprogramm zum schnellen und effizienten Zugriff auf Daten nach der jeweiligen Dimension verwendet. Ein *zusammengesetzter Blockindex* wird ebenfalls automatisch erstellt. Er enthält alle Spalten über alle Dimensionen hinweg und dient zur Beibehaltung des Clustering der Daten bei Einfüge- und Aktualisierungsaktivitäten. Ein zusammengesetzter Blockindex wird nur in dem Fall erstellt, dass eine einzelne Dimension nicht bereits alle Dimensionsschlüsselspalten enthält. Der zusammengesetzte Blockindex kann ebenfalls vom Optimierungsprogramm ausgewählt werden, um effizient auf Daten zuzugreifen, die den Werten einer Untergruppe oder aller der Spaltendimensionen entsprechen.

Anmerkung: Die Zweckmäßigkeit dieses Index bei der Abfrageverarbeitung hängt von der Reihenfolge seiner Schlüsselteile ab. Die Reihenfolge der Schlüsselteile wird durch die Reihenfolge der Spalten bestimmt, die vom Parser bei der Analyse der Dimensionen ermittelt werden, die in der Klausel ORGANIZE BY DIMENSIONS in der Anweisung CREATE TABLE angegeben wurden. Weitere Informationen finden Sie in „Blockindexaspekte für MDC- und ITC-Tabellen" auf Seite 63.

Blockindizes sind strukturell mit regulären Indizes identisch, abgesehen von der Tatsache, dass sie nicht auf Datensätze, sondern auf Blöcke zeigen. Blockindizes sind kleiner als reguläre Indizes, und zwar um einen Faktor der Blockgröße, multipliziert mit der durchschnittlichen Anzahl von Datensätzen auf einer Seite. Die Anzahl von Seiten in einem Block ist gleich dem Wert für EXTENTSIZE des Tabellenbereichs, der zwischen 2 und 256 Seiten betragen kann. Die Seitengröße kann 4, 8, 16 oder 32 KB sein.

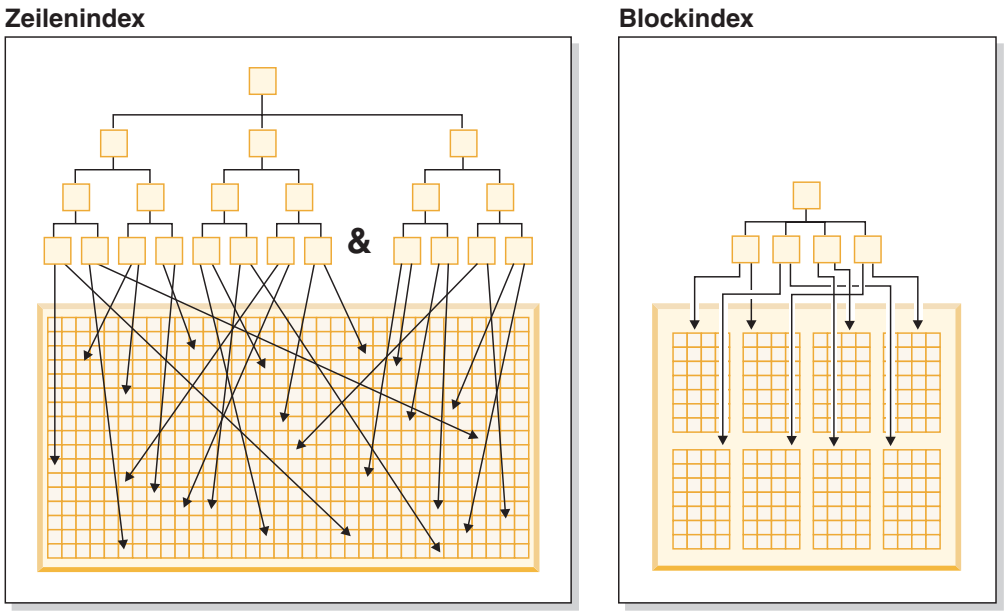


Abbildung 13. Unterschiede zwischen Zeilenindizes und Blockindizes

Wie in Abb. 13 zu sehen ist, gibt es in einem Blockindex jeweils einen Indexeintrag für jeden Block im Unterschied zu einem Eintrag für jede Zeile. Infolgedessen ermöglicht ein Blockindex eine erhebliche Verringerung der Plattenbelegung und einen wesentlich schnelleren Datenzugriff.

In einer MDC-Tabelle bildet jede eindeutige Kombination aus Dimensionswerten eine logische *Zelle*, die sich physisch aus einem oder mehreren Blöcken von Seiten zusammensetzen kann. Der logischen Zelle werden nur so viele Blöcke zugeordnet, wie zur Speicherung der Datensätze erforderlich sind, die den Dimensionswerten dieser logischen Zelle entsprechen. Wenn sich in der Tabelle keine Datensätze befinden, die den Dimensionswerten einer bestimmten logischen Zelle entsprechen, werden dieser logischen Zelle keine Blöcke zugeordnet. Die Gruppe von Blöcken, die Daten mit einem bestimmten Dimensionsschlüsselwert enthalten, werden als *Sektor* (engl. slice) bezeichnet.

Eine MDC-Tabelle kann partitioniert sein. Der Blockindex für eine partitionierte MDC-Tabelle kann nicht partitioniert oder partitioniert sein:

- Bei einer partitionierten MDC-Tabelle, die mit DB2 Version 9.7 Fixpack 1 (oder einem neueren Release) erstellt wurde, sind die Blockindizes für die Tabelle partitioniert.
- Bei einer partitionierten MDC-Tabelle, die mit DB2 V9.7 (oder einem früheren Release) erstellt wurde, sind die Blockindizes für die Tabelle nicht partitioniert.

Nicht partitionierte Blockindizes werden nach einem Upgrade der Datenbank auf DB2 V9.7 Fixpack 1 (oder ein neueres Release) unterstützt.

Szenario: MDC-Tabellen

Als Szenario für die Arbeit mit einer MDC-Tabelle stellen Sie sich eine MDC-Tabelle mit dem Namen „Umsatz" vor, die Verkaufsdaten für einen landesweit tätigen Einzelhändler aufzeichnet. Das Clustering der Tabelle erfolgt nach den Dimensionen „JahrUndMonat" und „Region". Datensätze in der Tabelle werden in Blöcken

gespeichert, die ausreichend aufeinander folgende Seiten enthalten, um einen EXTENTSIZE großen Speicherbereich zu füllen.

In Abb. 14 wird ein Block durch ein Rechteck dargestellt und nach der logischen Reihenfolge der zugeordneten EXTENTSIZE-Bereiche in der Tabelle durchnummeriert. Die Gitterlinien im Diagramm stellen die logische Einteilung dieser Blöcke dar, wobei jedes Quadrat eine logische Zelle bildet. Eine Spalte oder Zeile im Gitter stellt einen Sektor für eine bestimmte Dimension dar. Zum Beispiel befinden sich alle Datensätze, die in der Spalte „Region" den Wert 'Süd-Mitte' enthalten, in den Blöcken, die in dem Sektor enthalten sind, der durch die Spalte 'Süd-Mitte' im Gitter definiert wird. Jeder Block in diesem Sektor enthält ebenfalls nur Datensätze, die im Feld „Region" den Wert 'Süd-Mitte' aufweisen. Das heißt, ein Block ist nur dann in diesem Sektor oder dieser Spalte des Gitters enthalten, wenn er Datensätze enthält, die den Wert 'Süd-Mitte' im Feld „Region" besitzen.

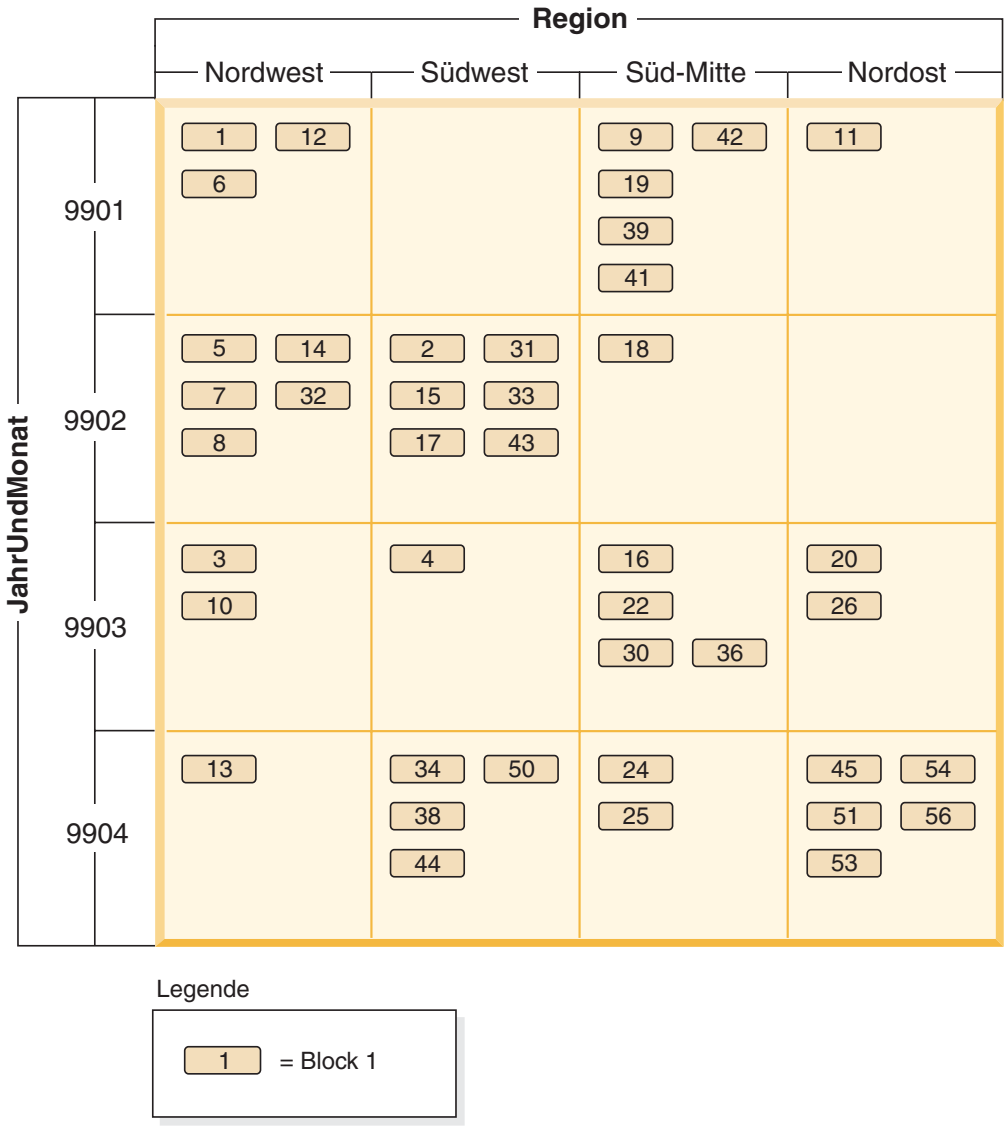


Abbildung 14. Mehrdimensionale Tabelle mit den Dimensionen 'Region' und 'JahrUndMonat' mit dem Namen 'Umsatz'

Zur Bestimmung, welche Blöcke einen Sektor bilden oder, damit äquivalent, welche Blöcke alle Datensätze mit einem bestimmten Dimensionsschlüsselwert enthalten, wird bei der Erstellung der Tabelle automatisch ein Dimensionsblockindex für jede Dimension erstellt.

In Abb. 15 auf Seite 70 wurde ein Dimensionsblockindex für die Dimension „JahrUndMonat" und ein weiterer für die Dimension „Region" erstellt. Jeder Dimensionsblockindex ist in gleicher Weise wie ein herkömmlicher Satz-ID-Index (RID-Index) strukturiert, abgesehen davon, dass die Schlüssel auf der Blattebene auf eine Block-ID (BID) und nicht auf eine Satz-ID (RID) verweisen. Eine Satz-ID (RID) gibt die Position eines Datensatzes in der Tabelle durch eine physische Seitennummer und eine Positionsnummer an, das heißt, der Position auf der Seite, an der sich der Datensatz befindet. Eine Block-ID (BID) stellt einen Block durch die physische Seitennummer der ersten Seite des entsprechenden EXTENTSIZE großen Speicherbereichs und einer Dummy-Position (0) dar. Da alle Seiten im Block angefangen bei dieser Seite physisch aufeinander folgen und die Größe des Blocks bekannt ist, können mithilfe dieser BID alle Datensätze im Block gefunden werden.

Ein Sektor bzw. die Menge von Blöcken, die Seiten mit allen Datensätzen enthalten, die einen bestimmten Schlüsselwert in einer Dimension haben, werden im zugeordneten Dimensionsblockindex durch eine BID-Liste für diesen Schlüsselwert dargestellt.

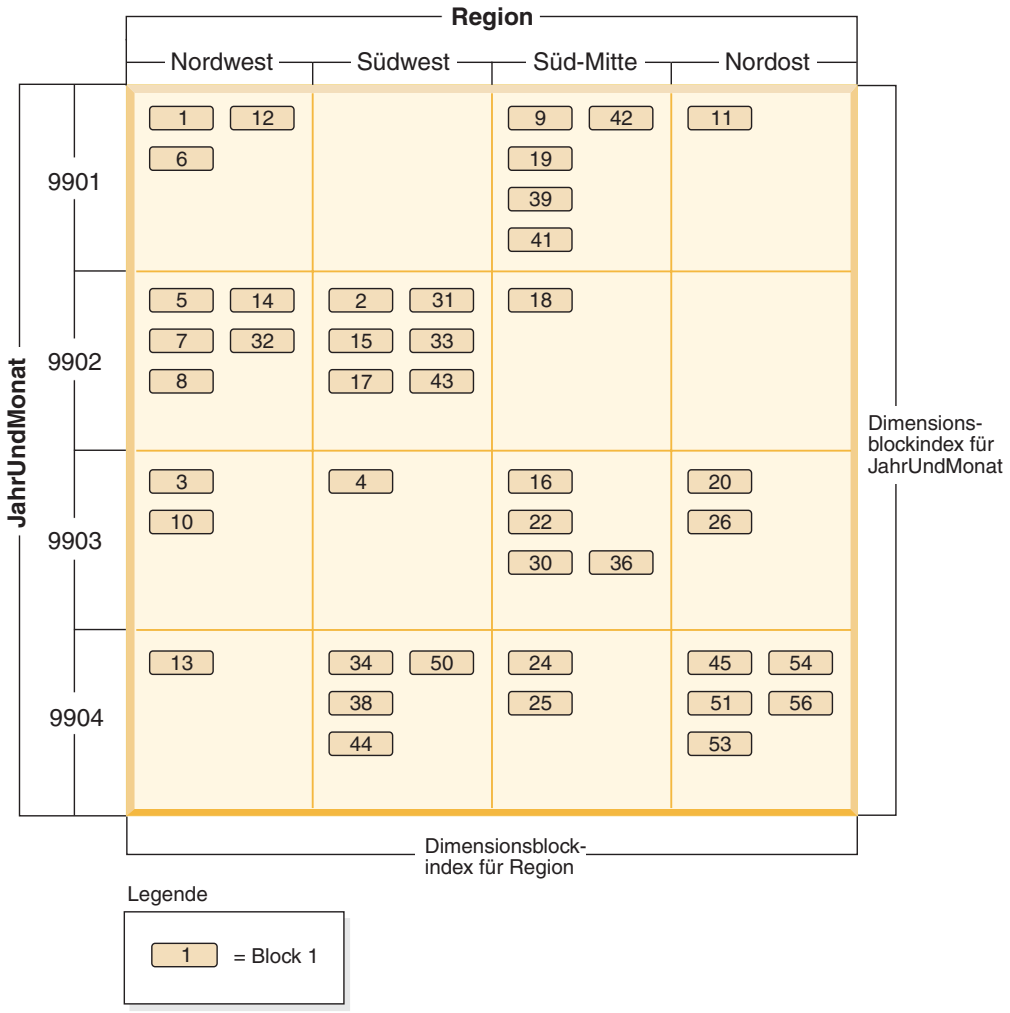


Abbildung 15. Tabelle 'Umsatz' mit den Dimensionen 'Region' und 'JahrUndMonat' mit gezeigten Dimensionsblockindizes

Abb. 16 zeigt, wie ein Schlüssel aus dem Dimensionsblockindex für „Region" aussehen würde. Der Schlüssel besteht aus einem Schlüsselwert, in diesem Fall 'Süd-Mitte', sowie einer Liste von Block-IDs (BIDs). Jede BID enthält eine Blockposition. In Abb. 16 stimmen die aufgeführten Blocknummern mit denen überein, die sich im Sektor 'Süd-Mitte' im Gitter der Umsatztabelle befinden (siehe Abb. 14 auf Seite 68).

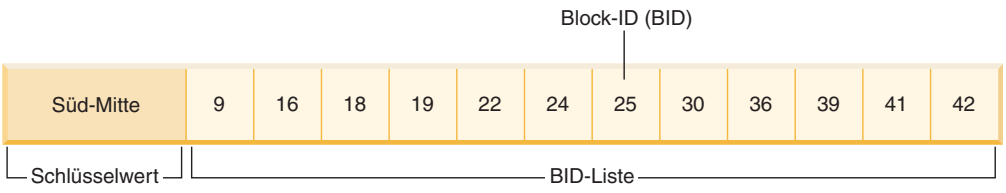


Abbildung 16. Schlüssel aus dem Dimensionsblockindex für 'Region'

Analog lässt sich die Liste von Blöcken, die alle Datensätze mit dem Wert '9902' für die Dimension „JahrUndMonat" enthalten, im Dimensionsblockindex für „JahrUndMonat" finden, der in Abb. 17 auf Seite 71 gezeigt wird.

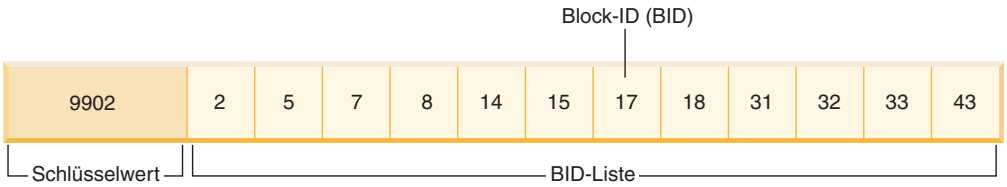


Abbildung 17. Schlüssel aus dem Dimensionsblockindex für 'JahrUndMonat'

Blockindizes und Abfrageleistung für MDC-Tabellen

Suchen in einem der Blockindizes einer MDC-Tabelle ermöglichen einen Clusterdatenzugriff, da jede Block-ID (BID) einer Gruppe sequenzieller Seiten in der Tabelle entspricht, die garantiert Daten enthalten, die den angegebenen Dimensionswert haben. Darüber hinaus kann auf Dimensionen oder Sektoren unabhängig voneinander über die zugehörigen Blockindizes zugegriffen werden, ohne den Clusterfaktor einer anderen Dimension oder eines anderen Sektors zu beeinträchtigen. Dies ermöglicht eine Mehrdimensionalität des mehrdimensionalen Clustering.

Abfragen, die die Vorteile des Blockindexzugriffs nutzen, können von einer Reihe Faktoren profitieren, die sich leistungssteigernd auswirken.

- Da ein Blockindex wesentlich kleiner als ein regulärer Index ist, bietet er eine hohe Sucheffizienz.
- Der Vorablesezugriff auf die Datenseiten hängt nicht von der Sequenzerkennung ab, wenn Blockindizes verwendet werden. Der DB2-Datenbankmanager durchsucht den Index vorausschauend (Look-Ahead) und liest Datenblöcke unter Verwendung von E/A-Operationen in großen Blöcken vorab in den Speicher ein, sodass sichergestellt wird, dass die Suche keinen E/A-Aufwand verursacht, wenn auf die Datenseiten in der Tabelle zugegriffen wird.
- Die Daten in der Tabelle werden auf sequenziell aufeinander folgenden Seiten zusammengefasst, wodurch die Ein-/Ausgabe optimiert und die Ergebnismenge nur aus einem ausgewählten Abschnitt der Tabelle ermittelt wird.
- Werden bei Verwendung eines blockbasierten Pufferpools mit einer Blockgröße, die dem Wert für EXTENTSIZE entspricht, MDC-Blöcke aus sequenziell angeordneten Seiten auf der Platte in sequenziell angeordnete Seiten im Arbeitsspeicher vorab eingelesen, wird der Effekt des Clustering auf die Leistung noch weiter erhöht.
- Die Datensätze aus den einzelnen Blöcken werden mithilfe einer relationalen Minisuche in den zugehörigen Datenseiten abgerufen, was in vielen Fällen eine schnellere Datensuchmethode ist als ein RID-basierter Abruf.

Abfragen können Blockindizes zur Eingrenzung der Suche auf einen Abschnitt der Tabelle verwenden, der einen bestimmten Dimensionswert oder -wertebereich enthält. Dies ermöglicht eine differenzierte Form eines „Datenbankpartitionsausschlusses", das heißt, eines Blockausschlusses. Infolgedessen kann sich für die Tabelle ein besserer gleichzeitiger Zugriff ergeben, weil andere Abfragen, Ladeoperationen, Einfügungen, Aktualisierungen und Löschungen vielleicht auf andere Blöcke in der Tabelle zugreifen, ohne mit der Datenmenge dieser einen Abfrage in Berührung zu kommen.

Wenn die Tabelle 'Umsatz' nach drei Dimensionen in Clustern angeordnet ist, können auch die einzelnen Dimensionsblockindizes zum Auffinden einer Gruppe von Blöcken verwendet werden, die Datensätze enthalten, die eine Abfrage auf eine

Untergruppe aller Dimensionen der Tabelle erfüllen. Wenn die Tabelle die Dimensionen „JahrUndMonat", „Region" und „Produkt" besitzt, kann diese Tabelle als logischer Würfel gedacht werden, wie in Abb. 18 veranschaulicht.

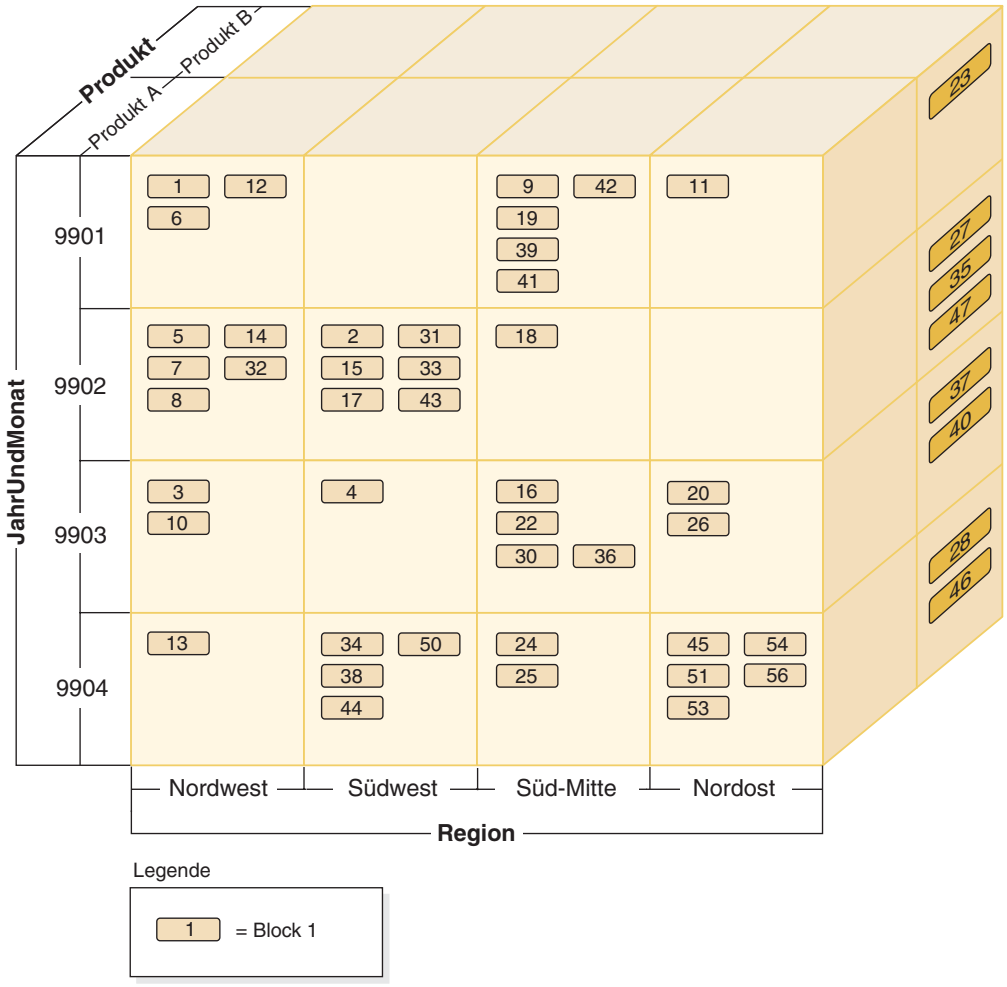


Abbildung 18. Mehrdimensionale Tabelle mit den Dimensionen 'Region', 'JahrUndMonat' und 'Produkt'

Für die MDC-Tabelle in Abb. 18 werden vier Blockindizes erstellt: je einer für die einzelnen Dimensionen „JahrUndMonat", „Region" und „Produkt" sowie ein weiterer mit einem Schlüssel, der all diese Dimensionsspalten umfasst. Um alle Datensätze abzurufen, bei denen „Produkt" gleich „Produkt A" und „Region" gleich „Nordost" gilt, würde der Datenbankmanager zunächst nach dem Schlüssel für Produkt A im Dimensionsblockindex für „Produkt" suchen. (Siehe Abb. 19.) Anschließend bestimmt der Datenbankmanager durch Suchen des Schlüssels „Nordost" im Dimensionsblockindex „Region" die Blöcke, die alle Datensätze mit „Region" gleich „Nordost" enthalten. (Siehe Abb. 20 auf Seite 73.)

Produkt A	1	2	3	...	11	...	20	22	24	25	26	30	...	56

Abbildung 19. Schlüssel aus dem Dimensionsblockindex für 'Produkt'

Nordost	11	20	23	26	27	28	35	37	40	45	46	47	51	53	54	56

Abbildung 20. Schlüssel aus dem Dimensionsblockindex für 'Region'

Blockindexsuchen können durch die logischen Operatoren UND (AND) und ODER (OR) kombiniert werden, wobei die zu durchsuchende resultierende Blockliste ebenfalls einen Clusterdatenzugriff ermöglicht.

Um im vorherigen Beispiel die Menge von Blöcken zu finden, die alle Datensätze mit beiden Dimensionswerten enthalten, müssen Sie die Schnittmenge dieser beiden Sektoren ermitteln. Dies geschieht durch eine logische UND-Operation zwischen den Block-ID-Listen aus den beiden Blockindexschlüsseln. Die gemeinsamen BID-Werte sind 11, 20, 26, 45, 54, 51, 53 und 56.

Das folgende Beispiel veranschaulicht, wie mithilfe der logischen ODER-Operation für Blockindizes eine Abfrage mit Vergleichselementen erfüllt wird, die sich auf zwei Dimensionen beziehen. In Abb. 21 auf Seite 74 wird eine MDC-Tabelle angenommen, die die beiden Dimensionen „Farbe" und „Land" besitzt. Das Ziel besteht darin, alle Datensätze in der MDC-Tabelle abzurufen, auf welche die Bedingungen „Farbe" gleich „blau" oder „Land" gleich „USA" zutreffen.

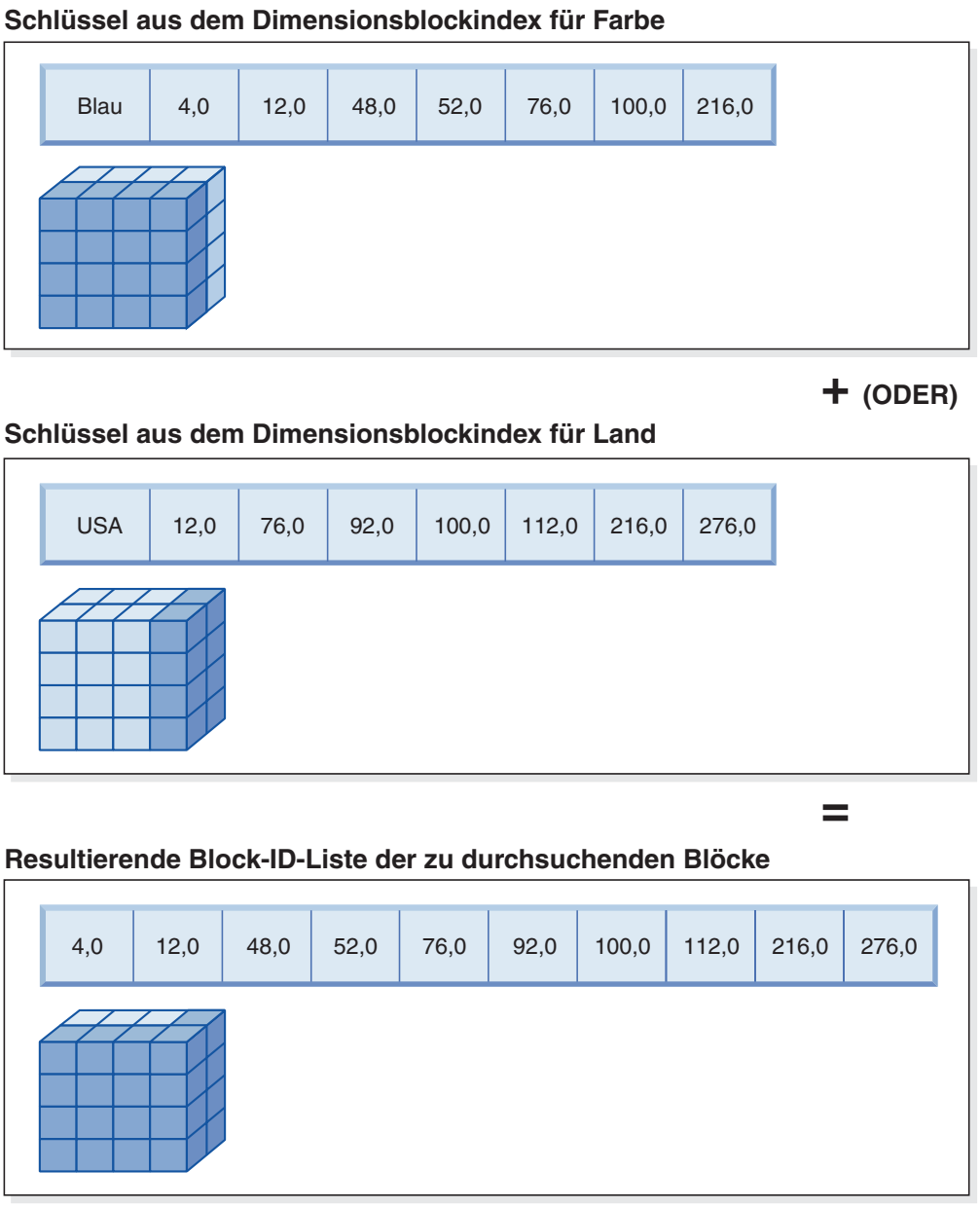


Abbildung 21. Verwendung der logischen ODER-Operation für Blockindizes

Diese Abbildung zeigt, wie die Ergebnisse aus zwei getrennten Blockindexsuchen zur Bestimmung des Bereichs von Werten, die den Einschränkungen der Vergleichselemente entsprechen, kombiniert werden können. (Die Zahlen geben Satz-IDs (RIDs), Bereichsfelder an.)

Auf der Grundlage der Vergleichselemente aus der SELECT-Anweisung werden zwei getrennte Suchen in den Dimensionsblockindizes ausgeführt: eine nach dem Sektor 'Blau' und eine nach dem Sektor 'USA'. Im Arbeitsspeicher wird eine logische ODER-Operation ausgeführt, um die Vereinigungsmenge (Union) der beiden Sektoren zu ermitteln und die kombinierte Gruppe von Blöcken, die sich in beiden Sektoren befinden, zu bestimmen (was eine Eliminierung doppelter Blöcke mit einschließt).

Wenn der Datenbankmanager eine Liste von Blöcken zum Durchsuchen ermittelt hat, kann er eine relationale Minisuche in den einzelnen Blöcken durchführen. Dabei kann ein Vorablesezugriff auf die Blöcke stattfinden, der gerade eine E/A-Operation pro Block erforderlich macht, da jeder Block als ein EXTENTSIZE großer Speicherbereich auf dem Datenträger gespeichert ist und als Einheit in den Pufferpool eingelesen werden kann. Wenn Vergleichselemente auf die Daten angewendet werden müssen, müssen Dimensionsvergleichselemente nur auf einen Datensatz im Block angewendet werden, weil alle Datensätze im Block garantiert die gleichen Dimensionsschlüsselwerte besitzen. Wenn weitere Vergleichselemente vorhanden sind, muss der Datenbankmanager diese nur an den übrigen Datensätzen im Block überprüfen.

MDC-Tabellen unterstützen außerdem reguläre RID-basierte Indizes. RID- und Blockindizes können durch eine logische UND-Operation oder eine ODER-Operation kombiniert werden. Blockindizes stellen dem Optimierungsprogramm zusätzliche Zugriffspläne zur Auswahl, verhindern andererseits die Verwendung der traditionellen Zugriffspläne (z. B. RID-Suchen, Joins und Tabellensuchen) nicht. Blockindexpläne werden durch das Optimierungsprogramm wie alle anderen möglichen Zugriffspläne für eine bestimmte Abfrage hinsichtlich ihres Aufwands geprüft, und der günstigste Plan wird ausgewählt.

Der DB2-Designadvisor kann RID-basierte Indizes für MDC-Tabellen oder MDC-Dimensionen für eine Tabelle empfehlen.

Automatisches Beibehalten des Clustering bei INSERT-Operationen

Die automatische Beibehaltung des Datenclustering in einer MDC-Tabelle wird durch den zusammengesetzten Blockindex sichergestellt. Er wird zur dynamischen Verwaltung und Beibehaltung des physischen Clustering (Gruppierung) von Daten in den Dimensionen der Tabelle über die Folge von INSERT-Operationen hinweg verwendet.

In diesem zusammengesetzten Blockindex findet sich nur je ein Schlüssel für diejenigen logischen Zellen der Tabelle, die Datensätze enthalten. Dieser Blockindex dient bei einer INSERT-Operation so zur raschen und effizienten Feststellung, ob eine logische Zelle in der Tabelle vorhanden ist. Nur wenn dies der Fall ist, wird außerdem festgestellt, welche Blöcke Datensätze mit der bestimmten Gruppe von Dimensionswerten dieser Zelle enthalten.

Wenn eine INSERT-Operation ausgeführt wird, geschieht Folgendes:

- Der zusammengesetzte Blockindex wird auf das Vorhandensein der logischen Zelle sondiert, die den Dimensionswerten des einzufügenden Datensatzes entspricht.
- Wenn der Schlüssel der logischen Zelle im Index gefunden wird, liefert die zugehörige Block-ID-Liste (BIDs) eine vollständige Liste der Blöcke in der Tabelle, die die Dimensionswerte der logischen Zelle enthalten. (Siehe Abb. 22 auf Seite 76.) Dies begrenzt die Anzahl von EXTENTSIZE großen Speicherbereichen der Tabelle, die nach Platz zum Einfügen des Datensatzes zu durchsuchen sind.
- Wenn der Schlüssel der logischen Zelle im Index nicht gefunden wird oder wenn die Speicherbereiche, die diese Werte enthalten, voll sind, wird der logischen Zelle ein neuer Block zugeordnet. Wenn möglich, erfolgt zuerst eine Wiederverwendung eines leeren Blocks in der Tabelle, bevor die Tabelle um einen neuen, EXTENTSIZE großen Speicherbereich von Seiten (einen neuen Block) erweitert wird.

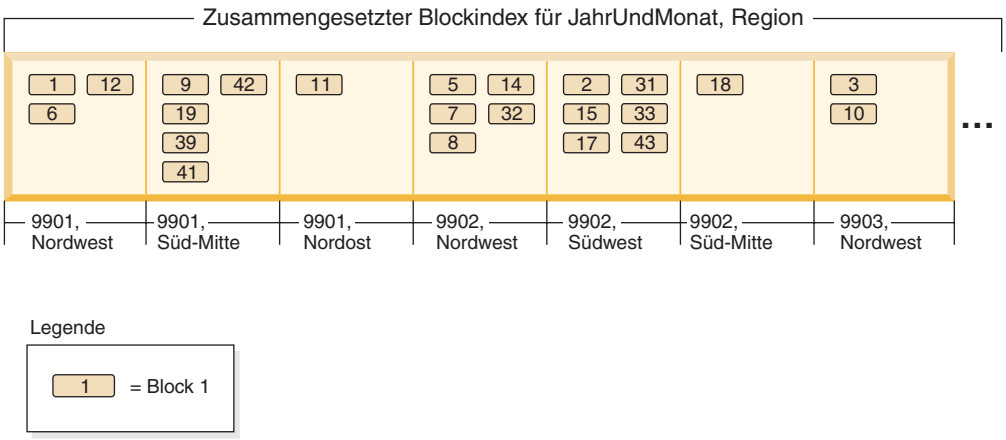


Abbildung 22. Zusammengesetzter Blockindex für 'JahrUndMonat', 'Region'

Datensätze mit bestimmten Dimensionswerten werden garantiert in einer Gruppe von Blöcken gefunden, die ausschließlich und gleichzeitig sämtliche Datensätze mit diesen Werten enthalten. Blöcke bestehen aus aufeinander folgenden Seiten auf der Platte. Infolgedessen erfolgt der Zugriff auf diese Datensätze sequenziell und ermöglicht ein Clustering. Dieses Clustering wird über die Zeit automatisch beibehalten, indem sichergestellt wird, dass Datensätze nur in Blöcke aus Zellen mit den Dimensionswerten der jeweiligen Datensätze eingefügt werden. Wenn vorhandene Blöcke in einer logischen Zelle voll sind, wird entweder ein leerer Block wiederverwendet oder ein neuer Block zugeordnet und der Gruppe von Blöcken für diese logische Zelle hinzugefügt. Wenn ein Block seiner Datensätze entleert wird, wird die Block-ID (BID) aus den Blockindizes entfernt. Dies hebt die Zuordnung des Blocks zu den Werten einer logischen Zelle auf, sodass er in Zukunft von einer anderen logischen Zelle wiederverwendet werden kann. Auf diese Weise werden Zellen und die ihnen zugeordneten Blockindexeinträge der Tabelle dynamisch hinzugefügt und aus ihr entfernt, um je nach Bedarf nur die Daten unterzubringen, die in der Tabelle vorhanden sind. Der zusammengesetzte Blockindex dient zur Verwaltung dieser Funktion, da er logische Zellwerte den Blöcken zuordnet, die Datensätze mit diesen Werten enthalten.

Da das Clustering auf diese Weise automatisch beibehalten wird, wird eine Reorganisation einer MDC-Tabelle zu keinem Zeitpunkt erforderlich, um das Clustering der Daten wiederherzustellen. Allerdings kann eine Reorganisation weiterhin zur Wiedergewinnung von Speicherplatz durchgeführt werden. Wenn Zellen zum Beispiel zahlreiche dünn gefüllte Blöcke haben, deren Daten in weniger Blöcke passen würden, oder wenn die Tabelle viele Zeiger-Überlauf-Paare hat, würde eine Reorganisation der Tabelle die Datensätze, die zu den einzelnen logischen Zellen gehören, in die kleinstmögliche Anzahl erforderlicher Blöcke zusammenfassen und die Zeiger-Überlauf-Paare entfernen.

Das folgende Beispiel veranschaulicht, wie der zusammengesetzte Blockindex zur Abfrageverarbeitung verwendet werden kann. Wenn Sie alle Datensätze der Tabelle in Abb. 22 abrufen wollen, die den Wert 'Nordwest' für „Region” und den Wert '9903' für „JahrUndMonat” haben, würde der Datenbankmanager den Schlüsselwert '9903, Nordwest' im zusammengesetzten Blockindex aufsuchen, wie in Abb. 23 auf Seite 77 gezeigt. Der Schlüssel besteht aus einem Schlüsselwert, in diesem Fall '9903, Nordwest', und einer Liste von BIDs. Wie Sie sehen, sind nur die BIDs 3 und 10 aufgelistet. Und tatsächlich sind in der Tabelle 'Umsatz' nur zwei Blöcke vorhanden, die Datensätze mit diesen beiden Werten enthalten.

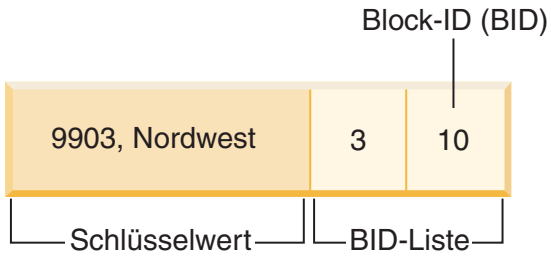


Abbildung 23. Schlüssel aus dem zusammengesetzten Blockindex für 'JahrUndMonat', 'Region'

Zur Veranschaulichung, wie der zusammengesetzte Blockindex bei Einfügeoperationen verwendet wird, soll das Beispiel der Einfügung eines weiteren Datensatzes mit den Dimensionswerten '9903' und 'Nordwest' betrachtet werden. Der Datenbankmanager würde diesen Schlüsselwert im zusammengesetzten Blockindex suchen und die BIDs für die Blöcke 3 und 10 finden. Diese Blöcke enthalten alle Datensätze und die einzigen Datensätze, die diese Dimensionsschlüsselwerte besitzen. Wenn Speicherplatz verfügbar ist, fügt der Datenbankmanager den neuen Datensatz in einen dieser Blöcke ein. Wenn kein Platz in den Seiten dieser Blöcke mehr vorhanden ist, ordnet der Datenbankmanager einen neuen Block für die Tabelle zu oder verwendet einen zuvor geleerten Block in der Tabelle. Beachten Sie, dass in diesem Beispiel Block 48 zurzeit nicht von der Tabelle verwendet wird. Der Datenbankmanager fügt den Datensatz in den Block ein und ordnet diesen Block der aktuellen logischen Zelle zu, indem er die Block-ID (BID) des Blocks dem zusammengesetzten Blockindex sowie den einzelnen Dimensionsblockindizes hinzufügt. Abb. 24 stellt die Schlüssel der Dimensionsblockindizes nach dem Hinzufügen von Block 48 dar.

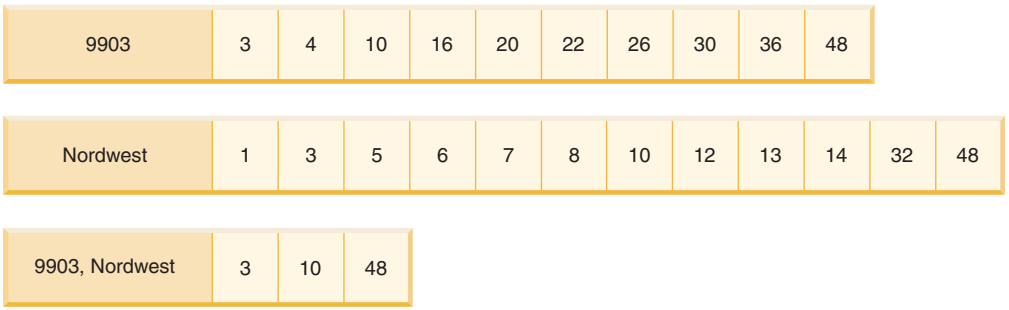


Abbildung 24. Schlüssel aus den Dimensionsblockindizes nach Hinzufügen von Block 48

Blockzuordnungen für MDC- und ITC-Tabellen

Wenn bei MDC-Tabellen ein Block geleert wird, wird seine Zuordnung zu den Werten der aktuellen logischen Zelle aufgehoben, indem seine Block-ID aus den Blockindizes entfernt wird. Der Block kann anschließend von einer anderen logischen Zelle wiederverwendet werden. Bei ITC-Tabellen sind alle Blöcke einer einzigen Zelle zugeordnet. Die Freigabe eines Blocks innerhalb einer Zelle bedeutet, dass er in einer nachfolgenden Einfügeoperation wiederverwendet werden kann. Diese Wiederverwendung verringert den Bedarf an neuen Blöcken zur Erweiterung der Tabelle.

Wenn ein neuer Block benötigt wird, müssen die zuvor geleerten Blöcke rasch gefunden werden können, ohne dass die Tabelle nach ihnen durchsucht werden muss.

Die Blockzuordnung ist eine Struktur, die eine Lokalisierung leerer Blöcke in der MDC- oder ITC-Tabelle vereinfacht. Die Blockzuordnung wird als separates Objekt gespeichert:

- In SMS-Tabellenbereichen als separate .BKM-Datei
- In DMS-Tabellenbereichen als neuer Objektdeskriptor in der Objekttabelle

Die Blockzuordnung ist eine Feldgruppe (Array), die für jeden Block der Tabelle einen Eintrag enthält. Jeder Eintrag besteht aus einem Satz von Statusbit für einen Block.

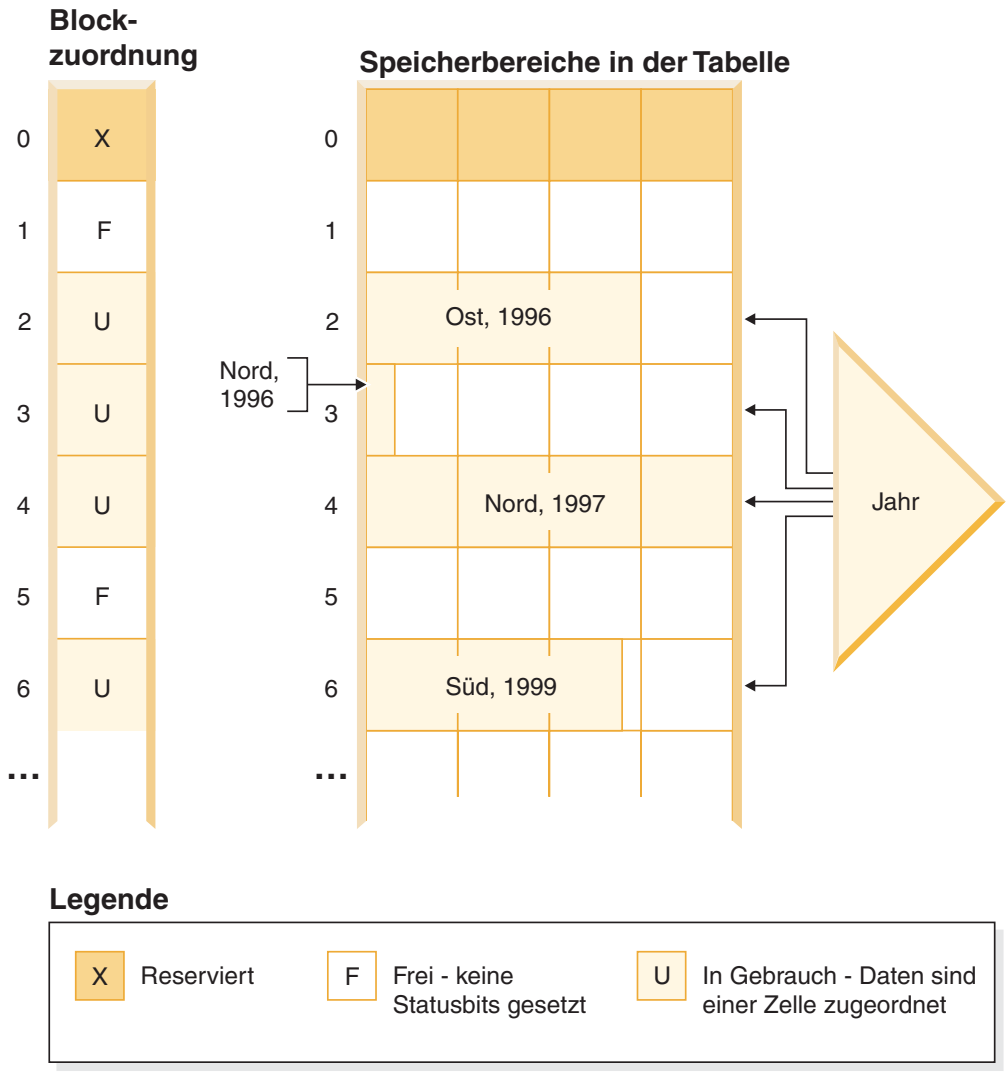


Abbildung 25. Funktionsweise einer Blockzuordnung

Die linke Seite in Abb. 25 zeigt die Blockzuordnungsfeldgruppe mit verschiedenen Einträgen für die Blöcke in der Tabelle. Die rechte Seite zeigt, wie die einzelnen EXTENTSIZE großen Speicherbereiche der Tabelle verwendet werden: einige sind frei, die meisten in Gebrauch. Datensätze werden nur in den Blöcken gefunden, die in der Blockzuordnung als in Gebrauch markiert sind. Aus Gründen der Übersichtlichkeit zeigt die Abbildung nur einen der beiden Dimensionsblockindizes.

Anmerkung:
1. Im Blockindex gibt es nur Zeiger auf Blöcke, die in der Blockzuordnung als „In Gebrauch" markiert sind.
2. Der erste Block ist reserviert. Dieser Block enthält Systemdatensätze für die Tabelle.

Freie Blöcke zur Verwendung in einer Zelle lassen sich durch eine Suche nach als „frei" markierten Blöcken (d. h. Blöcke, für die keine Bit gesetzt sind) in der Blockzuordnung leicht finden.

Tabellensuchen verwenden die Blockzuordnung ebenfalls, um nur auf Speicherbereiche zuzugreifen, die momentan Daten enthalten. Alle Speicherbereiche, die nicht in Gebrauch sind, brauchen in der Tabellensuche nicht berücksichtigt zu werden. Zur Veranschaulichung: Eine Tabellensuche würde in diesem Beispiel (Abb. 25 auf Seite 78) beim dritten Speicherbereich (Speicherbereich 2) in der Tabelle beginnen, indem sie den ersten reservierten Speicherbereich und den darauffolgenden leeren Speicherbereich überspringt, die Blöcke 2, 3 und 4 in der Tabelle durchsuchen, den nächsten Speicherbereich überspringen (ohne die Datenseiten dieses Speicherbereichs zu berühren) und schließlich die Suche von dort fortsetzen.

Löschen aus MDC- und ITC-Tabellen

Wenn ein Datensatz in einer MDC- oder ITC-Tabelle gelöscht wird, löscht der Datenbankmanager lediglich diesen Datensatz, sofern es sich nicht um den letzten Datensatz in einem Block handelt, und entfernt die Satz-ID (RID) aus allen datensatzbasierten Indizes, die für die Tabelle definiert sind.

Wenn eine Löschoperation den letzten Datensatz aus einem Block entfernt, gibt der Datenbankmanager den Block frei. Der Block wird freigegeben, indem das Statusbit IN_USE geändert wird und die Block-ID des Blocks aus allen Blockindizes entfernt wird. Falls datensatzbasierte Indizes vorhanden sind, wird die entsprechende Satz-ID natürlich auch aus diesen entfernt.

Anmerkung: Das heißt, Blockindexeinträge werden einmal für den ganzen Block entfernt, und dies nur, wenn der Block geleert wird, im Gegensatz zu Einträgen in datensatzbasierten Indizes, die einmal für die gelöschte Zeile entfernt werden müssen.

Aktualisierungen an MDC- und ITC-Tabellen

In einer MDC-Tabelle werden Aktualisierungen für Werte, die nicht zu einer Dimension gehören, ebenso wie bei regulären Tabellen an Ort und Stelle vorgenommen. Wenn sich die Aktualisierung eines Datensatzes in einer MDC- oder ITC-Tabelle auf die Länge des Datensatzes auswirkt und dieser nicht mehr auf die Seite passt, wird eine andere Seite mit ausreichend Speicherplatz gesucht.

Die Suche nach dieser neuen Seite beginnt im selben Block. Wenn in diesem Block kein geeigneter Speicherplatz vorhanden ist, wird der Algorithmus zum Einfügen eines neuen Datensatzes verwendet, um eine Seite in der logischen Zelle mit genügend Speicherplatz zu finden. Die Blockindizes müssen nur in dem Fall aktualisiert werden, dass kein Speicherplatz in der Zelle gefunden wird und der Zelle ein neuer Block hinzugefügt werden muss.

Wenn für eine ITC-Tabelle zu wenig Platz im Block vorhanden ist, um die aktualisierte Zeile unterzubringen, wird die Zeile in einen neuen Block verschoben. Durch dieses Verschieben wird die Zeile nicht mehr in Zeilen eingegliedert, die zeitgleich eingefügt wurden.

Aspekte nur zu MDC-Tabellen

Aktualisierungen von Dimensionswerten werden als Löschung des aktuellen Datensatzes mit anschließender Einfügung des geänderten Datensatzes behandelt, weil der Datensatz die logische Zelle wechselt, zu der er gehört. Wenn die Löschung des aktuellen Datensatzes dazu führt, dass ein Block vollständig geleert wird, muss der Blockindex aktualisiert werden. Analog muss der Blockindex aktualisiert werden, wenn die Einfügung des neuen Datensatzes einen neuen Block erforderlich macht.

MDC-Tabellen werden wie jede andere vorhandene Tabelle behandelt. Das bedeutet, dass für sie in gleicher Weise Trigger, referenzielle Integritätsbedingungen, Sichten und MQTs (MQT = Materialized Query Table, gespeicherte Abfragetabellen) definiert werden können.

Hinweise zu MDC- und ITC-Tabellen

Blockindizes müssen nur aktualisiert werden, wenn der erste Datensatz in einen Block eingefügt bzw. der letzte Datensatz aus einem Block gelöscht wird. Der Bedarf an Indexressourcen und Voraussetzungen für die Verwaltung und Protokollierung für Blockindizes ist daher wesentlich geringer als der Bedarf an Indexressourcen und Voraussetzungen, der mit regulären Indizes verbunden ist. Für jeden Blockindex, der ansonsten ein regulärer Index gewesen wäre, sind die Verwaltungs- und Protokollierungsressourcen sowie die entsprechenden Voraussetzungen beträchtlich reduziert.

Wenn Sie kürzlich geleerte Blöcke wiederverwenden, muss eine bedingte Sperre des Modus Z für den Block verwendet werden, um sicherzustellen, dass der Block gegenwärtig nicht von einer Suchfunktion für nicht festgeschriebene Lesevorgänge durchsucht wird.

Verwaltung von Speicherbereichen bei MDC- und ITC-Tabellen

Die Freigabe von Datenspeicherbereichen in der MDC-Tabelle (MDC = Multidimensional Clustering, Mehrdimensionales Clustering) oder der ITC-Tabelle (ITC = Insert Time Clustering, Clustering anhand der Einfügungszeit) erfolgt durch die Reorganisation der Tabelle.

In einer MDC- und ITC-Tabelle dient die Blockzuordnung zur Verfolgung aller zur Tabelle gehörenden Datenspeicherbereiche; die Blockzuordnung zeigt, welche Blöcke und Speicherbereiche Daten enthalten und welche nicht. Blöcke mit Daten werden mit „in Gebrauch" gekennzeichnet. Nach einem Löschvorgang für MDC- oder ITC-Tabellen oder einem Rollout für MDC-Tabellen werden die entsprechenden Blockeinträge in der Blockzuordnung nicht mehr mit „in Gebrauch" markiert, sondern für die Wiederverwendung durch die Tabelle freigegeben.

Diese Blöcke und Speicherbereiche können aber nicht von anderen Objekten im Tabellenbereich verwendet werden. Sie können diese freien Datenspeicherbereiche in der Tabelle durch die Reorganisation der Tabelle freigeben. Sie können den Befehl **REORG TABLE** mit dem Parameter **RECLAIM EXTENTS** verwenden, sodass die Tabelle für Ihre Benutzer verfügbar und online ist, während der Speicherbereich frei-

gegeben wird. Das Freigeben von Speicherbereichen in der MDC- oder ITC-Tabelle wird nur für Tabellen in DMS-Tabellenbereichen unterstützt.

Der Befehl **REORG TABLE** verwendet den Parameter **RECLAIM EXTENTS**, um Speicherbereiche aus der exklusiven Nutzung durch die MDC- oder ITC-Tabelle freizugeben und stellt den freien Speicher zur Nutzung durch andere Datenbankobjekte innerhalb des Tabellenbereichs zur Verfügung.

Die Option gibt Ihnen außerdem die Möglichkeit, den gleichzeitigen Zugriff auf die MDC- oder ITC-Tabelle zu steuern, während die Speicherbereiche freigegeben werden. Standardmäßig ist 'Schreibzugriff' eingestellt. Weitere Möglichkeiten zur Steuerung des gleichzeitigen Zugriffs sind 'Lesezugriff' und 'Kein Zugriff'.

Wenn die MDC- oder ITC-Tabelle auch bereichs- oder datenbankpartitioniert ist, erfolgt die Freigabe von Speicherbereichen standardmäßig für alle Daten- oder Datenbankpartitionen. Sie können den Befehl ausführen, um Speicherbereiche für nur eine bestimmte Partition freizugeben, indem Sie einen Partitionsnamen (für Datenpartitionen) oder eine Partitionsnummer (für Datenbankpartitionen) angeben.

Zum Freigeben von Speicherbereichen können der Befehl **REORG TABLE** und die API 'db2Reorg' verwendet werden.

Es steht eine automatische Unterstützungsfunktion zur Verfügung, mit der Sie die Freigabe von Speicherbereichen in die automatischen Verwaltungsaktivitäten für die Datenbank integrieren können. Zur Aktivierung einer Reorganisation zur Freigabe von Speicherbereichen in einer MDC- oder ITC-Tabelle müssen die Datenbankkonfigurationsparameter **auto_maint**, **auto_tbl_maint** und **auto_reorg** alle auf ON eingestellt sein. Die Konfiguration dieser Datenbankkonfigurationsparameter kann über die Befehlszeile vorgenommen werden. Bei einer DB2-Instanz, für die die Datenbankpartitionierungsfunktion aktiviert ist, muss die Konfiguration der Parameter in der Katalogpartition ausgeführt werden.

Durch eine Verwaltungsrichtlinie wird gesteuert, wann eine automatische Reorganisation einer MDC- oder ITC-Tabelle erfolgt, um nicht genutzte Speicherbereiche freizugeben. Die gespeicherten DB2-Systemprozeduren AUTOMAINT_SET_POLICY und AUTOMAINT_SET_POLICYFILE werden zum Einstellen dieser Verwaltungsrichtlinie verwendet. Zum Speichern der Richtlinie für die automatisierte Verwaltung wird XML verwendet.

Tabellenpartitionierung und MDC-Tabellen

In einer Tabelle, die sowohl MDC- als auch Datenpartitionstabelle ist, können Spalten in der Bereichspartitionsspezifikation (Range-Partition-Spec) und auch im MDC-Schlüssel für die Tabellenpartitionierung verwendet werden. Eine Tabelle, die diese beiden Merkmale aufweist, ermöglicht eine exaktere Differenzierung des Datenpartitions- und Blockausschlusses, als dies bei Verwendung nur einer dieser Funktionalitäten der Fall wäre.

Es gibt auch zahlreiche Anwendungen, in denen es sinnvoll ist, für den MDC-Schlüssel andere Spalten als die für die Tabellenpartitionierung genutzten Spalten anzugeben. Hierbei ist zu beachten, dass die Tabellenpartitionierung mit mehreren Spalten arbeitet, während beim MDC mit mehreren Dimensionen gearbeitet wird.

Merkmale eines normalen DB2 Data Warehouse

Die folgenden Empfehlungen beziehen sich auf typische, normale Data Warehouses, die in DB2 Version 9.1 neu implementiert wurden. Die folgenden Merkmale werden angenommen:

- Die Datenbank arbeitet auf mehreren Systemen oder in mehreren logischen AIX-Partitionen.
- Umgebungen mit partitionierten Datenbanken werden verwendet (Tabellen werden mithilfe der Klausel DISTRIBUTE BY HASH erstellt).
- Es wurden zwischen vier und 50 Datenpartitionen definiert.
- Die Tabelle, für die die Möglichkeit zur Verwendung des MDC und der Tabellenpartitionierung geprüft wird, ist eine der zentralen Fakttabellen.
- Die Tabelle umfasst zwischen 100.000.000 und 100.000.000.000 Zeilen.
- Neue Daten werden in unterschiedlichen Zeitrahmen geladen: Jeweils über Nacht, wöchentlich, monatlich.
- Das tägliche Aufnahmevolumen liegt zwischen 10.000 und 10 Millionen Datensätzen.
- Die Datenvolumen schwanken: Hierbei liegt das Volumen des Spitzenmonats um das Fünffache höher als das des Monats mit dem geringsten Datenaufkommen. Entsprechend beläuft sich auch der Umfang der größten Dimensionen (Produktlinie, Bereich) auf das Fünffache des Umfangs der kleinsten Dimensionen.
- Die Datenbank enthält detaillierte Datenbestände der letzten 1 - 5 Jahre.
- Abgelaufene Daten werden in monatlichen oder vierteljährlichen Abständen mit einer Rollout-Operation ausgelagert.
- Tabellen verwenden eine Vielzahl von Abfragetypen. Die Workload besteht jedoch größtenteils aus analytischen Abfragen, die in Bezug auf OLTP-Workloads die folgenden Merkmale aufweisen:
  - Es gibt umfangreichere Ergebnismengen mit bis zu 2 Millionen Zeilen.
  - Die Mehrzahl oder alle Abfragen beziehen sich auf Sichten und nicht auf Basistabellen.
- SQL-Klauseln zur Auswahl von Daten nach Bereichen (Klausel BETWEEN), Elementen in Listen etc.

Merkmale einer Fakttabelle eines normalen DB2 Data Warehouse der Version 9.1

Eine normale Data Warehouse-Fakttabelle kann z. B. das folgende Design verwenden:

- Erstellung von Datenpartitionen über die Spalte 'Month'.
- Definition einer Datenpartition für jeden Rollout-Zeitraum, z. B. 1 Monat, 3 Monate.
- Erstellung von MDC-Dimensionen für 'Day' und für 1 - 4 weitere Dimensionen. Typische Dimensionen sind: Produktlinie und Bereich.
- Alle Datenpartitionen und MDC-Cluster sind über alle Datenpartitionen verteilt.

Das MDC und die Tabellenpartitionierung bieten teilweise dieselben Vorteile. Die folgende Tabelle enthält mögliche Anforderungen innerhalb Ihres Unternehmens und Empfehlungen zu einem Organisationsschema, die auf der Basis der zuvor festgestellten Merkmale gegeben werden.

Tabelle 7. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Empfehlung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Tabellenpartitionierung und MDC	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines traditionellen Offlinezeitfensters	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines Mikro-Offlinezeitfensters (weniger als 1 Minute)	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums bei gleichzeitiger Erhaltung der Tabellenverfügbarkeit für Geschäftsbenutzer, die Abfragen ohne Serviceverluste weiterhin übergeben können	MDC	MDC kann nicht alle in diesem Zusammenhang geltenden Anforderungen voll erfüllen. Die Tabellenpartitionierung eignet sich hier nicht, da die Tabelle nur für kurze Zeit in den Offlinemodus versetzt wird.
Tägliches Laden von Daten (Befehl **LOAD** oder **INGEST**)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Kontinuierliches Laden von Daten (Befehl **LOAD** mit ALLOW READ ACCESS oder Befehl **INGEST**)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Leistung bei der Ausführung von Abfragen für traditionelle BI-Abfragen	Tabellenpartitionierung und MDC	MDC eignet sich besonders gut für Abfragen in Datenkuben und mehreren Dimensionen. Die Tabellenpartitionierung bietet Unterstützung über den Partitionsausschluss.

Tabelle 7. Verwendung der Tabellenpartitionierung mit MDC-Tabellen (Forts.)

Problemstellung	Empfohlenes Schema	Empfehlung
Minimierung des Reorganisationsaufwandes durch Vermeidung des Reorganisationsbedarfs oder Reduzierung des Aufwands für die Ausführung der Task	MDC	Beim MDC wird das Clustering beibehalten, sodass sich die Notwendigkeit von Reorganisationen verringert. Wenn das MDC verwendet wird, dann bietet die Datenpartitionierung auch in Teilbereichen keine Vorteile. Allerdings ermöglicht die Tabellenpartitionierung bei Nichtverwendung des MDC die Reduzierung des Reorganisationsbedarfs, indem auf Partitionsebene ein etwas groberes Clustering beibehalten wird.

Beispiel 1:

Sie arbeiten mit einer Tabelle mit den Schlüsselspalten 'YearAndMonth' und 'Province'. Bei der Planung dieser Tabelle wäre eine Partitionierung nach Datum mit einer Zeitspanne von zwei Monaten pro Datenpartition sinnvoll. Darüber hinaus können Sie die Daten auch nach der Spalte 'Province' organisieren, sodass alle Zeilen für ein bestimmtes Gebiet innerhalb eines Datumsbereichs von zwei Monaten in einer Gruppe zusammengefasst werden. Diese Vorgehensweise ist in Abb. 6 auf Seite 37 dargestellt.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

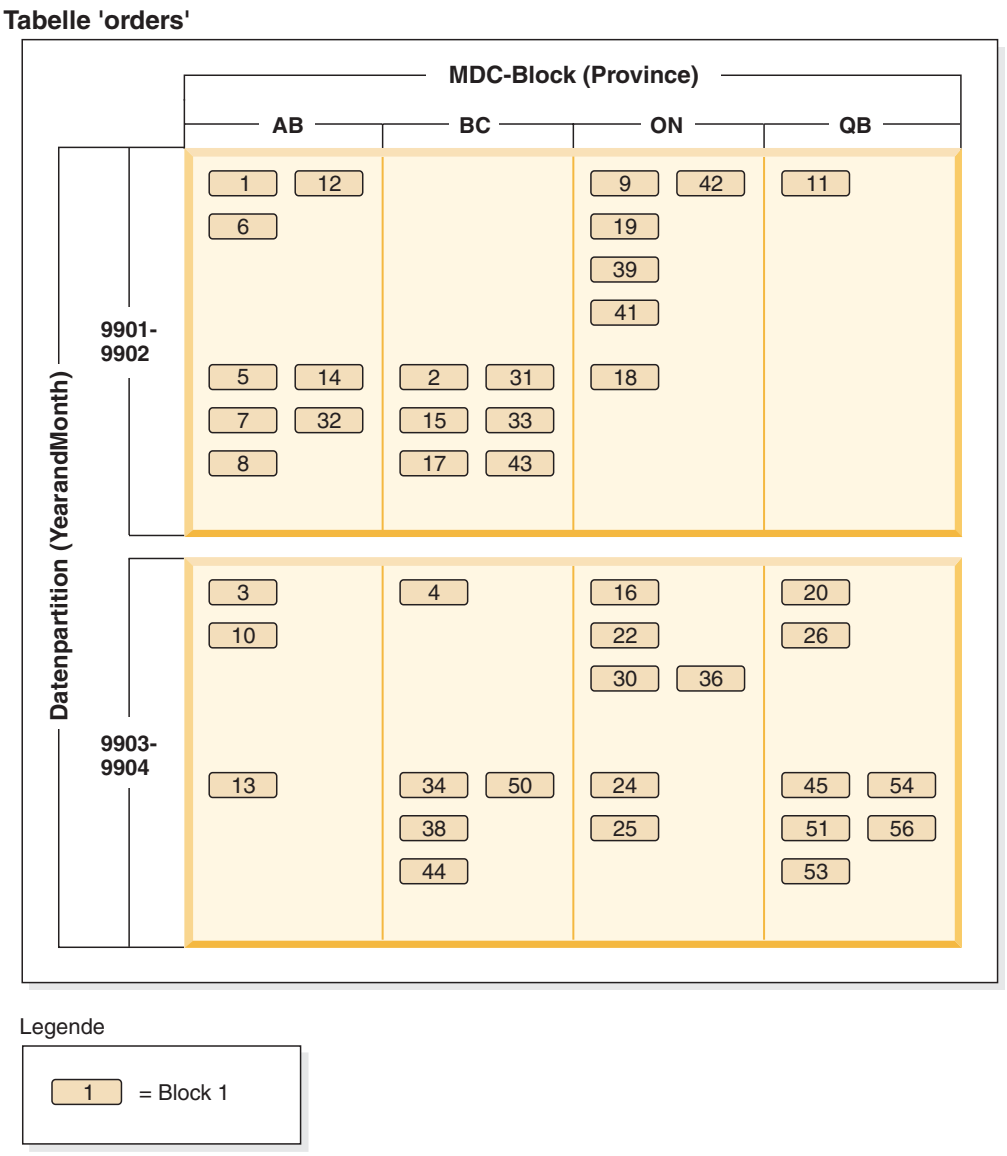


Abbildung 26. Eine nach 'YearAndMonth' partitionierte und nach 'Province' organisierte Tabelle

Beispiel 2:

Eine exaktere Differenzierung kann durch Hinzufügen von 'YearAndMonth' zur Klausel ORGANIZE BY DIMENSIONS (siehe hierzu Abb. 7 auf Seite 38) erzielt werden.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

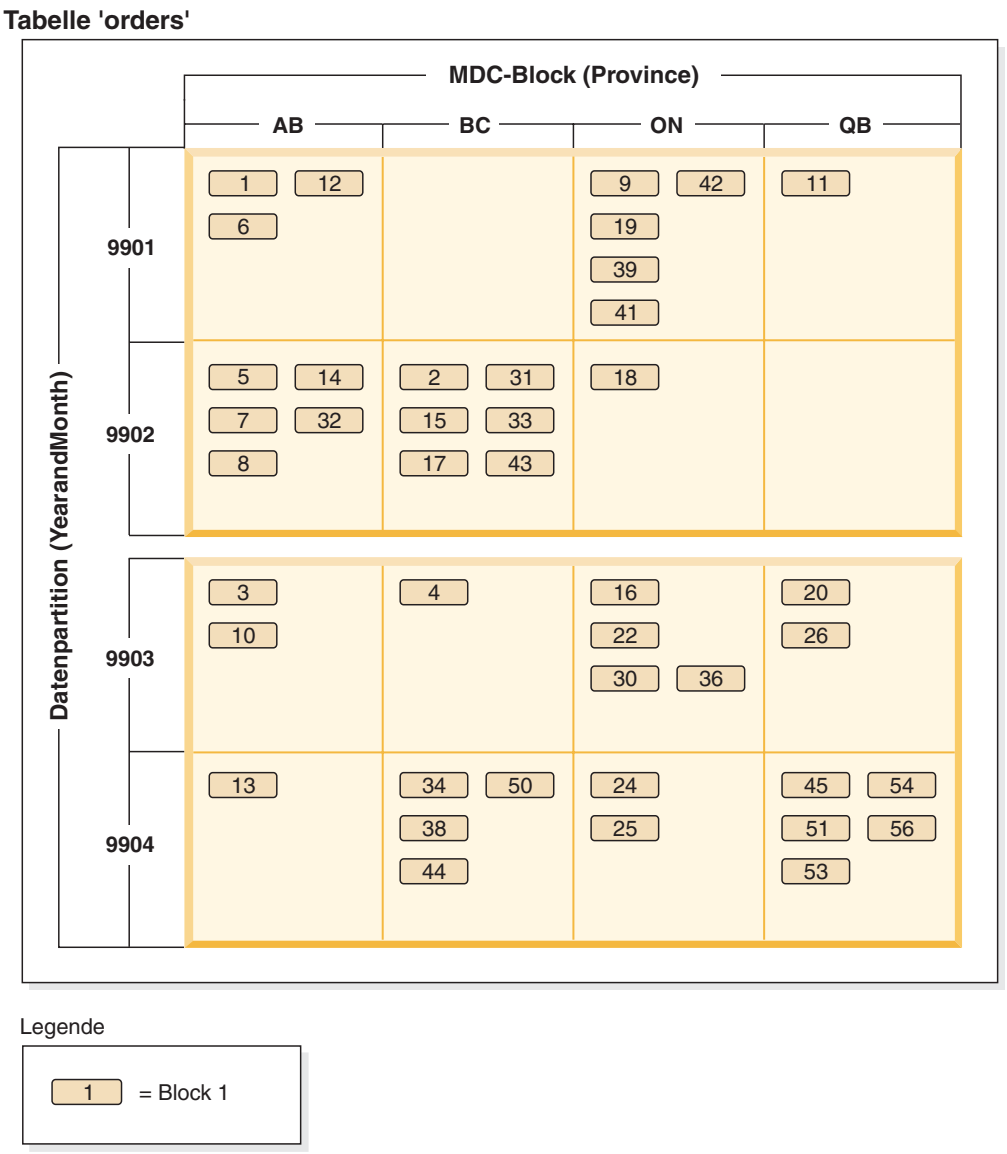


Abbildung 27. Eine nach 'YearAndMonth' partitionierte und nach 'Province' und 'YearAndMonth' organisierte Tabelle

Wenn die Partitionierung so festgelegt wurde, dass jeder Bereich nur einen einzigen Wert enthält, ergeben sich keine Vorteile, wenn die Tabellenpartitionierungsspalte in den MDC-Schlüssel aufgenommen wird.

Wichtige Hinweise

- Im Vergleich mit einer Basistabelle benötigen sowohl MDC-Tabellen als auch partitionierte Tabellen mehr Speicherplatz. Diese Speicherplatzanforderungen gelten zusätzlich zu den sonstigen Anforderungen, sind jedoch unter Berücksichtigung der sich daraus ergebenden Vorteile sinnvoll.
- Wenn Sie die Tabellenpartitionierung und die MDC-Funktionalität in Ihrer partitionierten Datenbankumgebung nicht zusammen einsetzen wollen, dann sollten Sie die Tabellenpartitionierung in den Fällen einsetzen, in denen die Datenverteilung verlässlich vorausgesagt werden kann. Dies ist normalerweise bei den hier erläuterten Systemtypen der Fall. Andernfalls sollten Sie die Verwendung von MDC in Betracht ziehen.

- Bei einer MDC-Datenpartitionstabelle, die mit DB2 Version 9.7 Fixpack 1 (oder einem neueren Release) erstellt wurde, sind die MDC-Blockindizes für die Tabelle partitioniert. Bei einer MDC-Datenpartitionstabelle, die mit DB2 V9.7 (oder einem früheren Release) erstellt wurde, sind die MDC-Blockindizes für die Tabelle nicht partitioniert.

Kapitel 4. Parallele Datenbanksysteme

Parallelität

Komponenten einer Task, wie zum Beispiel einer Datenbankabfrage, können parallel ausgeführt werden, um die Leistung erheblich zu erhöhen. Die Art der Task, die Datenbankkonfiguration und die Hardwareumgebung bestimmen, wie das DB2-Datenbankprodukt eine Task parallel ausführt.

Diese Faktoren stehen in Wechselbeziehung zueinander und sollten bei der Arbeit am physischen und logischen Entwurf einer Datenbank im Zusammenhang betrachtet werden. Das DB2-Datenbanksystem unterstützt die folgenden Parallelitätstypen:

- Ein-/Ausgabeparallelität
- Abfrageparallelität
- Dienstprogrammparallelität

Ein-/Ausgabeparallelität

Wenn mehrere Container für einen Tabellenbereich vorhanden sind, kann der Datenbankmanager die *parallele Ein-/Ausgabe* nutzen. Parallele E/A bezeichnet den Vorgang, bei dem Lese- bzw. Schreiboperationen mit zwei oder mehr Ein-/Ausgabeeinheiten gleichzeitig ausgeführt werden. Auf diese Weise kann der Durchsatz deutlich verbessert werden.

Abfrageparallelität

Es gibt zwei Arten der Abfrageparallelität: abfrageübergreifende und abfrageinterne Parallelität.

Abfrageübergreifende Parallelität bezeichnet die Fähigkeit einer Datenbank, Abfragen von mehreren Anwendungen gleichzeitig zu akzeptieren. Jede Abfrage wird unabhängig von den anderen ausgeführt, der Datenbankmanager führt jedoch alle Abfragen gleichzeitig aus. DB2-Datenbankprodukte haben diese Art der Parallelität schon immer unterstützt.

Abfrageinterne Parallelität bezeichnet die gleichzeitige Verarbeitung von Teilen einer einzelnen Abfrage mithilfe der partitionsinternen und/oder der partitionsübergreifenden Parallelität.

Partitionsinterne Parallelität

Der Begriff *partitionsinterne Parallelität* bezeichnet die Fähigkeit, eine Abfrage in mehrere Teile zu untergliedern. Einige DB2-Dienstprogramme arbeiten ebenfalls mit dieser Art der Parallelität.

Bei der partitionsinternen Parallelität wird das, was im Allgemeinen als eine einzige Datenbankoperation betrachtet wird (z. B. die Indexerstellung, das Laden von Daten oder SQL-Abfragen) in mehrere Teile unterteilt, von denen viele oder alle parallel innerhalb einer einzigen Datenbankpartition ausgeführt werden können.

Abb. 28 zeigt eine Abfrage, die in drei Teile aufgeteilt ist, die parallel ausgeführt werden können, wobei die Ergebnisse schneller geliefert werden als bei serieller Ausführung der Abfrage. Die Teile sind jeweils Kopien voneinander. Zur Nutzung der partitionsinternen Parallelität muss die Datenbank entsprechend konfiguriert werden. Sie können den Grad der Parallelität selbst auswählen oder ihn vom System festlegen lassen. Der Grad der Parallelität stellt die Anzahl der Teile einer Abfrage dar, die parallel ausgeführt werden.

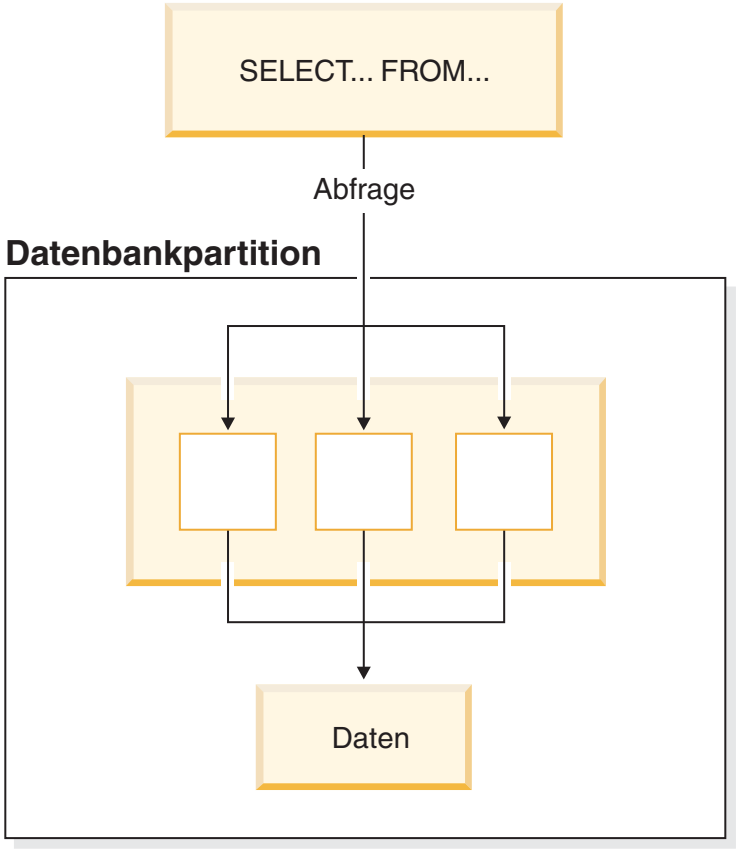


Abbildung 28. Partitionsinterne Parallelität

Partitionsübergreifende Parallelität

Der Begriff *partitionsübergreifende Parallelität* bezeichnet die Fähigkeit, eine Abfrage in mehreren Teilen über mehrere Partitionen einer partitionierten Datenbank auf einer oder mehreren Maschinen zu verteilen. Die Abfrage wird parallel ausgeführt. Einige DB2-Dienstprogramme arbeiten ebenfalls mit dieser Art der Parallelität.

Bei der partitionsübergreifenden Parallelität wird das, was im Allgemeinen als eine einzige Datenbankoperation betrachtet wird (z. B. die Indexerstellung, das Laden von Daten oder SQL-Abfragen), in mehrere Teile unterteilt, von denen viele oder alle parallel über mehrere Partitionen einer partitionierten Datenbank hinweg auf einer oder mehreren Maschinen ausgeführt werden können.

Abb. 29 auf Seite 91 zeigt eine Abfrage, die in drei Teile aufgeteilt ist, die parallel ausgeführt werden können, wobei die Ergebnisse schneller zurückgeliefert werden als bei serieller Ausführung in einer einzigen Datenbankpartition.

Der Grad der Parallelität wird im Wesentlichen durch die Anzahl der Datenbankpartitionen, die Sie erstellen, und die Art und Weise der Definition der Datenbankpartitionsgruppen bestimmt.

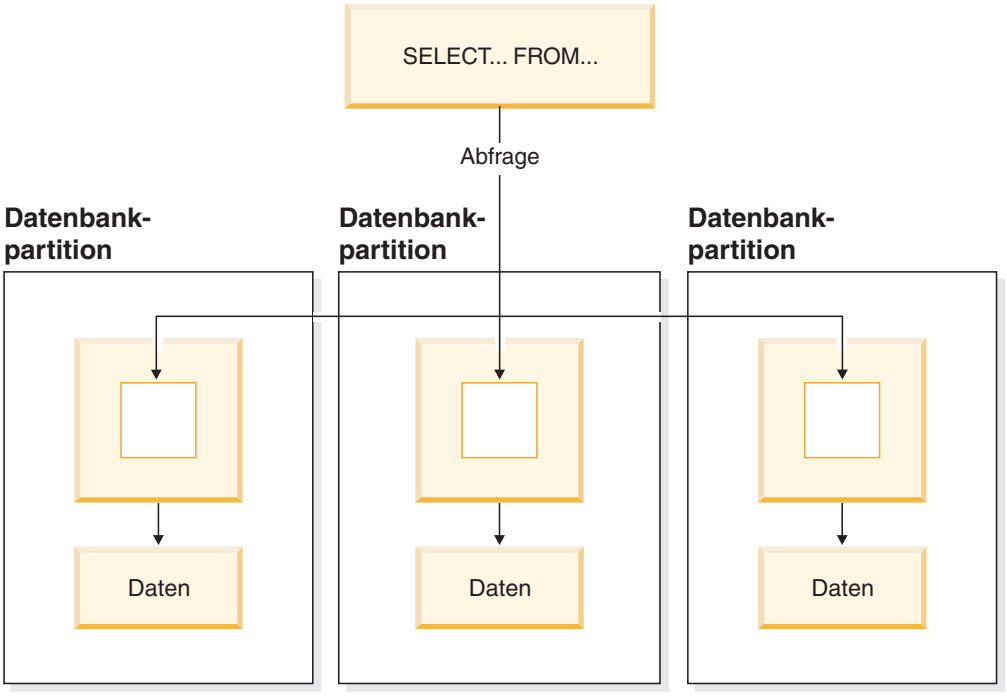


Abbildung 29. Partitionsübergreifende Parallelität

Simultane partitionsinterne und partitionsübergreifende Parallelität

Sie können die partitionsinterne Parallelität und die partitionsübergreifende Parallelität gleichzeitig nutzen. Diese Kombination bietet zwei Dimensionen der Parallelität, wodurch sich ein weiterer wesentlicher Anstieg der Verarbeitungsgeschwindigkeit für Abfragen ergibt.

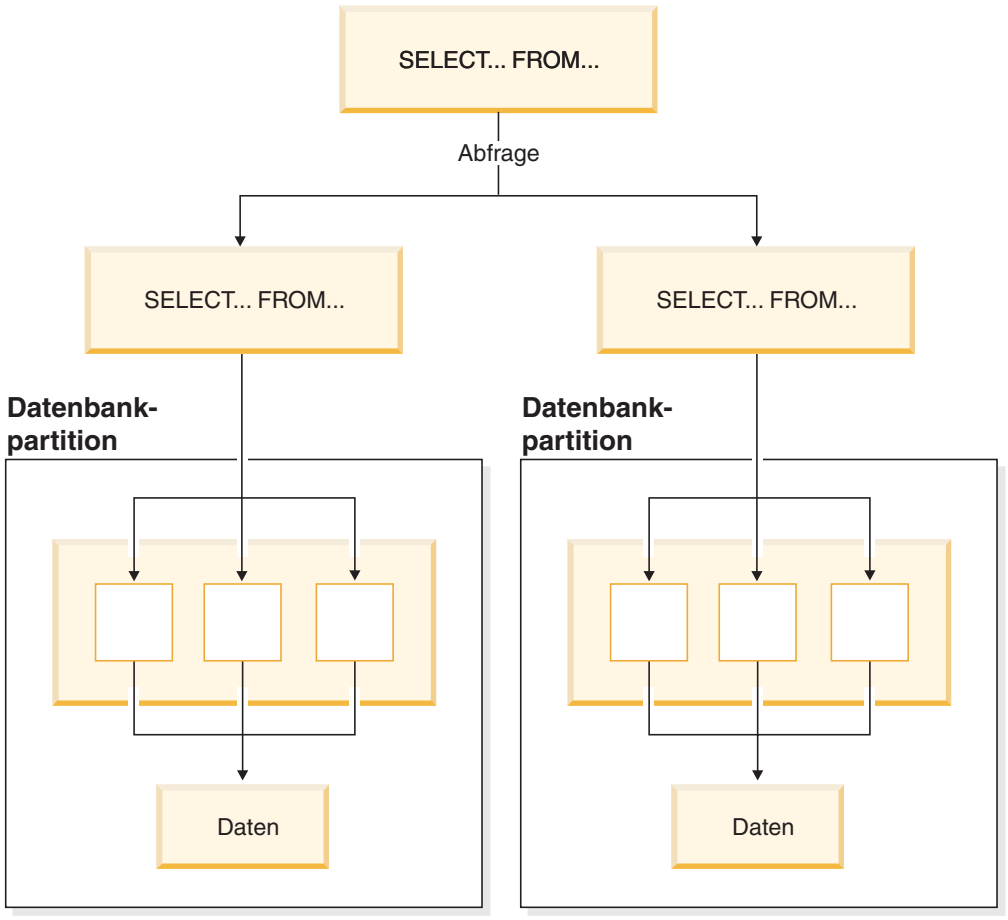


Abbildung 30. Simultane partitionsinterne und partitionsübergreifende Parallelität

Dienstprogrammparallelität

DB2-Dienstprogramme können die Vorteile der partitionsinternen Parallelität nutzen. Sie können außerdem die Vorteile der partitionsübergreifenden Parallelität nutzen. Wenn mehrere Datenbankpartitionen vorhanden sind, werden die Dienstprogramme in den einzelnen Datenbankpartitionen parallel ausgeführt.

Das Dienstprogramm LOAD kann die Vorteile der partitionsinternen Parallelität und der E/A-Parallelität nutzen. Das Laden von Daten in eine Tabelle ist eine CPU-intensive Operation. Das Dienstprogramm LOAD nutzt die Möglichkeit mehrerer Prozessoren bei Operationen wie dem Analysieren und Formatieren von Daten. Darüber hinaus kann es parallele E/A-Server zum gleichzeitigen Schreiben der Daten in Container verwenden.

In einer Umgebung mit partitionierten Datenbanken nutzt der Befehl **LOAD** die Vorteile der partitionsinternen, partitionsübergreifenden und der E/A-Parallelität durch parallele Aufrufe in allen Datenbankpartitionen, in denen sich die Tabelle befindet.

Während der Indexerstellung erfolgt das Durchsuchen und das nachfolgende Sortieren der Daten parallel. Das DB2-System nutzt sowohl die E/A-Parallelität als auch die partitionsinterne Parallelität bei der Erstellung eines Index. Dadurch wird

die Indexerstellung bei der Verarbeitung der Anweisung CREATE INDEX, beim Neustart (wenn ein Index als ungültig markiert ist) und bei der Reorganisation von Daten beschleunigt.

Das Backup und der Restore von Daten sind Operationen, die wesentlich von der E/A-Leistung abhängig sind. Das DB2-System nutzt sowohl die E/A-Parallelität als auch partitionsinterne Parallelität bei der Durchführung von Backup- und Restore-Operationen. Der Befehl BACKUP nutzt die E/A-Parallelität, indem er von mehreren Tabellenbereichscontainern parallel liest und asynchron auf mehrere Backup-Medien parallel schreibt.

Umgebungen mit partitionierten Datenbanken

Eine Umgebung mit partitionierten Datenbanken ist eine Datenbankinstallation, die die Verteilung von Daten für Datenbankpartitionen unterstützt.

- Eine *Datenbankpartition* ist ein Teil einer Datenbank, der aus seinen eigenen Daten, Indizes, Konfigurationsdateien und Transaktionsprotokollen besteht. Eine Umgebung mit partitionierten Datenbanken ist eine Datenbankinstallation, die die Verteilung von Daten für Datenbankpartitionen unterstützt.
- Eine *Einzelpartitionsdatenbank* ist eine Datenbank, die nur über eine Datenbankpartition verfügt. Sämtliche Daten in der Datenbank werden in dieser einen Datenbankpartition gespeichert. In diesem Fall bieten Datenbankpartitionsgruppen, wenn sie vorhanden sind, keine zusätzlichen Leistungsvorteile.
- Eine *Mehrpartitionsdatenbank* ist eine Datenbank mit zwei oder mehr Datenbankpartitionen. Die Tabellen können in einer oder mehreren Datenbankpartitionen gespeichert werden. Wenn eine Tabelle in einer Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen gespeichert ist, heißt dies, dass einige ihrer Zeilen in einer Datenbankpartition gespeichert werden, während sich andere Zeilen in anderen Datenbankpartitionen befinden.

In der Regel befindet sich auf jedem physischen System nur eine Datenbankpartition. Die Prozessoren des Systems werden vom Datenbankmanager dazu verwendet, den in der jeweiligen Datenbankpartition gespeicherten Teil der Gesamtdaten der Datenbank zu verwalten.

Da die Daten über Partitionen verteilt sind, können Sie das Potenzial mehrerer Prozessoren auf mehreren physischen Systemen zur Erfüllung von Informationsanforderungen nutzen. Anforderungen zur Abfrage und Aktualisierung von Daten werden automatisch in Unteranforderungen zerlegt und in den betroffenen Datenbankpartitionen parallel ausgeführt. Die Tatsache, dass Datenbanken auf Datenbankpartitionen verteilt sind, ist für Benutzer, die SQL-Anweisungen ausführen, transparent.

Die Benutzerinteraktion erfolgt über eine einzige Datenbankpartition, die als *Koordinatorpartition* für den jeweiligen Benutzer bezeichnet wird. Die Koordinatorpartition ist in derselben Datenbankpartition aktiv wie die Anwendung bzw., im Fall einer fernen Anwendung, in der Datenbankpartition, mit der die Anwendung verbunden ist. Alle Datenbankpartitionen können als Koordinatorpartition verwendet werden.

Der Datenbankmanager ermöglicht es, Daten über verschiedene Datenbankpartitionen in der Datenbank verteilt speichern. Dies bedeutet, dass die Daten zwar physisch über mehr als eine Datenbankpartition verteilt werden, jedoch so auf sie zugegriffen werden kann, als ob sie sich an derselben Speicherposition befänden.

Anwendungen und Benutzer, die auf Daten in einer Mehrpartitionsdatenbank zugreifen, kennen die physische Speicherposition der Daten nicht.

Die Daten werden trotz der physischen Trennung wie eine logische Einheit verwendet und verwaltet. Benutzer können wählen, wie ihre Daten verteilt werden, indem Sie Verteilungsschlüssel deklarieren. Darüber hinaus können Benutzer bestimmen, auf welche und auf wie viele Datenbankpartitionen ihre Daten verteilt werden, indem sie den Tabellenbereich und die zugeordnete Datenbankpartitionsgruppe auswählen, in der die Daten gespeichert werden sollen. Empfehlungen zur Verteilung und Replikation können mithilfe des DB2-Designadvisor ermittelt werden. Weiterhin wird eine aktualisierbare Verteilungszuordnung zusammen mit einem Hashalgorithmus verwendet, um die Zuordnung von Werten der Verteilungsschlüssel zu Datenbankpartitionen anzugeben, wodurch das Platzieren und Abrufen der einzelnen Datenzeilen festgelegt wird. Dadurch können Sie die Auslastung für große Tabellen über eine Mehrpartitionsdatenbank hinweg verteilen und kleinere Tabellen in einer oder mehreren Datenbankpartitionen speichern. Jede Datenbankpartition verfügt über lokale Indizes für die in ihr gespeicherten Daten, sodass für den lokalen Datenzugriff eine erhöhte Leistung erzielt wird.

Anmerkung: Sie sind nicht darauf beschränkt, alle Tabellen über alle Datenbankpartitionen in der Datenbank verteilt zu speichern. Der Datenbankmanager unterstützt eine *Teilentclusterung*, das heißt, Sie können Tabellen und die zugehörigen Tabellenbereiche auf eine Untergruppe der Datenbankpartitionen im System verteilen.

Eine alternative Methode, die Sie in Erwägung ziehen können, wenn Sie möchten, dass Tabellen in den einzelnen Datenbankpartitionen angelegt werden, ist die Verwendung und Replikation von MQTs (MQT = Materialized Query Table, gespeicherte Abfragetabelle). Sie können eine MQT mit den von Ihnen benötigten Informationen erstellen und diese dann auf jede Datenbankpartition replizieren.

Eine Nichtrootinstallation eines DB2-Datenbankprodukts unterstützt die Datenbankpartitionierung nicht. Die Datei `db2nodes.cfg` sollte nicht manuell aktualisiert werden. Manuelles Aktualisieren verursacht einen Fehler (SQL6031N).

Datenbankpartitions- und Prozessorumgebungen

Dieser Abschnitt enthält eine Übersicht über die Konfiguration von Einzeldatenbankpartitionen und Datenbanken mit mehreren Partitionen. Zu den erstgenannten gehören Konfigurationen mit Einzelprozessormaschinen und Multiprozessormaschinen (SMP), zu den anderen Datenbankpartitionen mit einem Prozessor (MPP) oder mehreren Prozessoren (Cluster aus SMP-Systemen) und logische Datenbankpartitionen.

Kapazität bezieht sich hier auf die Anzahl der Benutzer und Anwendungen, die auf die Datenbank zugreifen können. Dies wird großenteils durch Hauptspeicher, Agenten, Sperren, E/A-Operationen und Speicherverwaltung bestimmt. *Skalierbarkeit* bezeichnet die Fähigkeit einer Datenbank, bei zunehmender Größe weiterhin die gleichen Betriebsmerkmale und Antwortzeiten zu zeigen. Für jede Umgebung werden die Kapazität und Skalierbarkeit behandelt.

Einzeldatenbankpartition auf Einzelprozessormaschine

Diese Umgebung verfügt über Hauptspeicher und Platte, enthält aber nur eine CPU (siehe Abb. 31 auf Seite 95). Die Datenbank in dieser Umgebung erfüllt die Anforderungen einer Abteilung oder eines kleinen Büros, wobei die Daten und

Systemressourcen (einschließlich des Einzelprozessors bzw. einer CPU) von einem einzelnen Datenbankmanager verwaltet werden.

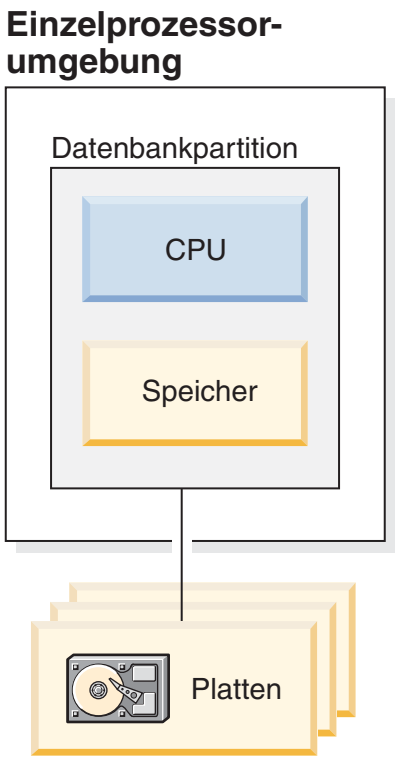


Abbildung 31. Einzeldatenbankpartition auf Einzelprozessormaschine

Kapazität und Skalierbarkeit

In dieser Umgebung können Sie weitere Platten hinzufügen. Durch den Einsatz eines oder mehrerer E/A-Server für jede Platte kann mehr als eine E/A-Operation gleichzeitig stattfinden.

Ein Einzelprozessorsystem ist durch die Menge an Plattenspeicherplatz, den der Prozessor handhaben kann, eingeschränkt. Ungeachtet aller zusätzlichen Komponenten wie Arbeitsspeicher oder Festplattenspeicher, die Sie zur schnelleren Verarbeitung von Benutzeranforderungen möglicherweise hinzufügen, kann eine einzelne CPU bei weiterer Zunahme der Auslastung Benutzeranforderungen nicht mehr schneller verarbeiten. Wenn Sie das Maximum an Kapazität und Skalierbarkeit erreicht haben, können Sie die Umrüstung auf ein System mit einer Einzeldatenbankpartition und mehreren Prozessoren in Erwägung ziehen.

Einzeldatenpartition auf Multiprozessormaschine

Diese Umgebung besteht gewöhnlich aus mehreren gleich starken Prozessoren innerhalb derselben Maschine (siehe Abb. 32 auf Seite 96) und wird als *symmetrisches Multiprozessorsystem (SMP-System)* bezeichnet. (SMP steht für Symmetric Multiprocessor). Ressourcen wie Plattenspeicher und Hauptspeicher werden gemeinsam genutzt.

Wenn mehrere Prozessoren verfügbar sind, können verschiedene Datenbankoperationen schneller durchgeführt werden. DB2-Datenbanksysteme können auch die Arbeit einer einzigen Abfrage auf die verfügbaren Prozessoren verteilen, um die

Verarbeitungsgeschwindigkeit zu erhöhen. Andere Datenbankoperationen wie das Laden von Daten, das Backup und der Restore von Tabellenbereichen sowie die Erstellung von Indizes für vorhandene Daten können die Verfügbarkeit mehrerer Prozessoren ausnutzen.

Symmetrische Multiprozessorumgebung (SMP)

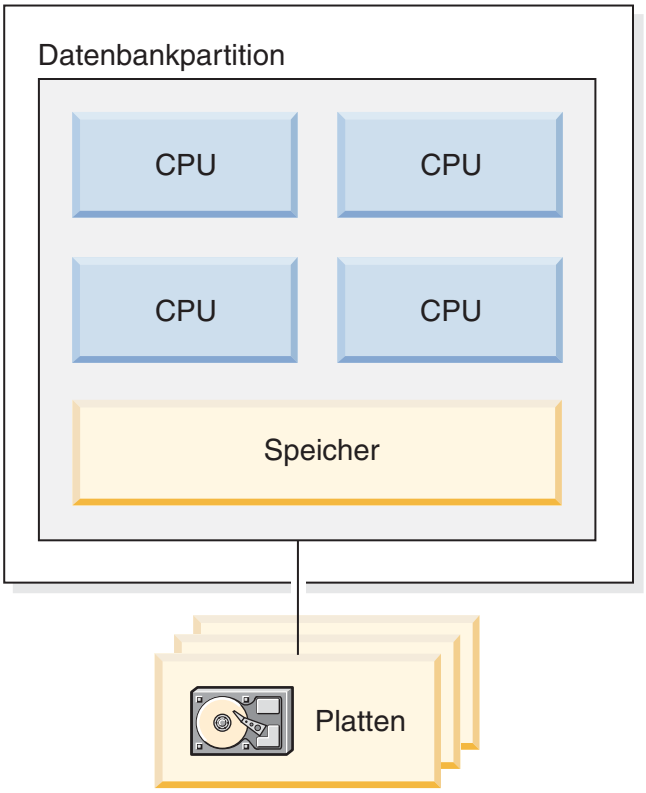


Abbildung 32. Einzelpartitionsdatenbank in einer symmetrischen Multiprozessorumgebung

Kapazität und Skalierbarkeit

Sie können die E/A-Kapazität der Datenbankpartition, die Ihrem Prozessor zugeordnet ist, durch eine größere Anzahl von Platten erhöhen. Sie können E/A-Server speziell zur Verarbeitung von E/A-Anforderungen einrichten. Durch den Einsatz eines oder mehrerer E/A-Server für jede Platte kann mehr als eine E/A-Operation gleichzeitig stattfinden.

Wenn Sie das Maximum an Kapazität und Skalierbarkeit erreicht haben, können Sie die Umrüstung auf ein System mit mehreren Datenbankpartitionen in Erwägung ziehen.

Konfigurationen mit mehreren Datenbankpartitionen

Sie können eine Datenbank in mehrere Datenbankpartitionen aufteilen, die sich jeweils auf einer eigenen Maschine befinden. Mehrere Maschinen mit mehreren Datenbankpartitionen können in einer Gruppe zusammengefasst werden. In diesem Abschnitt werden die folgenden Konfigurationen von Datenbankpartitionen beschrieben:

- Datenbankpartitionen auf Systemen mit einem Prozessor

- Datenbankpartitionen auf Systemen mit mehreren Prozessoren
- Logische Datenbankpartitionen

Datenbankpartitionen mit einem Prozessor

In dieser Umgebung gibt es viele Datenbankpartitionen. Jede Datenbankpartition befindet sich auf einer eigenen Maschine mit eigenem Prozessor, eigenem Hauptspeicher und eigenen Platten (Abb. 33). Alle Maschinen sind über eine Kommunikationseinrichtung verbunden. Für eine solche Umgebung werden viele verschiedene Bezeichnungen gebraucht, wie zum Beispiel: Cluster, Cluster von Einzelprozessoren, Umgebung mit exklusiver Parallelverarbeitung (MPP - Massively Parallel Processing) oder Konfiguration ohne gemeinsame Nutzung von Ressourcen. Die letztgenannte Bezeichnung charakterisiert die Anordnung der Ressourcen. Im Gegensatz zu einer SMP-Umgebung findet in einer MPP-Umgebung keine gemeinsame Nutzung von Hauptspeicher oder Platten statt. In der MPP-Umgebung fallen daher auch die Einschränkungen aufgrund der gemeinsamen Nutzung von Hauptspeicher und Platten fort.

Durch die Partitionierung in einer Umgebung kann eine Datenbank trotz ihrer physischen Verteilung auf mehrere Datenbankpartitionen eine logische Gesamtheit bilden. Die Tatsache, dass die Daten verteilt sind, bleibt für die Mehrheit der Benutzer transparent. Die Arbeit kann unter den Datenbankmanagern aufgeteilt werden. Die Datenbankmanager in den einzelnen Datenbankpartitionen verrichten die Arbeit jeweils am eigenen Teil der Datenbank.

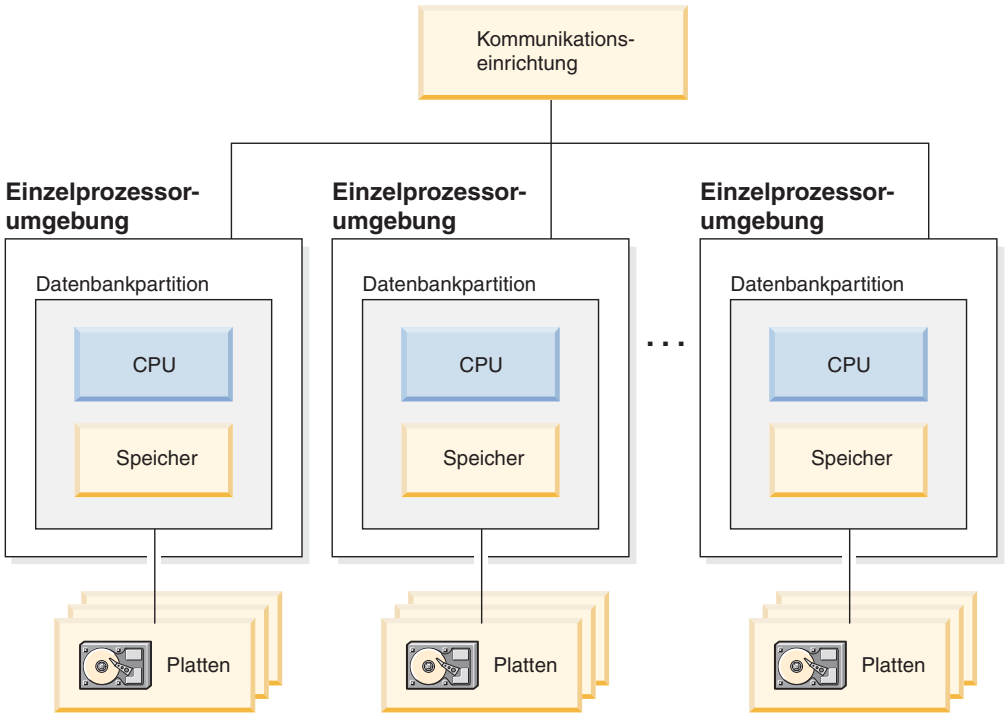


Abbildung 33. Umgebung zur exklusiven Parallelverarbeitung (MPP-Umgebung)

Kapazität und Skalierbarkeit

In dieser Umgebung können Sie Ihrer Konfiguration weitere Datenbankpartitionen hinzufügen. Auf einigen Plattformen beträgt die höchstmögliche Anzahl 512 Daten-

bankpartitionen. Jedoch kann es praktische Grenzen für die Verwaltung einer höheren Anzahl von Maschinen und Instanzen geben.

Wenn Sie das Maximum an Kapazität und Skalierbarkeit erreicht haben, können Sie die Umrüstung auf ein System in Erwägung ziehen, in dem jede Datenbankpartition mehrere Prozessoren hat.

Datenbankpartitionen mit mehreren Prozessoren

Eine Alternative zu einer Konfiguration, in der jede Datenbankpartition nur einen Prozessor hat, ist eine Konfiguration, in der jede Datenbankpartition über mehrere Prozessoren verfügt. Eine solche Konfiguration wird als *SMP-Cluster* bezeichnet (Abb. 34).

Diese Art der Konfiguration verbindet die Vorteile von SMP- und MPP-Parallelität. Dies bedeutet, dass eine Abfrage in einer Einzeldatenbankpartition mithilfe mehrerer Prozessoren durchgeführt werden kann. Darüber hinaus kann eine Abfrage auch parallel in mehreren Datenbankpartitionen durchgeführt werden.

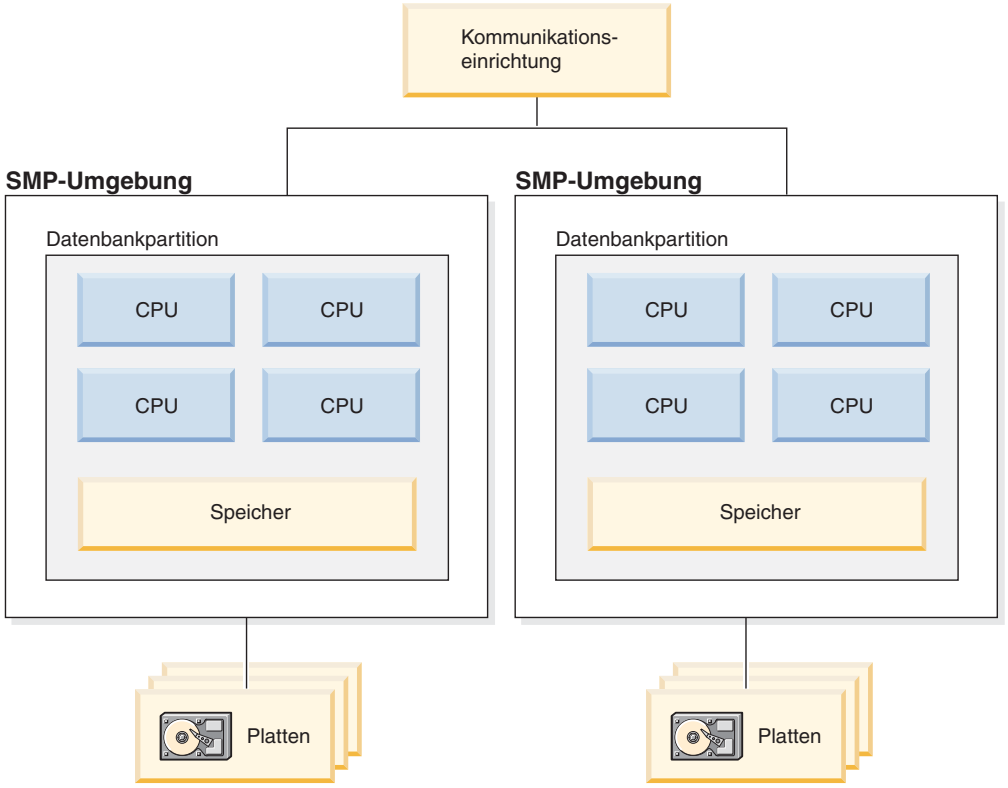


Abbildung 34. Mehrere symmetrische Multiprozessorumgebungen (SMP) in einem Cluster

Kapazität und Skalierbarkeit

In dieser Umgebung können Sie weitere Datenbankpartitionen und den vorhandenen Datenbankpartitionen weitere Prozessoren hinzufügen.

Logische Datenbankpartitionen

Eine logische Datenbankpartition unterscheidet sich von einer physischen Partition darin, dass ihr nicht die Steuerung einer ganzen Maschine unterliegt. Obwohl die

Maschine über gemeinsam genutzte Ressourcen verfügt, nutzen Datenbankpartitionen die Ressourcen nicht gemeinsam. Prozessoren werden gemeinsam genutzt, Platten und Hauptspeicher jedoch nicht.

Logische Datenbankpartitionen bieten Skalierbarkeit. Mehrere Datenbankmanager, die in mehreren logischen Partitionen ausgeführt werden, können die verfügbaren Ressourcen möglicherweise erschöpfender nutzen, als dies ein einzelner Datenbankmanager kann. Abb. 35 veranschaulicht die Möglichkeit, durch Hinzufügen weiterer Datenbankpartitionen auf einer SMP-Maschine mehr Skalierbarkeit zu gewinnen. Dies gilt insbesondere für Maschinen mit vielen Prozessoren. Durch die Verteilung der Datenbank können Sie jede Datenbankpartition getrennt verwalten und wiederherstellen.

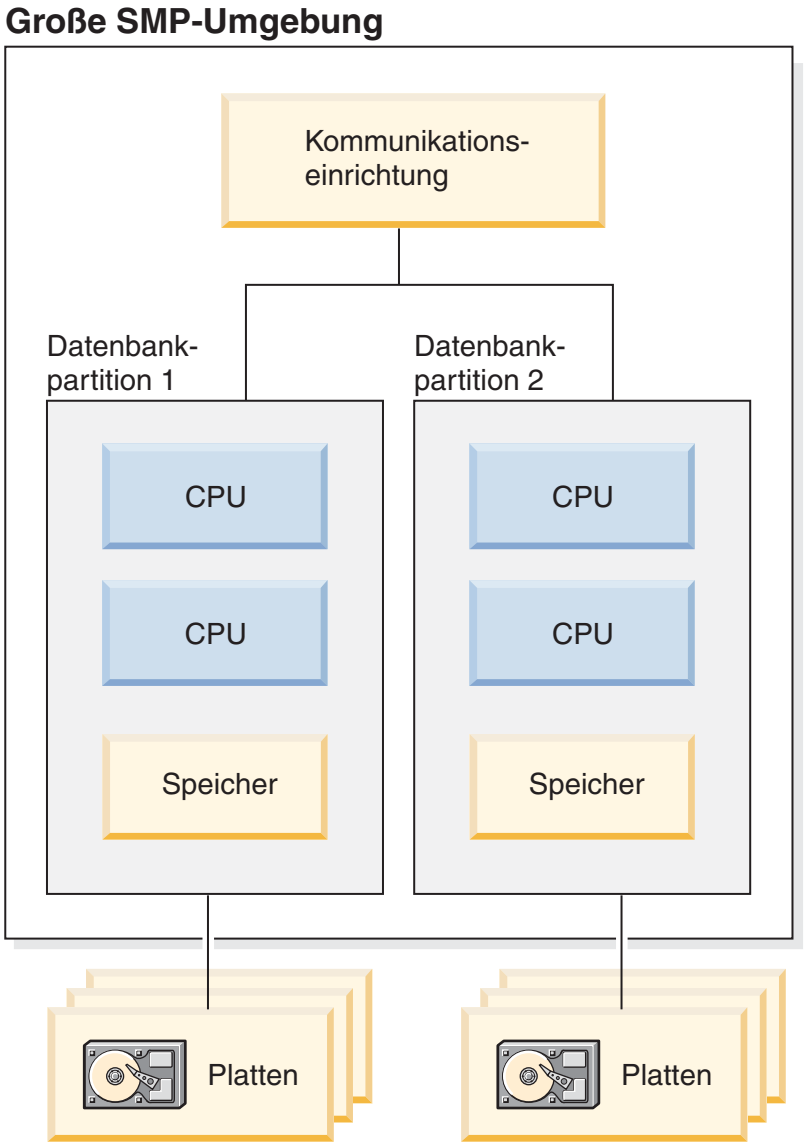


Abbildung 35. Partitionierte Datenbank mit symmetrischer Multiprozessorumgebung

Abb. 36 auf Seite 100 veranschaulicht die Möglichkeit, die in Abb. 35 gezeigte Konfiguration zu vervielfachen, um die Verarbeitungsleistung zu erhöhen.

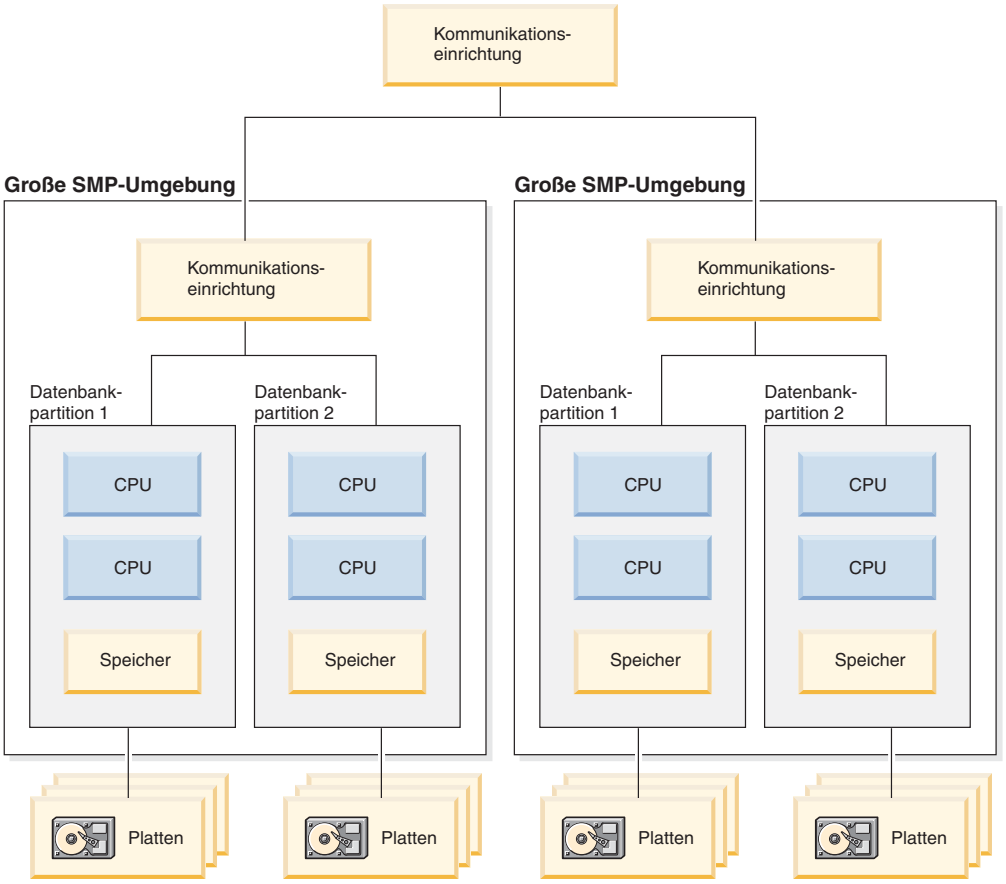


Abbildung 36. Partitionierte Datenbank mit SMP-Umgebungen in einem Cluster

Anmerkung: Die Möglichkeit, zwei oder mehr Datenbankpartitionen nebeneinander auf derselben Maschine zu haben (ungeachtet der Anzahl der Prozessoren), bietet eine größere Flexibilität bei der Einrichtung hoch verfügbarer Konfigurationen und der Implementierung von Übernahmestrategien bei Systemausfällen. Bei Ausfall einer Maschine kann eine Datenbankpartition automatisch von einer anderen Maschine, die bereits eine andere Datenbankpartition derselben Datenbank enthält, übernommen und wieder gestartet werden.

Zusammenfassung der in den einzelnen Hardwareumgebungen realisierbaren Parallelität

Die folgende Tabelle gibt eine Übersicht über die Arten der Parallelität, mit denen die verschiedenen Hardwareumgebungen am besten genutzt werden können.

Tabelle 8. Mögliche Arten der Parallelität in den verschiedenen Hardwareumgebungen

Hardwareumgebung	E/A-Parallelität	Abfrageinterne Parallelität	
		Partitionsinterne Parallelität	Partitionsübergreifende Parallelität
Einzeldatenbankpartition, Einzelprozessor	Ja	Nein [1]	Nein
Einzeldatenbankpartition, mehrere Prozessoren (SMP)	Ja	Ja	Nein

Tabelle 8. Mögliche Arten der Parallelität in den verschiedenen Hardwareumgebungen (Forts.)

Hardwareumgebung	**E/A-Parallelität**	**Abfrageinterne Parallelität**	
		Partitionsinterne Parallelität	**Partitionsübergreifende Parallelität**
Mehrere Datenbankpartitionen, ein Prozessor (MPP)	Ja	Nein [1]	Ja
Mehrere Datenbankpartitionen, mehrere Prozessoren (Cluster von SMP-Systemen)	Ja	Ja	Ja
Logische Datenbankpartitionen	Ja	Ja	Ja

[1] Es kann vorteilhaft sein, den Grad der Parallelität (mithilfe eines der Konfigurationsparameter) auch auf einem Einzelprozessor auf einen Wert größer als 1 zu setzen, besonders wenn die ausgeführten Abfragen die CPU nicht voll auslasten (z. B. wenn die Abfragen E/A-lastig sind).

Teil 2. Installationshinweise

Kapitel 5. Installationsvoraussetzungen

Installieren von DB2-Datenbankservern mit dem DB2-Installationsassistenten (Windows)

Diese Task beschreibt das Starten des DB2-Installationsassistenten unter Windows. Der DB2-Installationsassistent wird verwendet, um die gewünschte Installation zu definieren und das DB2-Datenbankprodukt auf dem System zu installieren.

Vorbereitende Schritte

Vor dem Starten des **DB2-Installationsassistenten**:

- Wenn Sie beabsichtigen, eine Umgebung mit partitionierten Datenbanken zu konfigurieren, finden Sie hierzu im Abschnitt "Einrichten einer Umgebung mit partitionierten Datenbanken" weitere Informationen.
- Stellen Sie sicher, dass Ihr System die Anforderungen im Hinblick auf die Installation, den Hauptspeicher und die Plattenspeicherkapazität erfüllt.
- Wenn Sie LDAP verwenden möchten, um den DB2-Server in Active Directory bei Windows-Betriebssystemen zu registrieren, müssen Sie das Verzeichnisschema vor der Installation erweitern. Andernfalls müssen Sie die Registrierung der Datenbank und die Katalogisierung der Datenbank manuell durchführen. Weitere Informationen hierzu finden Sie im Abschnitt „Erweitern des Active Directory-Schemas für LDAP-Verzeichnisservices (Windows)".
- Sie benötigen das lokale Konto für Benutzer mit Administratorberechtigung mit den empfohlenen Benutzerberechtigungen zum Ausführen der Installation. Bei DB2-Datenbankservern, bei denen die Benutzer-ID LocalSystem (lokales System) als DAS und DB2-Instanzbenutzer verwendet werden kann und bei denen die Umgebung mit partitionierten Datenbanken nicht verwendet wird, kann ein Nicht-Administrator mit erweiterten Zugriffsrechten die Installation durchführen.

  Anmerkung: Wenn ein Benutzer mit einem Benutzerkonto ohne Administratorberechtigung die Produktinstallation durchführen soll, muss die Laufzeitbibliothek VS2010 installiert werden, bevor ein DB2-Datenbankprodukt installiert wird. Die Laufzeitbibliothek VS2010 wird auf dem Betriebssystem benötigt, bevor das DB2-Datenbankprodukt installiert werden kann. Die Laufzeitbibliothek VS2010 ist auf der Download-Website für Microsoft-Laufzeitbibliotheken verfügbar. Sie haben zwei Auswahlmöglichkeiten: `vcredist_x86.exe` für 32-Bit-Systeme, `vcredist_x64.exe` für 64-Bit-Systeme.
- Auch wenn es nicht unbedingt erforderlich ist, sollten Sie alle Programme schließen, damit das Installationsprogramm alle Dateien auf dem Computer aktualisieren kann, ohne dass dazu ein Warmstart erforderlich ist.
- Die Installation von DB2-Produkten von einem virtuellen Laufwerk oder einem nicht zugeordneten Netzlaufwerk (z. B. *hostname**sharename* im Windows-Explorer) wird nicht unterstützt. Bevor Sie versuchen, DB2-Produkte zu installieren, müssen Sie das Netzlaufwerk einem Windows-Laufwerksbuchstaben zuordnen (z. B. Z:).

Einschränkungen

- Pro Benutzerkonto kann jeweils nur eine Instanz des DB2-Installationsassistenten ausgeführt werden.

- Der Name der DB2-Kopie und der Name der Instanz dürfen nicht mit einem numerischen Wert beginnen.Die DB2-Kopie ist auf 64 Zeichen des englischen Alphabets begrenzt; dabei sind die Zeichen A - Z, a - z und die Ziffern 0 - 9 zulässig.
- Der Name der DB2-Kopie und der Instanzname müssen für sämtliche DB2-Kopien eindeutig sein.
- Die Verwendung von XML-Funktionen ist auf Datenbanken beschränkt, die nur über eine Datenbankpartition verfügen.
- Kein anderes DB2-Datenbankprodukt darf im selben Pfad installiert werden, wenn eine der folgenden Komponenten bereits installiert ist:
  - IBM® Data Server Runtime Client
  - IBM Data Server Driver Package
  - *DB2 Information Center*
- In den Feldern des **DB2-Installationsassistenten** werden keine Sonderzeichen der jeweiligen Landessprache akzeptiert.
- Wenn Sie unter Windowsdie erweiterte Sicherheit aktivieren, müssen Benutzer zu der Gruppe DB2ADMNS oder DB2USERS gehören, um lokale DB2-Befehle und -Anwendungen auszuführen, da eine zusätzliche Sicherheitsfunktion (User Access Control) die Zugriffsrechte einschränkt, die lokalen Administratoren standardmäßig erteilt werden. Wenn Benutzer nicht zu einer dieser beiden Gruppen gehören, haben sie keinen Lesezugriff auf lokale DB2-Konfigurations- und -Anwendungsdaten.

Vorgehensweise

Gehen Sie wie folgt vor, um den **DB2-Installationsassistenten** zu starten:

1. Melden Sie sich mit dem für die Installation von DB2 definierten lokalen Administratorkonto am System an.
2. Wenn Sie über die DB2-Datenbankprodukt-DVD verfügen, legen Sie sie in das DVD-Laufwerk ein. Das **DB2 Setup-Launchpad** wird von der Funktion für automatische Ausführung automatisch gestartet, sofern diese Funktion aktiviert ist. Wenn die Funktion für automatische Ausführung nicht ordnungsgemäß reagieren sollte, durchsuchen Sie im Windows-Explorer die DB2-Datenbankprodukt-DVD und klicken das Installationssymbol (**setup**) doppelt an, um das **DB2 Setup-Launchpad** zu starten.
3. Wenn Sie das DB2-Datenbankprodukt von Passport Advantage heruntergeladen haben, führen Sie die ausführbare Datei aus, um die Installationsdateien des DB2-Datenbankprodukts zu extrahieren. Durchsuchen Sie im Windows-Explorer die DB2-Installationsdateien, und klicken Sie das Installationssymbol (**setup**) doppelt an, um das **DB2 Setup-Launchpad** zu starten.
4. Im **DB2 Setup-Launchpad** können Sie die Installationsvoraussetzungen und die Releaseinformationen anzeigen oder direkt mit der Installation fortfahren. Es empfiehlt sich, die Voraussetzungen für die Installation und die Releaseinformationen zu lesen, um die neuesten Informationen zu erhalten.
5. Klicken Sie **Produkt installieren** an. Im Fenster **Produkt installieren** werden die Produkte angezeigt, die zur Installation zur Verfügung stehen.

   Wenn auf dem Computer noch keine DB2-Datenbankprodukte installiert sind, starten Sie die Installation, indem Sie **Neue installieren** anklicken. Führen Sie die Installation aus, indem Sie den Eingabeaufforderungen des **DB2-Installationsassistenten** folgen.

   Wenn Sie mindestens ein DB2-Datenbankprodukt auf dem Computer installiert haben, haben Sie folgende Möglichkeiten:

- Klicken Sie **Neue installieren** an, um eine neue DB2-Kopie zu erstellen.
- Klicken Sie **Mit Vorhandenen arbeiten** an, um eine vorhandene DB2-Kopie zu aktualisieren, Funktionen zu einer DB2-Kopie hinzuzufügen, ein Upgrade für eine vorhandene DB2-Kopie der Version 9.7, Version 9.8 oder Version 10.1 durchzuführen oder ein Add-on-Produkt zu installieren.

6. Der **DB2-Installationsassistent** ermittelt die Systemsprache und startet das Installationsprogramm für diese Sprache. Es steht eine Onlinehilfefunktion zur Verfügung, die Sie durch die verbleibenden Schritte leitet. Klicken Sie **Hilfe** an oder drücken Sie die Funktionstaste **F1**, um die Onlinehilfe aufzurufen. Sie können die Installation jederzeit durch Anklicken von **Abbrechen** beenden.
7. Beispielfenster im DB2-Installationsassistenten führen Sie durch den Installationsprozess. Weitere Informationen finden Sie mithilfe der zugehörigen Links.

Ergebnisse

Das DB2-Datenbankprodukt wird standardmäßig im Verzeichnis *Program_Files*\IBM\sqllib installiert, wobei *Program_Files* für die Position des Verzeichnisses Programme darstellt.

Wenn Sie die Installation auf einem System vornehmen, auf dem dieses Verzeichnis bereits verwendet wird, wird dem Installationspfad für das DB2-Datenbankprodukt die Kennung *_xx* hinzugefügt, wobei *xx* Ziffern sind, die mit 01 beginnen und je nach Anzahl der installierten DB2-Kopien ansteigen.

Sie können auch einen eigenen Pfad für die Installation des DB2-Datenbankprodukts angeben.

Nächste Schritte

- Überprüfen Sie Ihre Installation.
- Führen Sie die erforderlichen Tasks nach der Installation aus.

Informationen zu Fehlern, die während der Installation aufgetreten sind, enthält die Installationsprotokolldatei im Verzeichnis Eigene Dateien\DB2LOG\. Die Protokolldatei verwendet das folgende Format: DB2-*produktabkürzung*-*datum_uhrzeit*.log. Beispiel: DB2-ESE-Tue Apr 04 17_04_45 2012.log.

Wenn es sich hierbei um eine Installation eines neuen DB2-Produkts unter Windows (64 Bit) handelt und Sie einen OLE-Datenbank-Provider (32 Bit) verwenden, müssen Sie die DLL-Datei IBMDADB2 manuell registrieren. Führen Sie zum Registrieren dieser DLL-Datei den folgenden Befehl aus:

```
c:\windows\SysWOW64\regsvr32 /s c:\Programme\IBM\SQLLIB\bin\ibmdadb2.dll
```

Dabei steht *Programme* für die Speicherposition des Verzeichnisses Programme.

Wenn Sie möchten, dass Ihr DB2-Datenbankprodukt auf die DB2-Dokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netz zugreifen kann, müssen Sie das *DB2 Information Center* installieren. Das *DB2 Information Center* enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen DB2-Produkte. Standardmäßig werden die Informationen zu DB2 aus dem Web abgerufen, wenn das *DB2 Information Center* nicht lokal installiert ist.

IBM Data Studio kann mithilfe des **DB2-Installationsassistenten** installiert werden.

Speicherbegrenzungen für DB2 Express Edition und DB2 Workgroup Server Edition Beim Installieren von DB2 Express Edition beträgt der maximal zulässige Speicherbereich für die Instanz 4 GB.

Beim Installieren von DB2 Workgroup Server Edition beträgt der maximal zulässige Speicherbereich für die Instanz 64 GB.

Wie groß der für die Instanz zugeordnete Speicherbereich ist, hängt vom Konfigurationsparameter **INSTANCE_MEMORY** des Datenbankmanagers ab.

Wichtige Hinweise für das Durchführen eines Upgrades von V9.7, V9.8 oder V10.1:

- Der Manager für den Speicher mit automatischer Leistungsoptimierung vergrößert den Gesamtspeicher für die Instanz nicht über die Lizenzgrenzwerte hinaus.

Vorbereiten der Umgebung für partitionierte DB2-Server (Windows)

Dieser Abschnitt beschreibt die erforderlichen Schritte zum Vorbereiten Ihrer Windows-Umgebung für eine partitionierte Installation des DB2-Datenbankprodukts.

Vorbereitende Schritte

Wird eine neue Maschine als Partition in einer Umgebung mit partitionierten Datenbanken hinzugefügt, muss die neue Maschine die folgenden Voraussetzungen erfüllen:

- Die Betriebssystemversion muss mit der Version der Maschine übereinstimmen, der die Instanz gehört.
- Die CPU-Architektur (x32 Bit oder x64 Bit) muss mit der Architektur der Maschine übereinstimmen, der die Instanz gehört.

Wenn die neue Maschine diese Voraussetzungen nicht erfüllt, schlägt das Hinzufügen der Partition möglicherweise fehl.

Vorgehensweise

Gehen Sie wie folgt vor, um die Windows-Umgebung für die Installation vorzubereiten:

1. Stellen Sie sicher, dass der Primärcomputer und die zugehörigen Computer zu derselben Windows-Domäne gehören. Überprüfen Sie die Domäne, zu der der Computer gehört, mithilfe des Dialogfelds zu den Systemeigenschaften in der Systemsteuerung.
2. Stellen Sie sicher, dass die Einstellungen für Uhrzeit und Datum auf dem primären und den zugehörigen Computern konsistent sind. Damit die Zeit als gleich betrachtet wird, dürfen die Unterschiede in der Zeit (bezogen auf GMT - Westeuropäische Zeit) auf allen Computern nicht größer als eine Stunde sein.

   Das Systemdatum und die Systemzeit können im Dialog für Datum und Uhrzeit über die Systemsteuerung geändert werden. Sie können den Konfigurationsparameter **max_time_diff** verwenden, um diese Einschränkung zu ändern. Die Standardeinstellung ist `max_time_diff = 60`. Sie ermöglicht einen Unterschied von weniger als 60 Minuten.
3. Stellen Sie sicher, dass jedes Computerobjekt, das an einer Umgebung mit partitionierten Datenbanken beteiligt ist, das Zugriffsrecht zum Akzeptieren des Computers für die Delegierung ("Trust computer for delegation") markiert hat. Sie können überprüfen, ob das Kontrollkästchen "Trust computer for delegati-

on" auf der Registerkarte für allgemeine Optionen des Fensters für Kontoeigenschaften auf jedem Computer in der Konsole für Active Directory-Benutzer und -Computer markiert ist.

4. Stellen Sie sicher, dass alle zugehörigen Computer über TCP/IP miteinander kommunizieren können:
   a. Geben Sie auf einem der zugehörigen Computer den Befehl **hostname** ein, der den Hostnamen des Computers zurückgibt.
   b. Geben Sie auf einem anderen zugehörigen Computer den folgenden Befehl ein:

      ```
      ping hostname
      ```

      Hierbei ist *hostname* der Hostname des Primärcomputers. Ist der Test erfolgreich, erhalten Sie eine Ausgabe ähnlich der folgenden:

      ```
      Pinging ServerA.ibm.com [9.21.27.230] with 32 bytes of data:

      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      ```

      Wiederholen Sie diese Schritte, bis Sie überprüft haben, ob alle zugehörigen Computer über TCP/IP miteinander kommunizieren können. Jeder Computer muss eine statische IP-Adresse haben.

      Wenn Sie planen, mehrere Netzadapter zu verwenden, können Sie angeben, welcher Adapter für die Kommunikation zwischen den Datenbankpartitionsservern verwendet werden soll. Verwenden Sie den Befehl **db2nchg**, um nach Abschluss der Installation das Feld 'netname' in der Datei `db2nodes.cfg` anzugeben.
5. Während der Installation werden Sie aufgefordert, ein Benutzerkonto für den DB2-Verwaltungsserver (DAS) anzugeben. Dabei handelt es sich um ein lokales Benutzerkonto oder ein Domänenbenutzerkonto, das für den DB2-Verwaltungsserver (DAS) verwendet wird. Der DAS ist ein Verwaltungsservice, der verwendet wird, um die GUI-Tools zu unterstützen und bei Verwaltungstasks zu helfen. Sie können an dieser Stelle einen Benutzer definieren oder den DB2-Installationsassistenten einen Benutzer erstellen lassen. Wenn der DB2-Installationsassistent einen neuen Domänenbenutzer erstellen soll, muss das für die Installation verwendete Konto über eine Berechtigung zum Erstellen von Domänenbenutzern verfügen.
6. Auf dem Primärcomputer, auf dem die Partition, der die Instanz gehört, installiert werden soll, muss ein Domänenbenutzerkonto vorhanden sein, das zur lokalen Gruppe der *Administratoren* gehört. Wenn Sie DB2-Datenbankprodukte installieren, melden Sie sich als dieser Benutzer an. Sie müssen der lokalen Gruppe der *Administratoren* auf jedem der zugehörigen Computer jeweils das gleiche Benutzerkonto hinzufügen. Dieser Benutzer muss über die Benutzerberechtigung *Als Teil des Betriebssystems handeln* verfügen.
7. Stellen Sie sicher, dass der lokale Laufwerkbuchstabe vor dem Datenbankverzeichnis auf allen Computern in der Instanz identisch ist. Sie können dies überprüfen, indem Sie den Befehl **GET DATABASE CONFIGURATION** ausführen, und den Wert des DBM Konfigurationsparameters **dftdbpath** prüfen.
8. Während der Installation werden Sie aufgefordert, ein Domänenbenutzerkonto anzugeben, das der DB2-Instanz zugeordnet werden soll. Jeder DB2-Instanz ist ein Benutzer zugeordnet. Beim Starten der Instanz wird das DB2-Datenbanksystem mit diesem Benutzernamen gestartet. Sie können an dieser Stelle einen Benutzer definieren oder den DB2-Installationsassistenten einen neuen Domänenbenutzer erstellen lassen.

Beim Hinzufügen eines neuen Knotens zu einer partitionierten Umgebung muss der Name der DB2-Kopie auf allen Computern identisch sein.

Wenn der DB2-Installationsassistent einen neuen Domänenbenutzer erstellen soll, muss das für die Installation verwendete Konto über eine Berechtigung zum Erstellen von Domänenbenutzern verfügen. Das Domänenkonto des Instanzbenutzers muss auf allen zugehörigen Computern zur lokalen Gruppe *Administratoren* gehören und erhält die folgenden Benutzerberechtigungen:

- Als Teil des Betriebssystems handeln
- Erstellen von Tokenobjekten
- Sperren von Seiten im Speicher
- Anmelden als Service
- Anheben von Quoten
- Ersetzen eines Tokens auf Prozessebene

Wenn die erweiterte Sicherheit ausgewählt wurde, muss das Konto auch Mitglied der Gruppe DB2ADMNS sein. Die Gruppe DB2ADMNS verfügt bereits über diese Zugriffsrechte; die Zugriffsrechte wurden also bereits explizit zum Konto hingefügt.

Fast Communications Manager (Windows)

In Umgebungen mit mehreren Membern hat jedes Member ein Paar FCM-Dämonen zur Unterstützung der mit Agentenanforderungen verbundenen Kommunikation zwischen Membern. Der eine Dämon dient zum Senden von Übertragungen, der andere zum Empfangen. Diese Dämonen und die unterstützende Infrastruktur werden aktiviert, wenn eine Instanz gestartet wird. Die FCM-Kommunikation wird außerdem für Agenten verwendet, die innerhalb desselben Members aktiv sind. Dieser Typ von Kommunikation wird auch als memberinterne Kommunikation bezeichnet.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter **fcm_num_buffers** des Datenbankmanagers festlegen. Sie können die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter **fcm_num_channels** des Datenbankmanagers festlegen. Standardmäßig sind die Konfigurationsparameter **fcm_num_buffers** und **fcm_num_channels** des Datenbankmanagers auf den Wert AUTOMATIC gesetzt. Bei der Einstellung AUTOMATIC, die auch die empfohlene Einstellung ist, überwacht FCM die Ressourcennutzung und passt Ressourcen an den Workloadbedarf an.

Übersicht über die Installation von DB2-Datenbankservern (Linux und UNIX)

Dieser Abschnitt beschreibt die Schritte zum Installieren Ihres DB2-Serverprodukts unter AIX, HP-UX, Linux und Solaris.

Vorgehensweise

Führen Sie die folgenden Schritte aus, um das DB2-Serverprodukt zu installieren:

1. Prüfen Sie die Voraussetzungen für das DB2-Produkt.
2. Lesen Sie die Informationen zum DB2-Upgrade (sofern anwendbar).
3. Ändern Sie die Kernelparameter unter HP-UX, Linux und Solaris. Auf allen Plattformen außer Linux auf x86_32 müssen Sie einen 64-Bit-Kernel installieren, bevor Sie mit der Installation fortfahren können; andernfalls schlägt die Installation fehl.

4. Bereiten Sie die Installationsmedien vor:

   Produkt-DVD
   Wenn die DB2-Produkt-DVD nicht automatisch angehängt wird, hängen Sie die DB2-Produkt-DVD selbst an.

   Installationsimage
   Wenn Sie ein Installationsimage heruntergeladen haben, müssen Sie die Datei entpacken.
5. Installieren Sie das DB2-Produkt mit einer der zur Verfügung stehenden Methoden:
   - DB2-Installationsassistent
   - Unbeaufsichtigte Installation mithilfe einer Antwortdatei
   - Implementierung mit Nutzdatendateien

   Für DB2-Server können Sie den DB2-Installationsassistenten verwenden, um die folgenden Installations- und Konfigurationstasks auszuführen:
   - Auswählen des DB2-Installationstyps (typisch, komprimiert oder angepasst).
   - Auswählen der Position für die DB2-Produktinstallation.
   - Installieren der Sprachen, von denen zu einem späteren Zeitpunkt eine als voreingestellte Sprache für die Produktschnittstelle und die Nachrichten angegeben werden kann.
   - Installieren oder Durchführen eines Upgrades von IBM Tivoli System Automation for Multiplatforms (Linux und AIX).
   - Konfigurieren einer DB2-Instanz.
   - Installieren des DB2-Verwaltungsservers (DB2 Administration Server, DAS), einschließlich der Einrichtung von DAS-Benutzern.
   - Konfigurieren des DB2 Text Search-Servers
   - Einrichten der Verwaltungsansprechpartner und der Benachrichtigungen des Diagnosemonitors.
   - Installieren und Konfigurieren von Instanzen (einschließlich der Einrichtung von Instanzbenutzern).
   - Einrichten der Informix-Datenquellenunterstützung.
   - Vorbereiten des Katalogs der vorhandenen DB2-Tools.
   - Angeben des Ports des DB2 Information Center.
   - Erstellen von Antwortdateien.
6. Wenn Sie einen DB2-Server mit einer anderen Methode als der Verwendung des DB2-Installationsassistenten installiert haben, sind Konfigurationsschritte nach der Installation erforderlich.

DB2-Installationsmethoden

Es gibt mehrere Methoden zur Installation von DB2-Datenbankprodukten. Dabei entspricht jede dieser Methoden einer bestimmten Installationssituation.

Die nachstehende Tabelle zeigt die für die einzelnen Betriebssysteme jeweils verfügbaren Installationsmethoden.

Tabelle 9. Installationsmethode nach Betriebssystem

Installationsmethode	Windows	Linux oder UNIX
DB2-Installationsassistent	Ja	Ja
Installation mithilfe einer Antwortdatei	Ja	Ja

Tabelle 9. Installationsmethode nach Betriebssystem (Forts.)

Installationsmethode	Windows	Linux oder UNIX
Befehl **db2_install**	Nein	Ja
Implementierung mit Nutzdatendateien	Nein	Ja

Wichtig: Der Befehl **db2_install** ist veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Verwenden Sie stattdessen den Befehl **db2setup** oder die Installationsmethode mithilfe einer Antwortdatei.

Die nachstehende Liste beschreibt die DB2-Installationsmethoden.

DB2-Installationsassistent
Der DB2-Installationsassistent ist ein GUI-Installationsprogramm, das für Linux, UNIX und Windows zur Verfügung steht. Er bietet eine benutzerfreundliche Schnittstelle für die Installation von DB2-Datenbankprodukten und die Durchführung der anfänglichen Installations- und Konfigurationstasks.

Der DB2-Installationsassistent kann auch zum Erstellen von DB2-Instanzen und Antwortdateien verwendet werden, mit denen diese Installation auf andere Maschinen dupliziert werden kann.

Anmerkung: Auf nicht als Root ausgeführten Installationen unter Linux und UNIX darf nur eine DB2-Instanz vorhanden sein. Er erstellt die nicht als Root ausgeführte Instanz automatisch.

Unter Linux und UNIX ist ein X-Server erforderlich, um den DB2-Installationsassistenten anzuzeigen.

Installation mithilfe einer Antwortdatei
Eine Antwortdatei ist eine Textdatei, die Werte für die Installation und Konfiguration enthält. Die Datei wird vom Programm **DB2 Setup** gelesen, und die Installation wird auf der Grundlage der in dieser Datei angegebenen Werte ausgeführt.

Eine Installation mithilfe einer Antwortdatei kann auch als unbeaufsichtigte Installation bezeichnet werden.

Darüber hinaus ermöglichen Antwortdateien den Zugriff auf Parameter, die nicht über den DB2-Installationsassistenten definiert werden können.

Wenn Sie unter Linux- und UNIX-Betriebssystemen das DB2-Installationsimage in Ihre Anwendung integrieren, ermöglichen Sie dadurch Ihrer Anwendung, Informationen zum Fortgang der Installation und Anweisungen des Installationsprogramms in einer für den Computer lesbaren Form vom Installationsprogramm zu empfangen. Dieses Verhalten wird durch das Schlüsselwort **INTERACTIVE** der Antwortdatei gesteuert.

Antwortdateien können auf unterschiedliche Arten erstellt werden:

Verwenden des Antwortdateigenerators
Sie können mithilfe des Antwortdateigenerators eine Antwortdatei erstellen, die eine bestehende Installation repliziert. Wenn Sie beispielsweise einen IBM Data Server-Client installieren, würden Sie den Client zunächst vollständig konfigurieren und dann eine Antwortdatei generieren, um die Installation und Konfiguration des Clients auf anderen Computern zu replizieren.

Verwenden des DB2-Installationsassistenten

Mithilfe des DB2-Installationsassistenten können Sie eine Antwortdatei auf der Basis der Optionen erstellen, die Sie mit dem DB2-Installationsassistenten auswählen. Die von Ihnen gewählten Optionen werden in einer Antwortdatei aufgezeichnet, die Sie in einem Verzeichnis auf Ihrem System speichern können. Wenn Sie die Installation einer partitionierten Datenbank auswählen, werden zwei Antwortdateien generiert: eine für den Computer, der Instanzeigner ist, und eine für die zugehörigen Computer.

Ein Vorteil dieser Installationsmethode besteht darin, dass Sie eine Antwortdatei erstellen können, ohne eine Installation durchzuführen. Diese Funktion kann hilfreich sein, um die erforderlichen Optionen zum Installieren des DB2-Datenbankprodukts zu erfassen. Die Antwortdatei kann später verwendet werden, um das DB2-Datenbankprodukt genau mit den angegebenen Optionen zu installieren.

Sie können ein Client- oder Serverprofil mit dem Befehl **db2cfexp** exportieren, um die Client- oder Serverkonfiguration zu speichern. Anschließend können Sie das Profil mit dem Befehl **db2cfimp** importieren. Mit dem Befehl **db2cfexp** exportierte Client- und Serverprofile können auch bei einer Installation mit Antwortdatei mithilfe des Schlüsselworts **CLIENT_IMPORT_PROFILE** importiert werden.

Exportieren Sie das Client- oder Serverprofil erst nach Durchführung der Installation und Katalogisierung der Datenquellen.

Durch Anpassung der Musterantwortdateien, die mit allen DB2-Datenbankprodukten geliefert werden

Wenn Sie für die Erstellung einer Antwortdatei nicht den Antwortdateigenerator oder den DB2-Installationsassistenten verwenden möchten, haben Sie auch die Möglichkeit, eine Musterantwortdatei manuell zu ändern. Musterantwortdateien werden auf der DB2-Datenbankprodukt-DVD bereitgestellt. Die Musterantwortdateien enthalten detaillierte Angaben zu den gültigen Schlüsselwörtern für die einzelnen Produkte.

Befehl db2_install (nur Linux- und UNIX-Betriebssysteme)

Der Befehl **db2_install** installiert *alle* Komponenten für das von Ihnen angegebene DB2-Datenbankprodukt für die englische Schnittstelle. Weitere Sprachenunterstützungen können Sie mithilfe des Parameters **-L** auswählen. Sie können keine Komponenten auswählen oder löschen.

Der Befehl **db2_install** installiert zwar alle Komponenten für das von Ihnen angegebene DB2-Datenbankprodukt, erstellt jedoch keine Benutzer, Gruppen und Instanzen und führt keine Konfiguration durch. Diese Installationsmethode ist unter Umständen in Fällen vorzuziehen, in denen das Konfigurieren nach der Installation erforderlich ist. Wenn Sie das DB2-Datenbankrodukt während der Installation konfigurieren möchten, sollten Sie die Verwendung des DB2-Installationsassistenten in Betracht ziehen.

Wenn Sie unter Linux- und UNIX-Betriebssystemen das DB2-Installationsimage in Ihre Anwendung integrieren, ermöglichen Sie dadurch Ihrer Anwendung, Informationen zum Fortgang der Installation und Anweisungen des Installationsprogramms in einer für den Computer lesbaren Form vom Installationsprogramm zu empfangen.

Bei dieser Installationsmethode sind nach dem Implementieren der Produktdateien manuelle Konfigurationsschritte erforderlich.

Hinweis: Der Befehl **db2_install** ist veraltet und wird möglicherweise in einem zukünftigen Release entfernt.

Implementierung mit Nutzdatendateien (nur Linux und UNIX)
Bei dieser Methode handelt es sich um eine fortgeschrittene Installationsmethode, die für die meisten Benutzer nicht zu empfehlen ist. Der Benutzer muss in ihrem Verlauf manuell Dateien mit Nutzdaten installieren. Eine Datei mit Nutzdaten ist eine komprimierte TAR-Datei, in der alle Dateien und Metadaten für eine installierbare Komponente enthalten sind.

Diese Methode wird für DB2 pureScale-Installationen nicht unterstützt. Bei dieser Installationsmethode sind nach dem Implementieren der Produktdateien manuelle Konfigurationsschritte erforderlich.

Anmerkung: DB2-Datenbankproduktinstallationen sind unter Linux und UNIX keine Betriebssystempakete mehr. Sie können für die Installation daher keine Betriebssystembefehle mehr verwenden. Alle vorhandenen Scripts, die Sie für die Schnittstelle mit und zum Abfragen von DB2-Datenbankproduktinstallationen verwenden, müssen geändert werden.

Installieren von DB2-Servern mit dem DB2-Installationsassistenten (Linux und UNIX)

In diesem Abschnitt wird beschrieben, wie der DB2-Installationsassistent unter Linux- und UNIX-Betriebssystemen gestartet wird. Der **DB2-Installationsassistent** wird verwendet, um die gewünschten Installationsvorgaben festzulegen und das DB2-Datenbankprodukt auf dem System zu installieren.

Vorbereitende Schritte

Vor dem Starten des **DB2-Installationsassistenten**:

- Wenn Sie vorhaben, eine Umgebung mit partitionierten Datenbanken einzurichten, finden Sie hierzu weitere Informationen im Abschnitt „Einrichten einer Umgebung mit partitionierten Datenbanken" im Handbuch *DB2-Server - Installation*.
- Stellen Sie sicher, dass Ihr System die Anforderungen im Hinblick auf die Installation, den Hauptspeicher und die Plattenspeicherkapazität erfüllt.
- Stellen Sie sicher, dass ein unterstützter Browser installiert ist.
- Sie können einen DB2-Datenbankserver entweder mit oder ohne Rootberechtigung installieren. Weitere Informationen zur Installation ohne Rootberechtigung finden Sie im Abschnitt „Installation ohne Rootberechtigung (Linux und UNIX - Übersicht)" in *DB2-Server - Installation*.
- Das DB2-Datenbankproduktimage muss verfügbar sein. DB2-Installationsimages sind entweder durch den Erwerb einer physischen DB2-Datenbankprodukt-DVD oder durch Herunterladen eines Installationsimages von Passport Advantage erhältlich.
- Wenn Sie landessprachliche Versionen eines DB2-Datenbankprodukts installieren, benötigen Sie die entsprechenden Landessprachenpakete.
- Der DB2-Installationsassistent ist ein grafisch orientiertes Installationsprogramm. Um den **DB2-Installationsassistenten** auf Ihrer Maschine ausführen zu können, benötigen Sie eine X Window System-Software, die eine grafische Benutzerschnittstelle wiedergeben kann. Stellen Sie sicher, dass der X Window-Server aktiv ist. Stellen Sie sicher, dass Sie Ihre Anzeige (DISPLAY) ordnungsgemäß exportiert haben. Beispiel: `export DISPLAY=9.26.163.144:0.`

- Wird in Ihrer Umgebung Sicherheitssoftware verwendet, müssen Sie die erforderlichen DB2-Benutzer manuell erstellen, bevor Sie den **DB2-Installationsassistenten** starten.

Einschränkungen

- Pro Benutzerkonto kann jeweils nur eine Instanz des DB2-Installationsassistenten ausgeführt werden.
- Die Verwendung von XML-Funktionen ist auf Datenbanken beschränkt, die mit dem codierten Zeichensatz UTF-8 definiert sind und nur über eine Datenbankpartition verfügen.
- In den Feldern des **DB2-Installationsassistenten** werden keine Sonderzeichen der jeweiligen Landessprache akzeptiert.
- Bei HP-UX 11i V2 auf Itanium-basierten HP Integrity Series Systems besteht auf Benutzer, die mit dem DB2-Installationsassistenten für Instanzeigner, abgeschirmte Benutzer oder DAS erstellt wurden, kein Zugriff mit dem Kennwort, das im DB2-Installationsassistenten angegeben wurde. Nach dem Abschluss des Installationsassistenten müssen Sie das Kennwort für diese Benutzer zurücksetzen. Dies hat keine Auswirkungen auf die Instanz- oder DAS-Erstellung mit dem Installationsassistenten, sodass Sie die Instanz bzw. den DAS nicht erneut erstellen müssen.

Vorgehensweise

Gehen Sie wie folgt vor, um den **DB2-Installationsassistenten** zu starten:

1. Wenn Sie über eine physische DB2-Datenbankprodukt-DVD verfügen, wechseln Sie in das Verzeichnis, in dem die DB2-Datenbankprodukt-DVD angehängt ist. Geben Sie dazu den folgenden Befehl ein:

   ```
   cd /dvdrom
   ```

   Dabei steht */dvdrom* für den Mountpunkt der DB2-Datenbankprodukt-DVD.
2. Wenn Sie das DB2-Datenbankproduktimage heruntergeladen haben, müssen Sie die Produktdatei extrahieren und entpacken.
   a. Extrahieren Sie die Produktdatei:

      ```
      gzip -d produkt.tar.gz
      ```

      Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.
   b. Entpacken Sie die Produktdatei:

      Unter Linux-Betriebssystemen

      ```
      tar -xvf produkt.tar
      ```

      Unter AIX-, HP-UX- und Solaris-Betriebssystemen

      ```
      gnutar -xvf produkt.tar
      ```

      Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.
   c. Wechseln Sie das Verzeichnis:

      ```
      cd ./produkt
      ```

      Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.

Anmerkung: Wenn Sie das Landessprachenpaket heruntergeladen haben, entpacken Sie es in demselben Verzeichnis. So werden die Unterverzeichnisse (z. B. `./nlpack`) in demselben Verzeichnis erstellt und das Installationsprogramm kann die Installationsimages automatisch und ohne Aufforderung an den Benutzer finden.

3. Geben Sie den Befehl **./db2setup** von dem Verzeichnis aus ein, in dem sich das Datenbankproduktimage befindet, um den DB2-Installationsassistenten zu starten.
4. Das Fenster **IBM DB2 Setup - Launchpad** wird geöffnet. In diesem Fenster können Sie die Installationsvoraussetzungen und die Release-Informationen anzeigen oder direkt mit der Installation fortfahren. Sie können auch die Installationsvoraussetzungen und die Release-Informationen aufrufen, um die neuesten Informationen abzurufen.
5. Klicken Sie **Produkt installieren** an. Im Fenster **Produkt installieren** werden die Produkte angezeigt, die zur Installation zur Verfügung stehen.

   Starten Sie die Installation, indem Sie **Neue installieren** anklicken. Führen Sie die Installation aus, indem Sie den Eingabeaufforderungen des **DB2-Installationsassistenten** folgen.
6. Beispielfenster im DB2-Installationsassistenten führen Sie durch den Installationsprozess. Weitere Informationen finden Sie mithilfe der zugehörigen Links.

   Rufen Sie nach der Initialisierung der Installation die einzelnen Installationsanzeigen des DB2-Installationsassistenten nacheinander auf und wählen Sie die gewünschten Optionen aus. Informationen zur Ausführung der restlichen Schritte finden Sie in der Installationshilfe. Klicken Sie zum Aufrufen der Installationshilfe **Hilfe** an oder drücken Sie die Taste F1. Sie können die Installation jederzeit durch Anklicken von **Abbrechen** beenden.

Ergebnisse

Installationen von DB2-Datenbankprodukten ohne Rootberechtigung werden immer im Verzeichnis *$HOME*/sqllib installiert. Dabei ist *$HOME* das Ausgangsverzeichnis des Benutzers ohne Rootberechtigung.

Bei Installationen mit Rootberechtigung werden DB2-Datenbankprodukte standardmäßig im folgenden Verzeichnis bzw. einem der folgenden Verzeichnisse installiert:

AIX, HP-UX oder Solaris
/opt/IBM/db2/V10.5

Linux /opt/ibm/db2/V10.5

Wenn Sie die Installation auf einem System vornehmen, auf dem dieses Verzeichnis bereits verwendet wird, wird dem Installationspfad für das DB2-Datenbankprodukt die Kennung *_xx* hinzugefügt, wobei *_xx* für Ziffern steht, die mit 01 beginnen und je nach Anzahl der installierten DB2-Kopien ansteigen.

Sie können auch einen eigenen Pfad für die Installation des DB2-Datenbankprodukts angeben.

Für DB2-Installationspfade gelten die folgenden Regeln:

- Sie dürfen Kleinbuchstaben (a-z), Großbuchstaben (A-Z) und das Unterstreichungszeichen (_) enthalten.
- Sie dürfen nicht länger als 128 Zeichen sein.
- Sie dürfen keine Leerzeichen enthalten.

- Sie dürfen keine Sonderzeichen der jeweiligen Landessprache enthalten.

Die folgenden Installationsprotokolldateien werden verwendet:

- Die DB2-Installationsprotokolldatei. Diese Datei erfasst alle DB2-Installationsinformationen einschließlich Fehlern.
  - Bei Rootinstallationen lautet der Name der DB2-Installationsprotokolldatei `db2setup.log`.
  - Bei nicht als Root ausgeführten Installationen lautet der Name der DB2-Installationsprotokolldatei `db2setup_`*`benutzername`*`.log`. Dabei ist *benutzername* die Benutzer-ID ohne Rootberechtigung, unter der die Installation durchgeführt wurde.
- Die DB2-Fehlerprotokolldatei. Diese Datei erfasst alle Fehlernachrichten, die von Java™ zurückgegeben werden (z. B. Nachrichten zu Ausnahmebedingungen und Traps).
  - Bei Rootinstallationen lautet der Name der DB2-Fehlerprotokolldatei `db2setup.err`.
  - Bei nicht als Root ausgeführten Installationen lautet der Name der DB2-Fehlerprotokolldatei `db2setup_`*`benutzername`*`.err`. Dabei ist *benutzername* die Benutzer-ID ohne Rootberechtigung, unter der die Installation durchgeführt wurde.

Diese Protokolldateien befinden sich standardmäßig im Verzeichnis `/tmp`. Die Speicherposition der Protokolldateien kann angegeben werden.

Die Datei `db2setup.his` wird nicht mehr verwendet. Stattdessen speichert das DB2-Installationsprogramm eine Kopie der DB2-Installationsprotokolldatei im Verzeichnis `DB2_DIR/install/logs/` und benennt sie in `db2install.history` um. Wenn der Name bereits vorhanden ist, benennt das DB2-Installationsprogramm die Datei in `db2install.history.`*`xxxx`* um. Dabei ist *xxxx* eine Zahl von `0000` bis `9999`, die davon abhängt, wie viele Installationen sich auf der Maschine befinden.

Jede Installationskopie verfügt über eine separate Liste an Protokolldateien. Wenn eine Installationskopie entfernt wird, werden auch die Protokolldateien in diesem Installationspfad entfernt. Dieser Kopiervorgang wird gegen Ende der Installation ausgeführt. Wenn das Programm vor dem Abschluss der Installation gestoppt oder abgebrochen wird, wird die Protokolldatei nicht erstellt.

Nächste Schritte

- Überprüfen Sie Ihre Installation.
- Führen Sie die erforderlichen Tasks nach der Installation aus.

IBM Data Studio kann mithilfe des **DB2-Installationsassistenten** installiert werden.

Landessprachenpakete können auch durch Ausführen des Befehls **`./db2setup`** in dem Verzeichnis, in dem sich das Landessprachenpaket befindet, installiert werden, nachdem das DB2-Datenbankprodukt installiert wurde.

Wenn Sie mit der x86-Version von Linux arbeiten und möchten, dass Ihr DB2-Datenbankprodukt auf die DB2-Dokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netz zugreifen kann, müssen Sie das *DB2 Information Center* installieren. Das *DB2 Information Center* enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen DB2-Produkte.

Speicherbegrenzungen für DB2 Express Edition und DB2 Workgroup Server Edition Beim Installieren von DB2 Express Edition beträgt der maximal zulässige Speicherbereich für die Instanz 4 GB.

Beim Installieren von DB2 Workgroup Server Edition beträgt der maximal zulässige Speicherbereich für die Instanz 64 GB.

Wie groß der für die Instanz zugeordnete Speicherbereich ist, hängt vom Konfigurationsparameter **INSTANCE_MEMORY** des Datenbankmanagers ab.

Wichtige Hinweise für das Durchführen eines Upgrades von V9.7, V9.8 oder V10.1:

- Wenn die Speicherkonfiguration für Ihr DB2-Datenbankprodukt der V9.7, V9.8 oder V10.1 den zulässigen Grenzwert überschreitet, lässt sich das DB2-Datenbankprodukt nach der Durchführung eines Upgrades auf die aktuelle Version möglicherweise nicht starten.
- Der Manager für den Speicher mit automatischer Leistungsoptimierung vergrößert den Gesamtspeicher für die Instanz nicht über die Lizenzgrenzwerte hinaus.

Fast Communications Manager (Linux und UNIX)

Fast Communications Manager (FCM) stellt die Kommunikationsunterstützung für partitionierte Datenbankumgebungen zur Verfügung.

In Umgebungen mit mehreren Membern hat jedes Member ein Paar FCM-Dämonen zur Unterstützung der mit Agentenanforderungen verbundenen Kommunikation zwischen Membern. Der eine Dämon dient zum Senden von Übertragungen, der andere zum Empfangen. Diese Dämonen und die unterstützende Infrastruktur werden aktiviert, wenn eine Instanz gestartet wird. Die FCM-Kommunikation wird außerdem für Agenten verwendet, die innerhalb desselben Members aktiv sind. Dieser Typ von Kommunikation wird auch als memberinterne Kommunikation bezeichnet.

Der FCM-Dämon erfasst Informationen zu Kommunikationsaktivitäten. Mithilfe des Datenbanksystemmonitors können Sie Informationen zur FCM-Kommunikation abrufen. Schlägt die Kommunikation zwischen Membern fehl oder wird die Kommunikation zwischen Membern wiederhergestellt, aktualisieren die FCM-Dämonen Monitorelemente mit diesen Informationen. Die FCM-Dämonen können auch die entsprechende Aktion für dieses Ereignis auslösen. Ein Beispiel für eine entsprechende Aktion ist die Rollback-Operation für eine betroffene Transaktion. Sie können den Datenbanksystemmonitor verwenden, um Unterstützung beim Einstellen der FCM-Konfigurationsparameter zu erhalten.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter **fcm_num_buffers** des Datenbankmanagers festlegen. Sie können die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter **fcm_num_channels** des Datenbankmanagers festlegen. Standardmäßig sind die Konfigurationsparameter **fcm_num_buffers** und **fcm_num_channels** des Datenbankmanagers auf den Wert AUTOMATIC gesetzt. Bei der Einstellung AUTOMATIC, die auch die empfohlene Einstellung ist, überwacht FCM die Ressourcennutzung und passt Ressourcen an den Workloadbedarf an.

Kapitel 6. Installationsvorbereitungen

Zusätzliche Tasks zur Installationsvorbereitung für eine Umgebung mit partitionierten Datenbanken (Linux und UNIX)

Aktualisieren der Umgebungseinstellungen für eine partitionierte DB2-Installation (AIX)

Diese Task beschreibt die Umgebungseinstellungen, die Sie auf allen Computern, die Ihrem partitionierten Datenbanksystem angehören sollen, aktualisieren müssen.

Vorgehensweise

Um die AIX-Umgebungseinstellungen zu aktualisieren, gehen Sie wie folgt vor:

1. Melden Sie sich am Computer als Benutzer mit Rootberechtigung an.
2. Setzen Sie das AIX-Einheitenattribut `maxuproc` (maximale Anzahl der Prozesse pro Benutzer) auf `4096`, indem Sie den folgenden Befehl eingeben:

   ```
   chdev -l sys0 -a maxuproc='4096'
   ```

   Anmerkung: Für den Wechsel zu einem 64-Bit-Kernel kann der Befehl `bosboot/reboot` erforderlich sein, wenn ein anderes Image ausgeführt wird.
3. Setzen Sie auf allen Workstations, die dem partitionierten Datenbanksystem angehören, die Parameter für das TCP/IP-Netz auf die nachfolgend aufgelisteten Werte. Diese Werte stellen die Mindestwerte für diese Parameter dar. Ist ein Netzparameter bereits auf einen höheren Wert eingestellt, lassen Sie den Wert unverändert.

   ```
   thewall       = 65536
   sb_max        = 1310720
   rfc1323       = 1
   tcp_sendspace = 221184
   tcp_recvspace = 221184
   udp_sendspace = 65536
   udp_recvspace = 65536
   ipqmaxlen     = 250
   somaxconn     = 1024
   ```

   Um die aktuellen Einstellungen aller Netzparameter aufzulisten, geben Sie den folgenden Befehl ein:

   ```
   no -a | more
   ```

   Geben Sie den folgenden Befehl ein, um einen Parameter einzustellen:

   ```
   no -o parametername=wert
   ```

   Dabei gilt Folgendes:

   - *parametername* steht für den Parameter, der eingestellt werden soll
   - *wert* steht für den Wert, der für diesen Parameter verwendet werden soll

   Geben Sie beispielsweise den folgenden Befehl ein, um den Parameter `tcp_sendspace` auf `221184` einzustellen:

   ```
   no -o tcp_sendspace=221184
   ```
4. Wenn Sie eine Hochgeschwindigkeitsverbindung verwenden, müssen Sie die Parameter `spoolsize` und `rpoolsize` für `css0` auf folgende Werte einstellen:

```
spoolsize     16777216
rpoolsize     16777216
```

Um die aktuellen Einstellungen dieser Parameter aufzulisten, geben Sie den folgenden Befehl ein:

```
lsattr -l css0 -E
```

Geben Sie die folgenden Befehle ein, um diese Parameter einzustellen:

```
/usr/lpp/ssp/css/chgcss -l css0 -a spoolsize=16777216
/usr/lpp/ssp/css/chgcss -l css0 -a rpoolsize=16777216
```

Wenn Sie die Datei /tftpboot/tuning.cst zum Optimieren Ihres Systems nicht verwenden, können Sie die Beispielscriptdatei *DB2DIR*/misc/rc.local.sample verwenden, um die Netzparameter nach der Installation zu aktualisieren. Dabei steht *DB2DIR* für den Pfad, in dem das DB2-Datenbankprodukt installiert wurde. Um die Netzparameter mithilfe der Beispielscriptdatei nach der Installation zu aktualisieren, gehen Sie wie folgt vor:

a. Kopieren Sie diese Scriptdatei in das Verzeichnis /etc und berechtigen Sie den Benutzer mit Rootberechtigung zum Ausführen der Datei. Geben Sie hierfür die folgenden Befehle ein:

   ```
   cp /usr/opt/db2_09_01/misc/rc.local.sample /etc/rc.local
   chown root:sys /etc/rc.local
   chmod 744 /etc/rc.local
   ```

b. Zeigen Sie die Datei /etc/rc.local an und aktualisieren Sie sie falls erforderlich.

c. Fügen Sie einen Eintrag zur Datei /etc/inittab hinzu, sodass das Script /etc/rc.local bei jedem Neustart der Maschine ausgeführt wird. Mit dem Befehl **mkitab** können Sie einen Eintrag zur Datei /etc/inittab hinzufügen. Geben Sie zum Hinzufügen des Eintrags den folgenden Befehl ein:

   ```
   mkitab "rclocal:2:wait:/etc/rc.local > /dev/console 2>&1"
   ```

d. Überprüfen Sie, ob der Eintrag /etc/rc.nfs in der Datei /etc/inittab vorhanden ist, indem Sie den folgenden Befehl eingeben:

   ```
   lsitab rcnfs
   ```

e. Aktualisieren Sie die Netzparameter, ohne Ihr System neu zu starten, indem Sie den folgenden Befehl eingeben:

   ```
   /etc/rc.local
   ```

5. Stellen Sie sicher, dass der Paging-Bereich für die Ausführung einer partitionierten Installation von DB2 Enterprise Server Edition groß genug ist. Falls der Paging-Bereich nicht ausreicht, bricht das Betriebssystem den Prozess, der am meisten virtuellen Speicher verwendet, mit dem Befehl 'kill' ab. Dies wäre wahrscheinlich einer der DB2-Prozesse. Um die Größe des verfügbaren Paging-Bereichs zu überprüfen, geben Sie den folgenden Befehl ein:

   ```
   lsps -a
   ```

   Die von diesem Befehl zurückgegebene Ausgabe sieht etwa wie folgt aus:

   ```
   Page Space  Physical Volume  Volume Group  Size  %Used  Active  Auto  Type
   paging00    hdisk1           rootvg        60MB     19     yes   yes    lv
   hd6         hdisk0           rootvg        60MB     21     yes   yes    lv
   hd6         hdisk2           rootvg        64MB     21     yes   yes    lv
   ```

   Der verfügbare Paging-Bereich sollte doppelt so groß sein wie der auf dem Computer installierte physische Speicher.

6. Wenn Sie ein kleines bis mittelgroßes partitioniertes Datenbanksystem erstellen, sollte die Anzahl der Network File System-Dämonen (NFSDs) auf dem Computer, dem die Instanz gehört, etwa dem folgenden Wert entsprechen:

   ```
   Anzahl der biod-Prozesse auf einem Computer * Anzahl der Computer in einer Instanz
   ```

   Idealerweise sollten auf jedem Computer 10 biod-Prozesse ausgeführt werden. Nach der angegebenen Formel werden auf einem System mit vier Computern mit jeweils 10 biod-Prozessen daher 40 NFSDs verwendet.

   Wenn Sie ein größeres System installieren, können Sie bis zu 120 NFSDs auf dem Computer verwenden.

   Weitere Informationen zu NFS finden Sie in der NFS-Dokumentation.

Einrichten eines Arbeitsverbunds zum Verteilen von Befehlen an mehrere AIX-Knoten

In einer Umgebung mit partitionierten Datenbanken unter AIX können Sie einen Arbeitsverbund einrichten, um Befehle an die Gruppe von System p SP-Workstations zu verteilen, die Ihrem partitionierten Datenbanksystem angehören. Befehle können mithilfe des Befehls **dsh** an die Workstations verteilt werden.

Vorbereitende Schritte

Diese Funktion kann beim Installieren und Verwalten eines partitionierten Datenbanksystems unter AIX nützlich sein, da auf diese Weise dieselben Befehle schnell auf allen Computern ausgeführt werden können und gleichzeitig weniger Möglichkeiten bestehen, hierbei einen Fehler zu machen.

Sie müssen den Hostnamen jedes Computers kennen, den Sie in den Arbeitsverbund aufnehmen wollen.

Sie müssen als Benutzer mit Rootberechtigung an der Steuerworkstation angemeldet sein.

Eine Datei muss verfügbar sein, in der die Hostnamen aller Workstations aufgelistet werden, die dem partitionierten Datenbanksystem angehören sollen.

Vorgehensweise

Gehen Sie wie folgt vor, um den Arbeitsverbund so zu definieren, dass Befehle an eine Liste mit Workstations verteilt werden:

1. Erstellen Sie eine Datei mit dem Namen `nodelist.txt`; in dieser Datei werden die Hostnamen aller Workstations aufgelistet, die dem Arbeitsverbund angehören sollen.

   Angenommen, Sie wollen einen Arbeitsverbund mit den beiden Workstations `workstation1` und `workstation2` erstellen. Der Inhalt der Datei `nodelist.txt` wäre in diesem Fall:

   ```
   workstation1
   workstation2
   ```

2. Aktualisieren Sie die Umgebungsvariable des Arbeitsverbunds. Geben Sie den folgenden Befehl ein, um diese Liste zu aktualisieren:

   ```
   export DSH_NODE_LIST=Pfad/nodelist.txt
   ```

   Dabei ist *Pfad* die Speicherposition, an der die Datei `nodelist.txt` erstellt wurde, und `nodelist.txt` ist der Name der von Ihnen erstellten Datei, in der die Workstations in dem Arbeitsverbund aufgelistet werden.

3. Prüfen Sie, ob die Namen im Arbeitsverbund tatsächlich den gewünschten Workstations entsprechen, indem Sie folgenden Befehl eingeben:

   ```
   dsh -q
   ```

   Die zurückgegebene Ausgabe sieht etwa wie folgt aus:

   ```
   Working collective file /nodelist.txt:
      workstation1
      workstation2
      Fanout: 64
   ```

Prüfen, ob NFS aktiv ist (Linux und UNIX)

Vor dem Einrichten einer Umgebung mit partitionierten Datenbanken müssen Sie prüfen, ob Network File System (NFS) auf allen Computern aktiv ist, die dem partitionierten Datenbanksystem angehören sollen.

Vorgehensweise

Zum Überprüfen, ob NFS auf jedem Computer ausgeführt wird, gehen Sie wie folgt vor:

- AIX-Betriebssysteme

  Geben Sie auf jedem Computer den folgenden Befehl ein:

  ```
  lssrc -g nfs
  ```

  Das Feld `Status` für die NFS-Prozesse sollte `active` anzeigen. Nachdem Sie geprüft haben, ob NFS auf jedem System aktiv ist, müssen Sie nach den spezifischen NFS-Prozessen suchen, die für DB2-Datenbankprodukte erforderlich sind. Hierbei handelt es sich um folgende Prozesse:

  ```
  rpc.lockd
       rpc.statd
  ```

- HP-UX- und Solaris-Betriebssysteme:

  Geben Sie auf jedem Computer den folgenden Befehl ein:

  ```
  showmount -e hostname
  ```

  Geben Sie den Befehl **showmount** ohne den Parameter *hostname* ein, um das lokale System zu prüfen. Ist NFS nicht aktiv, wird eine Nachricht ähnlich der folgenden zurückgegeben:

  ```
  showmount: ServerA: RPC: Program not registered (Programm ist nicht registriert)
  ```

  Nachdem Sie geprüft haben, ob NFS auf jedem System aktiv ist, müssen Sie nach den spezifischen NFS-Prozessen suchen, die für DB2-Datenbankprodukte erforderlich sind:

  ```
  rpc.lockd
       rpc.statd
  ```

  Sie können diese Prozesse mithilfe der folgenden Befehle suchen:

  ```
  ps -ef | grep rpc.lockd
       ps -ef | grep rpc.statd
  ```

- Linux-Betriebssysteme:

  Geben Sie auf jedem Computer den folgenden Befehl ein:

  ```
  showmount -e hostname
  ```

  Geben Sie den Befehl **showmount** ohne den Parameter *hostname* ein, um das lokale System zu prüfen.

Ist NFS nicht aktiv, wird eine Nachricht ähnlich der folgenden zurückgegeben:

```
showmount: ServerA: RPC: Program not registered (Programm ist nicht registriert)
```

Nachdem Sie geprüft haben, ob NFS auf jedem System aktiv ist, müssen Sie nach den spezifischen NFS-Prozessen suchen, die für DB2-Datenbankprodukte erforderlich sind. Der Prozess `rpc.statd` ist erforderlich.

Mithilfe der Befehle **ps -ef | grep rpc.statd** kann nach diesem Prozess gesucht werden.

Sind diese Prozesse nicht aktiv, entnehmen Sie bitte weitere Informationen der Dokumentation des Betriebssystems.

Prüfen der Verfügbarkeit des Portbereichs auf zugehörigen Computern (Linux und UNIX)

Diese Task beschreibt die erforderlichen Schritte zum Prüfen der Verfügbarkeit des Anschlussbereichs auf zugehörigen Computern. Der Anschlussbereich wird von Fast Communications Manager (FCM) verwendet. FCM ist eine Funktion von DB2, die die Kommunikation zwischen Datenbankpartitionsservern steuert.

Vorbereitende Schritte

Das Prüfen der Verfügbarkeit des Portbereichs auf zugehörigen Computern sollte erfolgen, nachdem Sie den Datenbankpartitionsserver installiert haben, dem die Instanz gehört und bevor Sie zugehörige Datenbankpartitionsserver installieren.

Wenn Sie den Datenbankpartitionsserver, dem die Instanz gehört, auf der primären Maschine installieren, reserviert DB2 einen Portbereich entsprechend der angegebenen Anzahl logischer Datenbankpartitionsserver, die der Umgebung mit partitionierten Datenbanken angehören. Der Standardbereich besteht aus vier Ports. Für jeden Server, der der Umgebung mit partitionierten Datenbanken angehört, müssen Sie die Datei `/etc/services` für die FCM-Ports manuell konfigurieren. Der Bereich der FCM-Ports ist davon abhängig, wie viele logische Partitionen auf dem zugehörigen Computer verwendet werden sollen. Mindestens die beiden Einträge `DB2_`*`instanz`* und `DB2_`*`instanz`*`_END` sind erforderlich. Weitere Anforderungen für die auf den zugehörigen Computern angegebenen FCM-Ports:

- Die Anfangsportnummer muss mit der Anfangsportnummer des primären Computers übereinstimmen.
- Weitere Ports müssen fortlaufend nummeriert werden.
- Die angegebenen Portnummern müssen frei sein.

Zum Vornehmen von Änderungen an der Datei `services` benötigen Sie Rootberechtigung.

Vorgehensweise

Um die Verfügbarkeit des Portbereichs auf zugehörigen Computern zu prüfen, gehen Sie wie folgt vor:

1. Öffnen Sie die Datei `services` im Verzeichnis `/etc/services`.
2. Suchen Sie die Ports, die für DB2 Fast Communications Manager (FCM) reserviert sind. Die Einträge sollten ungefähr wie im folgenden Beispiel aussehen:

```
DB2_db2inst1         60000/tcp
DB2_db2inst1_1       60001/tcp
DB2_db2inst1_2       60002/tcp
DB2_db2inst1_END     60003/tcp
```

DB2 reserviert die ersten vier verfügbaren Anschlüsse nach 60000.

3. Öffnen Sie auf jedem zugehörigen Computer die Datei `services` und prüfen Sie, ob die für DB2 FCM reservierten Anschlüsse in der Datei 'services' des Primärcomputers tatsächlich nicht verwendet werden.
4. Falls die erforderlichen Ports auf einem zugehörigen Computer bereits verwendet werden, ermitteln Sie einen verfügbaren Portbereich für alle Computer und aktualisieren Sie jeweils alle Servicedateien, einschließlich der entsprechenden Datei auf dem Primärcomputer.

Nächste Schritte

Nachdem Sie den Datenbankpartitionsserver, dem die Instanz gehört, auf der primären Maschine installiert haben, müssen Sie Ihr DB2-Datenbankprodukt auf den zugehörigen Datenbankpartitionsservern installieren. Sie können die für die Partitionierungsserver generierte Antwortdatei verwenden (Standardname ist `db2ese_addpart.rsp`), Sie müssen jedoch die Dateien in `/etc/services` für die FCM-Ports manuell konfigurieren. Der Bereich der FCM-Ports ist davon abhängig, wie viele logische Partitionen auf dem aktuellen Computer verwendet werden sollen. Für `DB2_` und `DB2__END` sind mindestens zwei Einträge mit anschließenden fortlaufenden freien Portnummern erforderlich. Die auf den einzelnen zugehörigen Maschinen verwendeten FCM-Portnummern müssen über dieselben Anfangsportnummern verfügen und die weiteren Ports müssen fortlaufend nummeriert werden.

Erstellen eines Dateisystems für ein partitioniertes Datenbanksystem (Linux)

Im Rahmen der Konfiguration Ihres partitionierten Datenbanksystems unter Linux-Betriebssystemen müssen Sie ein DB2-Ausgangsdateisystem erstellen. Dann müssen Sie das Ausgangsdateisystem über NFS exportieren und für jeden Computer, der am partitionierten Datenbanksystem beteiligt ist, anhängen.

Informationen zu diesem Vorgang

Sie benötigen ein Dateisystem, auf das alle Maschinen, die dem partitionierten Datenbanksystem angehören, zugreifen können. Dieses Dateisystem wird als Ausgangsverzeichnis der Instanz verwendet.

Für Konfigurationen, die mehr als eine Maschine für eine einzelne Datenbankinstanz verwenden, wird NFS (Network File System) verwendet, um dieses Dateisystem gemeinsam benutzbar zu machen. Normalerweise wird eine Maschine im Cluster verwendet, um das Dateisystem mit NFS zu exportieren, und die übrigen Maschinen im Cluster hängen das NFS-Dateisystem von dieser Maschine aus an. Auf der Maschine, die das Dateisystem exportiert, wird das Dateisystem lokal angehängt.

Die Dokumentation der entsprechenden Linux-Variante enthält weitere Informationen zu Befehlen.

Vorgehensweise

Um das DB2-Ausgangsdateisystem zu erstellen sowie in NFS zu exportieren und anzuhängen, gehen Sie wie folgt vor:

1. Wählen Sie auf einer Maschine eine Plattenpartition aus oder erstellen Sie eine Partition mithilfe des Befehls **`fdisk`**.

2. Verwenden Sie ein Dienstprogramm wie **mkfs**, und erstellen Sie auf dieser Partition ein Dateisystem. Das Dateisystem sollte groß genug sein, um die erforderlichen DB2-Programmdateien aufnehmen zu können, und über ausreichend Speicherkapazität für Ihre Datenbankanforderungen verfügen.
3. Hängen Sie das soeben erstellte Dateisystem lokal an und fügen Sie einen Eintrag zur Datei /etc/fstab hinzu, damit dieses Dateisystem bei jedem Neustart des Systems angehängt wird. Beispiel:

   ```
   /dev/hda1    /db2home    ext3   defaults   1 2
   ```

4. Fügen Sie einen Eintrag zur Datei /etc/exports hinzu, damit ein NFS-Dateisystem unter Linux bei jedem Neustart automatisch exportiert wird. In diesem Eintrag müssen alle Hostnamen, die dem Cluster angehören, sowie alle Namen, unter denen die Maschine bekannt sein könnte, enthalten sein. Stellen Sie außerdem sicher, dass jede Maschine im Cluster über Rootberechtigung für das exportierte Dateisystem verfügt, indem Sie die Option root" verwenden.

   Die Datei /etc/exports liegt im ASCII-Format vor und enthält folgende Informationen:

   ```
   /db2home  name_maschine1(rw)  name_maschine2(rw)
   ```

   Um das NFS-Verzeichnis zu exportieren, führen Sie folgenden Befehl aus:

   ```
   /usr/sbin/exportfs -r
   ```

5. Fügen Sie auf allen übrigen Maschinen im Cluster einen Eintrag zur Datei /etc/fstab hinzu, damit das Dateisystem beim Systemstart automatisch über NFS angehängt wird. Stellen Sie bei der Angabe der Optionen für den Mountpunkt sicher, dass das Dateisystem beim Systemstart angehängt wird, dass es über den Zugriff rw (Lesen und Schreiben) verfügt, dass ein absoluter Mount (Hard Mount) ausgeführt wird, dass die Option bg (Hintergrund) verwendet wird und dass **setuid**-Programme korrekt ausgeführt werden können. Diese Optionen werden im folgenden Beispiel dargestellt.

   ```
   fusion-en:/db2home /db2home nfs rw,timeo=7,
         hard,intr,bg,suid,lock
   ```

   Hierbei ist *fusion-en* der Name der Maschine.
6. Hängen Sie das exportierte Dateisystem über NFS auf allen verbleibenden Maschinen im Cluster an. Geben Sie den folgenden Befehl ein:

   ```
   mount /db2home
   ```

   Schlägt der Befehl **mount** fehl, können Sie den Befehl **showmount** verwenden, um den Status des NFS-Servers zu überprüfen. Beispiel:

   ```
   showmount -e fusion-en
   ```

   Hierbei ist *fusion-en* der Name der Maschine.

   Diese Version des Befehls **showmount** sollte die Dateisysteme auflisten, die aus der Maschine mit dem Namen fusion-en exportiert wurden. Wenn dieser Befehl fehlschlägt, wurde möglicherweise der NFS-Server nicht gestartet. Um den Server manuell zu starten, führen Sie den folgenden Befehl als Benutzer mit Rootberechtigung auf dem NFS-Server aus:

   ```
   /etc/rc.d/init.d/nfs restart
   ```

   Vorausgesetzt, die derzeitige Ausführungsebene ist 3, können Sie diesen Befehl beim Systemstart automatisch ausführen lassen, indem Sie im nachfolgenden Verzeichnis K20nfs in S20nfs umbenennen: /etc/rc.d/rc3.d.

Ergebnisse

Mit diesen Schritten haben Sie die folgenden Tasks ausgeführt:

1. Sie haben auf einem einzelnen Computer in der partitionierten Datenbankumgebung ein Dateisystem erstellt, das als Instanz- und Ausgangsverzeichnis verwendet wird.
2. Im Falle einer Konfiguration, die mehr als eine Maschine für eine einzelne Datenbankinstanz verwendet, haben Sie dieses Dateisystem über NFS exportiert.
3. Sie haben das exportierte Dateisystem auf jedem zugehörigen Computer im System angehängt.

Erstellen eines DB2-Ausgangsdateisystems für ein partitioniertes Datenbanksystem (AIX)

Im Rahmen der Konfiguration Ihres partitionierten Datenbanksystems müssen Sie ein DB2-Ausgangsdateisystem erstellen. Dann müssen Sie das Ausgangsdateisystem über NFS exportieren und für jeden Computer, der am partitionierten Datenbanksystem beteiligt ist, anhängen.

Vorbereitende Schritte

Es wird empfohlen, ein Ausgangsdateisystem zu erstellen, das mindestens den Inhalt der DB2-Datenbankprodukt-DVD aufnehmen kann. Sie können die Größe (angegeben in KB) mit dem folgenden Befehl ermitteln:

```
du -sk DVD-Mountpunkt
```

Eine DB2-Instanz benötigt einen Speicherbereich von mindestens 200 MB. Falls der freie Speicherbereich nicht ausreicht, können Sie die DB2-Datenbankprodukt-DVD auch über jeden Computer im System anhängen, anstatt ihren Inhalt auf die Festplatte zu kopieren.

Sie müssen:

- über die Berechtigung `Root` verfügen, um ein Dateisystem erstellen zu können.
- eine Datenträgergruppe erstellt haben, in der Ihr Dateisystem physisch vorhanden sein soll.

Vorgehensweise

Um das DB2-Ausgangsdateisystem zu erstellen sowie in NFS zu exportieren und anzuhängen, gehen Sie wie folgt vor:

1. Erstellen Sie das DB2-Ausgangsdateisystem.

   Melden Sie sich als Benutzer mit der Berechtigung `Root` am Primärcomputer (ServerA) Ihres partitionierten Datenbanksystems an, und erstellen Sie ein Ausgangsdateisystem mit dem Namen `/db2home` für dieses partitionierte Datenbanksystem.

   a. Geben Sie den Befehl **`smit jfs`** ein.
   b. Klicken Sie das Symbol **Add a Journaled File System** an.
   c. Klicken Sie das Symbol **Add a Standard Journaled File System** an.
   d. Wählen Sie in der Liste **Volume Group Name** eine vorhandene Datenträgergruppe aus, in der sich das Dateisystem physisch befinden soll.
   e. Definieren Sie das Feld für die Größe des Dateisystems **SIZE of file system (in 512-byte blocks) (Num.)**. Diese Größe ist nach 512-Byte-Blöcken nummeriert. Wenn Sie lediglich ein Dateisystem für das Ausgangsverzeichnis

der Instanz erstellen müssen, empfiehlt es sich, einen Wert von 180 000 (etwa 90 MB) zu verwenden. Wenn Sie für die Installation das Image der Produkt-DVD kopieren müssen, sollten Sie das Dateisystem mit einem Wert von 2 000 000 (etwa 1 GB) erstellen.

f. Geben Sie im Feld **MOUNT POINT** einen Mountpunkt für dieses Dateisystem ein. Im vorliegenden Beispiel ist der Mountpunkt `/db2home`.

g. Setzen Sie das Feld **Mount AUTOMATICALLY at system restart** auf `yes`.

   Für die übrigen Felder können die Standardeinstellungen belassen werden.

h. Klicken Sie **OK** an.

2. Exportieren Sie das DB2-Ausgangsdateisystem.

   Exportieren Sie das Dateisystem `/db2home` über NFS, damit alle Computer, die dem partitionierten Datenbanksystem angehören sollen, darauf zugreifen können.

   a. Geben Sie den Befehl **`smit nfs`** ein.

   b. Klicken Sie das Symbol **Network File System (NFS)** an.

   c. Klicken Sie das Symbol **Add a Directory to Exports List** an.

   d. Geben Sie den Pfadnamen und das zu exportierende Verzeichnis (beispielsweise `/db2home`) in das Feld **PATHNAME of directory to export** ein.

   e. Geben Sie in das Feld **HOSTS allowed root access** den Namen aller Workstations ein, die dem partitionierten Datenbanksystem angehören sollen. Verwenden Sie ein Komma (,) als Begrenzer zwischen den einzelnen Namen. Beispiel: `ServerA, ServerB, ServerC`. Falls Sie eine Hochgeschwindigkeitsverbindung verwenden, wird empfohlen, dass Sie in diesem Feld auch die Namen der einzelnen Workstations in der Hochgeschwindigkeitsverbindung angeben. Für die übrigen Felder können die Standardeinstellungen belassen werden.

   f. Klicken Sie **OK** an.

3. Melden Sie sich ab.

4. Hängen Sie das DB2-Ausgangsdateisystem für jeden beteiligten Computer an.

   Melden Sie sich an *jedem* Computer (ServerB, ServerC, ServerD), der dem partitionierten Datenbanksystem angehören soll, an und hängen Sie das soeben exportierte Dateisystem über NFS an. Führen Sie hierzu die folgenden Schritte aus:

   a. Geben Sie den Befehl **`smit nfs`** ein.

   b. Klicken Sie das Symbol **Network File System (NFS)** an.

   c. Klicken Sie das Symbol **Add a File System for Mounting** an.

   d. Geben Sie im Feld **PATHNAME of the mount point (Path)** den Pfadnamen des Mountpunkts ein.

      Der Pfadname des Mountpunkts ist die Speicherposition, an der Sie das DB2-Ausgangsverzeichnis erstellen sollten. Verwenden Sie in diesem Beispiel `/db2home`.

   e. Geben Sie im Feld **PATHNAME of the remote directory** den Pfadnamen des fernen Verzeichnisses ein.

      Im vorliegenden Beispiel sollten Sie den gleichen Wert eingeben wie im Feld **PATHNAME of the mount point (Path)**.

   f. Geben Sie im Feld **HOST where the remote directory resides** den Hostnamen (*hostname*) der Maschine ein, auf die Sie das Dateisystem exportiert haben.

      Dies ist der Hostname der Maschine, auf der das Dateisystem, das Sie anhängen, erstellt wurde.

Zum Verbessern der Leistung ist es möglicherweise empfehlenswert, das soeben erstellte Dateisystem über eine Hochgeschwindigkeitsverbindung über NFS anzuhängen. Wenn Sie das Dateisystem über eine Hochgeschwindigkeitsverbindung anhängen wollen, müssen Sie seinen Namen im Feld **HOST where remote directory resides** eingeben.

Bitte beachten Sie, dass eine eventuelle Nichtverfügbarkeit der Hochgeschwindigkeitsverbindung dazu führt, dass jede Workstation, die dem partitionierten Datenbanksystem angehört, den Zugriff auf das DB2-Ausgangsverzeichnis verliert.

g. Setzen Sie den Wert im Feld **MOUNT now, add entry to /etc/filesystems or both?** auf `both` (beide).

h. Setzen Sie den Wert im Feld **/etc/filesystems entry will mount the directory on system RESTART** auf `yes`.

i. Setzen Sie den Wert im Feld **MODE for this NFS file system** auf `read-write`.

j. Setzen Sie den Wert im Feld **Mount file system soft or hard** auf `hard`.

Bei einem bedingten Mount (Soft Mount) versucht der Computer *nicht* auf unbegrenzte Zeit, das Verzeichnis fern anzuhängen. Bei einem absoluten Mount (Hard Mount) versucht die Maschine auf unbegrenzte Zeit, das Verzeichnis anzuhängen. Dies könnte bei einem Systemabsturz zu Problemen führen. Es wird empfohlen, dieses Feld auf `hard` (absoluter Mount) zu setzen.

Für die übrigen Felder können die Standardeinstellungen belassen werden.

k. Stellen Sie sicher, dass das Dateisystem mit dem Wert `Yes` im Feld **Allow execution of SUID and sgid programs in this file system?** angehängt wird. Dies ist die Standardeinstellung.

l. Klicken Sie **OK** an.

m. Melden Sie sich ab.

Erforderliche Benutzer für eine Installation von DB2 pureScale Feature (Linux)

Für den Betrieb einer DB2-Datenbankumgebung unter Linux sind zwei Benutzer und zwei Gruppen erforderlich.

Vorbereitende Schritte

- Um Benutzer und Gruppen erstellen zu können, müssen Sie über Rootberechtigung verfügen.
- Wenn Sie Benutzer und Gruppen mit Sicherheitssoftware verwalten, sind beim Definieren von DB2-Benutzern und -Gruppen möglicherweise zusätzliche Schritte erforderlich.

Informationen zu diesem Vorgang

Sie benötigen zwei Benutzer, um die DB2 pureScale-Instanz erstellen zu können:

- Ein Benutzername für den Instanzeigner
- Ein Benutzername für den abgeschirmten Benutzer

Verwenden Sie zwei unterschiedliche Benutzernamen mit zwei verschiedenen Gruppen. Beide Benutzer müssen auf allen Hosts über dieselbe Benutzer-ID, denselben Gruppennamen und dasselbe Ausgangsverzeichnis verfügen. Stellen Sie sicher, dass alle Benutzernamen, die Sie verwenden möchten, auf allen Hosts vorliegen und über dieselben Eigenschaften verfügen. Diese erforderlichen Benutzer

müssen nicht unbedingt bereits vor der Installation vorhanden sein. Sie können sie beim Ausfüllen der Anzeigen des DB2-Installationsassistenten erstellen oder in Ihrer Antwortdatei angeben. Werden bereits vorhandene Benutzer verwendet, müssen diese Benutzer auf allen Hosts vorhanden sein und die angegebenen Anforderungen erfüllen.

Die in den folgenden Anweisungen verwendeten Benutzer- und Gruppennamen stellen die Standardnamen dar und werden in der folgenden Tabelle aufgelistet. Sie können Ihre eigenen Benutzer- und Gruppennamen angeben, sofern diese den Namensregeln für Ihr System und für DB2 entsprechen.

Tabelle 10. Standardbenutzer und -gruppen

Erforderlicher Benutzer	Benutzername	Gruppenname
Instanzeigner	db2sdin1	db2iadm1
Abgeschirmter Benutzer	db2sdfe1	db2fadm1

Die folgende Tabelle enthält die Namen der Benutzer und Gruppen, die in den nachstehenden Anweisungen verwendet werden. Sie können Ihre eigenen Benutzer- und Gruppennamen angeben, sofern diese den Namensregeln für Ihr System und für DB2 entsprechen.

Wenn Sie beabsichtigen, den DB2-Installationsassistenten zum Installieren des DB2-Datenbankprodukts zu verwenden, werden diese Benutzer vom DB2-Installationsassistenten erstellt.

Einschränkungen

Die von Ihnen erstellten Benutzernamen müssen sowohl den Namensregeln Ihres Betriebssystems als auch den Namensregeln des DB2-Datenbanksystems entsprechen.

Der Benutzername, den Sie auf verschiedenen Hosts erstellen werden, muss über dasselbe Ausgangsverzeichnis verfügen. Die Benutzernamen müssen jedoch nicht bereits auf den Hosts vorhanden sein. Werden bereits vorhandene Benutzernamen verwendet, müssen diese Namen auf allen Hosts mit derselben Benutzer-ID (uid) und Gruppen-ID (gid) sowie demselben Gruppennamen und `Ausgangsverzeichnis` vorhanden sein.

Vorgehensweise

Führen Sie die folgenden Schritte aus, um diese Benutzer zu erstellen:

1. Melden Sie sich am Host an.
2. Erstellen Sie eine Gruppe für den Instanzeigner (z. B. `db2iadm1`) und eine Gruppe, die UDFs oder gespeicherte Prozeduren ausführt (z. B. `db2fadm1`), indem Sie die folgenden Befehle eingeben:

   ```
   groupadd -g 999 db2iadm1
   groupadd -g 998 db2fadm1
   ```

   Stellen Sie sicher, dass die spezifischen Nummern, die Sie verwenden, derzeit auf keiner der Maschinen vorhanden sind.
3. Erstellen Sie einen Benutzer für jede Gruppe, die Sie im vorigen Schritt erstellt haben. Verwenden Sie hierzu die nachstehenden Befehle. Das Ausgangsverzeichnis für alle Benutzer ist das DB2-Ausgangsverzeichnis, das Sie zuvor erstellt und zur gemeinsamen Benutzung verfügbar gemacht haben (`db2home`).

```
useradd -u 1004 -g db2iadm1 -m -d /db2home/db2inst1 db2inst1
useradd -u 1003 -g db2fadm1 -m -d /db2home/db2fenc1 db2fenc1
```

4. Definieren Sie ein Anfangskennwort für jeden Benutzer, den Sie erstellt haben, indem Sie die folgenden Befehle eingeben:

   ```
   passwd db2inst1    passwd db2fenc1
   ```

5. Melden Sie sich ab.
6. Melden Sie sich am Primärcomputer unter jedem der von Ihnen erstellten Benutzernamen an (`db2inst1` und `db2fenc1`). Sie erhalten für jeden Benutzernamen möglicherweise die Aufforderung, das Kennwort zu ändern, da diese Namen zum ersten Mal am System angemeldet werden.
7. Melden Sie sich ab.
8. Erstellen Sie auf jedem Computer, der Ihrer Datenbankumgebung angehören soll, genau dieselben Benutzer- und Gruppenkonten.

Erstellen erforderlicher Benutzer für die Installation eines DB2-Servers in einer Umgebung mit partitionierten Datenbanken (AIX)

Für den Betrieb von DB2-Datenbanken in Umgebungen mit partitionierten Datenbank unter AIX sind drei Benutzernamen und drei Gruppennamen erforderlich.

Vorbereitende Schritte

- Um Benutzer und Gruppen erstellen zu können, müssen Sie über Rootberechtigung verfügen.
- Wenn Sie Benutzer und Gruppen mit Sicherheitssoftware verwalten, sind beim Definieren von DB2-Benutzern und -Gruppen möglicherweise zusätzliche Schritte erforderlich.

Informationen zu diesem Vorgang

Die folgende Tabelle enthält die Namen der Benutzer und Gruppen, die in den nachstehenden Anweisungen verwendet werden. Sie können Ihre eigenen Benutzer- und Gruppennamen angeben, sofern diese den Namensregeln für Ihr System und für DB2 entsprechen.

Wenn Sie beabsichtigen, den DB2-Installationsassistenten zum Installieren des DB2-Datenbankprodukts zu verwenden, werden diese Benutzer vom DB2-Installationsassistenten erstellt.

Tabelle 11. Erforderliche Benutzer und Gruppen

Erforderlicher Benutzer	Benutzername	Gruppenname
Instanzeigner	db2inst1	db2iadm1
Abgeschirmter Benutzer	db2fenc1	db2fadm1
Benutzer des DB2-Verwaltungsservers	dasusr1	dasadm1

Falls der Benutzername für den DB2-Verwaltungsserver bereits vorhanden ist, muss er vor der Installation auf allen zugehörigen Computern vorhanden sein. Wenn Sie den **DB2-Installationsassistenten** verwenden, um einen neuen Benutzer für den DB2-Verwaltungsserver auf dem Computer zu erstellen, der als Instanzeigner fungiert, wird dieser Benutzer (falls erforderlich) bei Installationen mit einer

Antwortdatei auf den zugehörigen Computern ebenfalls erstellt. Ist der Benutzername auf den zugehörigen Computern bereits vorhanden, muss er über dieselbe Primärgruppe verfügen.

Einschränkungen

Die von Ihnen erstellten Benutzernamen müssen sowohl den Namensregeln Ihres Betriebssystems als auch den Namensregeln des DB2-Datenbanksystems entsprechen.

Vorgehensweise

Führen Sie die folgenden Schritte aus, um die drei genannten Benutzer zu erstellen:

1. Melden Sie sich am Primärcomputer an.
2. Erstellen Sie eine Gruppe für den Instanzeigner (z. B. `db2iadm1`), die Gruppe, die UDFs oder gespeicherte Prozeduren ausführt (z. B. `db2fadm1`), und die Gruppe, der der DB2-Verwaltungsserver angehören soll (z. B. `dasadm1`). Geben Sie hierzu die folgenden Befehle ein:

   ```
   mkgroup id=999 db2iadm1
   mkgroup id=998 db2fadm1
   mkgroup id=997 dasadm1
   ```

3. Erstellen Sie einen Benutzer für jede Gruppe, die Sie im vorigen Schritt erstellt haben. Verwenden Sie hierzu die nachstehenden Befehle. Das Ausgangsverzeichnis für alle Benutzer ist das DB2-Ausgangsverzeichnis, das Sie zuvor erstellt und zur gemeinsamen Benutzung verfügbar gemacht haben (`db2home`).

   ```
   mkuser id=1004 pgrp=db2iadm1 groups=db2iadm1 home=/db2home/db2inst1
         core=-1 data=491519 stack=32767 rss=-1 fsize=-1 db2inst1
   mkuser id=1003 pgrp=db2fadm1 groups=db2fadm1 home=/db2home/db2fenc1
      db2fenc1
   mkuser id=1002 pgrp=dasadm1 groups=dasadm1 home=/home/dasusr1
      dasusr1
   ```

4. Definieren Sie ein Anfangskennwort für jeden Benutzernamen, den Sie erstellt haben, indem Sie die folgenden Befehle eingeben:

   ```
   passwd db2inst1
   passwd db2fenc1
   passwd dasusr1
   ```

5. Melden Sie sich ab.
6. Melden Sie sich am Primärcomputer als jeder der von Ihnen erstellten Benutzer an (`db2inst1`, `db2fenc1` und `dasusr1`). Sie erhalten für jeden Benutzernamen möglicherweise die Aufforderung, das Kennwort zu ändern, da diese Namen zum ersten Mal am System angemeldet werden.
7. Melden Sie sich ab.
8. Erstellen Sie auf jedem Computer, der Ihrer Umgebung mit partitionierten Datenbanken angehören soll, genau dieselben Benutzer- und Gruppenkonten.

Kapitel 7. Installieren des DB2-Serverprodukts

Einrichten einer Umgebung mit partitionierten Datenbanken

In diesem Abschnitt wird beschrieben, wie Sie eine Umgebung mit partitionierten Datenbanken einrichten. Sie verwenden den **DB2-Installationsassistenten**, um den Datenbankserver zu installieren, der Eigner der Instanz ist, und um die Antwortdateien zu erstellen, die wiederum zum Erstellen der zugehörigen Datenbankserver verwendet werden.

Vorbereitende Schritte

Anmerkung: Eine Umgebung mit partitionierten Datenbanken wird in nicht als Root ausgeführten Installationen nicht unterstützt.

- Stellen Sie sicher, dass Sie über den Lizenzschlüssel der Aktivierungs-CD für InfoSphere Warehouse verfügen, der auf alle beteiligten Computer kopiert werden muss.
- Auf jedem Computer, der der Umgebung mit partitionierten Datenbanken angehören soll, muss die gleiche Anzahl aufeinanderfolgender Ports frei sein. Wenn die Umgebung mit partitionierten Datenbanken beispielsweise vier Computer umfassen soll, müssen auf jedem der vier Computer die gleichen vier aufeinanderfolgenden Ports frei sein. Bei der Instanzerstellung wird eine bestimmte Anzahl von Ports, die der Anzahl der logischen Partitionen auf dem aktuellen Server entspricht, im Verzeichnis `/etc/services` unter Linux und UNIX bzw. im Verzeichnis `%SystemRoot%\system32\drivers\etc\services` unter Windows reserviert. Diese Ports werden von Fast Communications Manager (FCM) verwendet. Die reservierten Ports besitzen das folgende Format:

  ```
  DB2_InstanceName
     DB2_InstanceName_1
     DB2_InstanceName_2
     DB2_InstanceName_END
  ```

  Lediglich die Einträge des Ports am Anfang (DB2_InstanceName) und am Ende (DB2_InstanceName_END) sind verbindlich. Die anderen Einträge werden in der Servicedatei reserviert, damit die betreffenden Ports nicht von anderen Anwendungen genutzt werden.
- Zur Unterstützung mehrerer beteiligter DB2-Datenbankserver muss der Computer, auf dem Sie DB2 installieren möchten, zu einer Domäne gehören, auf die Zugriff besteht. Sie können diesem Computer jedoch auch dann lokale Partitionen hinzufügen, wenn er keiner Domäne angehört.
- Unter Linux- und UNIX-Systemen ist für partitionierte Datenbanksysteme ein Dienstprogramm für eine ferne Shell erforderlich. DB2-Datenbanksysteme unterstützen die folgenden Dienstprogramme für ferne Shells:
  - rsh
  - ssh

  DB2-Datenbanksysteme verwenden standardmäßig 'rsh' für die Ausführung von Befehlen auf fernen DB2-Knoten, beispielsweise zum Starten einer fernen DB2-Datenbankpartition. Zum Verwenden des DB2-Standards muss das Paket 'rsh-server' installiert sein. Weitere Informationen finden Sie im Abschnitt „Sicherheitsaspekte bei der Installation und Verwendung des DB2-Datenbankmanagers" im Handbuch *Datenbanksicherheit*.

Wenn Sie das Dienstprogramm 'rsh' für die ferne Shell ausführen möchten, muss auch 'inetd' (oder 'xinetd') installiert und aktiv sein. Wenn Sie das Dienstprogramm 'ssh' für die ferne Shell ausführen möchten, müssen Sie die Registrierdatenbankvariable **DB2RSHCMD** sofort nach Abschluss der Installation von DB2 festlegen. Wenn diese Registrierdatenbankvariable nicht eingestellt wird, wird 'rsh' verwendet.

- Stellen Sie unter Linux- und UNIX-Betriebssystemen sicher, dass die Datei `hosts` im Verzeichnis `etc` keinen Eintrag für „127.0.0.2" enthält, wenn die IP-Adresse dem vollständig qualifizierten Hostnamen der Maschine zugeordnet ist.

Informationen zu diesem Vorgang

Eine Datenbankpartition ist Bestandteil einer Datenbank, die aus eigenen Daten, Indizes, Konfigurationsdateien und Transaktionsprotokollen besteht. Eine partitionierte Datenbank ist eine Datenbank, die aus mindestens zwei Partitionen besteht.

Vorgehensweise

Um eine Umgebung mit partitionierten Datenbanken einzurichten, gehen Sie wie folgt vor:

1. Installieren Sie den Datenbankserver, der Eigner der Instanz ist, mithilfe des **DB2-Installationsassistenten**. Detaillierte Anweisungen hierzu finden Sie im entsprechenden Abschnitt zum „Installieren von DB2-Servern" für Ihre Plattform.
   - Stellen Sie im Fenster **Installation und/oder Antwortdateierstellung auswählen** sicher, dass die Option **Installationseinstellungen in einer Antwortdatei speichern** ausgewählt ist. Nach Abschluss der Installation werden zwei Dateien in das Verzeichnis kopiert, das im **DB2-Installationsassistenten** angegeben wurde: `PROD_ESE.rsp` und `PROD_ESE_addpart.rsp`. Die Datei `PROD_ESE.rsp` ist die Antwortdatei der Datenbankserver, die Eigner einer Instanz sind. Die Datei `PROD_ESE_addpart.rsp` ist die Antwortdatei für zugehörige Datenbankserver.
   - Stellen Sie im Fenster **Partitionierungsoptionen für die DB2-Instanz konfigurieren** sicher, dass die Option **Mehrfachpartitionsinstanz** ausgewählt ist, und geben Sie die maximale Anzahl für die logischen Partitionen ein.
2. Machen Sie das DB2-Installationsimage für alle zugehörigen Computer in der Umgebung mit partitionierten Datenbanken verfügbar.
3. Verteilen Sie die Antwortdatei der zugehörigen Datenbankserver (`PROD_ESE_addpart.rsp`).
4. Installieren Sie einen DB2-Datenbankserver auf jedem beteiligten Computer mit dem Befehl **db2setup** unter Linux und UNIX bzw. mit dem Befehl **setup** unter Windows:

   Linux und UNIX

   Wechseln Sie in das Verzeichnis, in dem der Produktcode der DB2-Datenbank verfügbar ist, und führen Sie Folgendes aus:

   ```
   ./db2setup -r /verzeichnis_der_antwortdatei/name_der_antwortdatei
   ```

   Windows

   ```
   setup -u x:\verzeichnis_der_antwortdatei\name_der_antwortdatei
   ```

   Der Befehl, der die Antwortdatei `PROD_ESE_addpart.rsp` verwendet, würde dann wie folgt lauten:

Linux und UNIX

Wechseln Sie in das Verzeichnis, in dem der Produktcode der DB2-Datenbank verfügbar ist, und führen Sie Folgendes aus:

```
./db2setup -r /db2home/PROD_ESE_addpart.rsp
```

Hierbei steht `/db2home` für das Verzeichnis, in das Sie die Antwortdatei kopiert haben.

Windows

```
setup -u c:\resp_files\PROD_ESE_addpart.rsp
```

Hierbei steht `c:\resp_files\` für das Verzeichnis, in das Sie die Antwortdatei kopiert haben.

5. (Nur Linux und UNIX) Konfigurieren Sie die Datei `db2nodes.cfg`. Bei der DB2-Installation wird nur die maximale Anzahl logischer Partitionen reserviert, die Sie für den aktuellen Computer verwenden möchten. Die Datei `db2nodes.cfg` wird nicht konfiguriert. Wenn Sie die Datei `db2nodes.cfg` nicht konfigurieren, bleibt die Instanz weiterhin eine Instanz mit einer einzigen Partition.
6. Aktualisieren Sie die Datei `services` auf den beteiligten Servern, um den entsprechenden FCM-Port für die DB2-Instanz zu definieren. Die Servicedatei befindet sich an der folgenden Position:
   - `/etc/services` unter Linux und UNIX
   - `%SystemRoot%\system32\drivers\etc\services` unter Windows
7. Bei Umgebungen mit partitionierten Datenbanken unter Windows 2000 oder neueren Windows-Betriebssystemen müssen Sie die Sicherheitsfunktion von DB2 Remote Command Service starten, um Ihre Daten und Ressourcen zu schützen.

   Für eine umfassende Sicherheit starten Sie entweder den Computer (falls der Service im Kontext des Kontos 'Lokales System' ausgeführt wird) oder einen Benutzer (falls der Service im Anmeldekontext eines Benutzers ausgeführt wird) für Delegierungszwecke.

   Gehen Sie wie folgt vor, um die Sicherheitsfunktion von DB2 Remote Command Service zu starten:

   a. Öffnen Sie das Fenster mit den Active Directory-Benutzern und -Computern im Domänencontroller: Klicken Sie **Start** an, wählen Sie **Einstellungen** > **Systemsteuerung** > **Verwaltung** und anschließend die Active Directory-Benutzer und -Computer aus.
   b. Klicken Sie im rechten Teilfenster mit der rechten Maustaste den Computer bzw. Benutzer an, der gestartet werden soll, und wählen Sie **Eigenschaften** aus.
   c. Klicken Sie die Registerkarte **Allgemein** an, und wählen Sie das Kontrollkästchen **Computer für Delegierungszwecke vertrauen** aus. Klicken Sie zur Benutzereinstellung die Registerkarte **Konto** an und wählen Sie das Kontrollkästchen **Konto wird für Delegierungszwecke vertraut** in der Gruppe mit den Optionen für Konten aus. Stellen Sie sicher, dass das Feld **Konto kann nicht delegiert werden** nicht ausgewählt ist.
   d. Klicken Sie **OK** an, um den Computer bzw. Benutzer für Delegierungszwecke zu starten.

   Wiederholen Sie diese Schritte für jeden Computer bzw. Benutzer, der gestartet werden soll. Sie müssen den Computer erneut starten, damit die Änderungen der Sicherheitseinstellungen wirksam werden.

Installieren von Datenbankpartitionsservern auf zugehörigen Computern mithilfe einer Antwortdatei (Windows)

In dieser Task wird die Antwortdatei verwendet, die Sie mithilfe des DB2-Installationsassistenten erstellt haben, um Datenbankpartitionsserver auf zugehörigen Computern zu installieren.

Vorbereitende Schritte

- Sie haben eine DB2-Kopie mithilfe des DB2-Installationsassistenten auf dem primären Computer installiert.
- Sie haben eine Antwortdatei für die Installation auf zugehörigen Computern erstellt und auf den zugehörigen Computer kopiert.
- Sie benötigen Administratorberechtigung für die zugehörigen Computer.

Vorgehensweise

Um zusätzliche Datenbankpartitionsserver mithilfe einer Antwortdatei zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich an dem Computer, der der Umgebung mit partitionierten Datenbanken angehören soll, mit dem für die Installation von DB2 definierten Benutzerkonto des lokalen Administrators an.
2. Wechseln Sie in das Verzeichnis, in dem sich der Inhalt der DB2-Datenbankprodukt-DVD befindet. Beispiel:

   ```
   cd c:\db2dvd
   ```

   Dabei ist `db2dvd` der Name des Verzeichnisses mit dem Inhalt der DB2-Datenbankprodukt-DVD.
3. Geben Sie an einer Eingabeaufforderung den Befehl **setup** wie folgt ein:

   ```
   setup -u verzeichnis_der_antwortdatei\name_der_antwortdatei
   ```

   Im folgenden Beispiel befindet sich die Antwortdatei `Addpart.file` im Verzeichnis `c:\antwortdatei`. Der entsprechende Befehl lautet wie folgt:

   ```
   setup -u c:\reponsefile\Addpart.file
   ```
4. Überprüfen Sie nach Abschluss der Installation die Nachrichten in der Protokolldatei. Die Protokolldatei befindet sich im Verzeichnis `Eigene Dateien\DB2LOG\`. Am Ende der Protokolldatei müssten Ausgabedaten ähnlich den folgenden stehen:

   ```
   === Logging stopped: 5/9/2007  10:41:32 ===
   MSI (c) (C0:A8) [10:41:32:984]: Product: DB2
   Enterprise Server Edition - DB2COPY1 -- Installation
   operation completed successfully.
   ```
5. Wenn Sie den Datenbankpartitionsserver, der Eigner der Instanz ist, auf der primären Maschine installieren, reserviert das DB2-Datenbankprodukt einen Portbereich entsprechend der angegebenen Anzahl logischer Datenbankpartitionsserver, die der Umgebung mit partitionierten Datenbanken angehören. Der Standardbereich besteht aus vier Ports. Für jeden Server, der der Umgebung mit partitionierten Datenbanken angehört, müssen Sie die Datei `/etc/services` für die FCM-Ports manuell konfigurieren. Der Bereich der FCM-Ports ist davon abhängig, wie viele logische Partitionen auf dem zugehörigen Computer verwendet werden sollen. Mindestens die beiden Einträge `DB2_`*`instance`* und `DB2_`*`instance`*`_END` sind erforderlich. Weitere Anforderungen für die auf den zugehörigen Computern angegebenen FCM-Ports:

- Die Anfangsportnummer muss mit der Anfangsportnummer des primären Computers übereinstimmen.
- Weitere Ports müssen fortlaufend nummeriert werden.
- Die angegebenen Portnummern müssen frei sein.

Ergebnisse

Sie müssen sich an jedem einzelnen zugehörigen Computer anmelden und dort diese Schritte wiederholen.

Nächste Schritte

Wenn Sie möchten, dass Ihr DB2-Datenbankprodukt auf die DB2-Dokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netz zugreifen kann, müssen Sie das *DB2 Information Center* installieren. Das *DB2 Information Center* enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen DB2-Produkte.

Installieren von Datenbankpartitionsservern auf zugehörigen Computern mithilfe einer Antwortdatei (Linux und UNIX)

In dieser Task wird die Antwortdatei verwendet, die Sie mithilfe des DB2-Installationsassistenten erstellt haben, um Datenbankpartitionsserver auf zugehörigen Computern zu installieren.

Vorbereitende Schritte

- Sie haben ein DB2-Datenbankprodukt mithilfe des DB2-Installationsassistenten auf dem primären Computer installiert und eine Antwortdatei für die Installation auf den zugehörigen Computern erstellt.
- Sie benötigen Rootberechtigung für die zugehörigen Computer.

Vorgehensweise

Um zusätzliche Datenbankpartitionsserver mithilfe einer Antwortdatei zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich als 'Root' an einem Computer an, der der Umgebung mit partitionierten Datenbanken angehören soll.
2. Wechseln Sie in das Verzeichnis, in das Sie den Inhalt der DB2-Datenbankprodukt-DVD kopiert haben. Beispiel:

   ```
   cd /db2home/db2dvd
   ```
3. Geben Sie den Befehl **db2setup** wie folgt ein:

   ```
   ./db2setup -r /verzeichnis_der_antwortdatei/name_der_antwortdatei
   ```

   Im vorliegenden Beispiel wurde die Antwortdatei `'AddPartitionResponse.file'` im Verzeichnis `'/db2home'` gespeichert. In diesem Fall lautet der Befehl wie folgt:

   ```
   ./db2setup -r /db2home/AddPartitionResponse.file
   ```
4. Überprüfen Sie nach Abschluss der Installation die Nachrichten in der Protokolldatei.

Ergebnisse

Sie müssen sich an jedem einzelnen zugehörigen Computer anmelden und dort jeweils die Installation mithilfe der Antwortdatei vornehmen.

Nächste Schritte

Wenn Sie möchten, dass Ihr DB2-Datenbankprodukt auf die DB2-Datenbankdokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netz zugreifen kann, müssen Sie das *DB2 Information Center* installieren. Das *DB2 Information Center* enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen DB2-Datenbankprodukte.

Kapitel 8. Installationsnachbereitung

Prüfen der Installation

Prüfen der Installation einer Umgebung mit partitionierten Datenbanken (Windows)

Um zu prüfen, ob Ihr DB2-Datenbankserver erfolgreich installiert wurde, erstellen Sie eine Beispieldatenbank und führen SQL-Befehle aus, um Beispieldaten abzurufen und um zu prüfen, ob die Daten an alle zugehörigen Datenbankpartitionsserver der Installation verteilt wurden.

Vorbereitende Schritte

Sie haben alle Installationsschritte vollständig ausgeführt.

Vorgehensweise

Um eine Beispieldatenbank (SAMPLE) zu erstellen, gehen Sie wie folgt vor:

1. Melden Sie sich am Primärcomputer (ServerA) als Benutzer mit der Berechtigung SYSADM an.
2. Geben Sie den Befehl **db2sampl** ein, um die Beispieldatenbank (SAMPLE) zu erstellen.

   Die Verarbeitung dieses Befehls kann einige Minuten in Anspruch nehmen. Wenn die Eingabeaufforderung des Befehls wieder angezeigt wird, ist die Verarbeitung abgeschlossen.

   Die Datenbank SAMPLE wird beim Erstellen automatisch mit dem Aliasnamen SAMPLE katalogisiert.
3. Starten Sie den Datenbankmanager durch Eingabe des Befehls **db2start**.
4. Geben Sie in einem DB2-Befehlsfenster die nachstehenden DB2-Befehle ein, um eine Verbindung zur Datenbank SAMPLE herzustellen und eine Liste aller Mitarbeiter ('staff') in Abteilung ('dept') 20 abzurufen:

```
   db2 connect to sample
db2 "select * from staff where dept = 20"
```

5. Um zu prüfen, ob diese Daten an alle Datenbankpartitionsserver verteilt wurden, geben Sie die folgenden Befehle über ein DB2-Befehlsfenster ein:

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   In der Ausgabe dieses Befehls werden die von der Tabelle `employee` verwendeten Datenbankpartitionen aufgelistet. Die spezifische Ausgabe hängt von der Anzahl der Datenbankpartitionen in der Datenbank ab sowie von der Anzahl der Datenbankpartitionen in der Datenbankpartitionsgruppe, die von dem Tabellenbereich verwendet wird, in dem die Tabelle `employee` erstellt wurde.

Nächste Schritte

Nachdem Sie die Installation geprüft haben, können Sie die Datenbank `SAMPLE` löschen, um Plattenspeicherplatz freizugeben. Es ist jedoch sinnvoll, die Beispieldatenbank beizubehalten, falls Sie beabsichtigen, Beispielanwendungen zu verwenden.

Geben Sie den Befehl **db2 drop database sample** ein, um die Datenbank `SAMPLE` zu löschen.

Prüfen der Installation eines partitionierten Datenbankservers (Linux und UNIX)

Mit dem Tool **db2val** können Sie die Kernfunktionen einer DB2-Kopie überprüfen; dabei werden die Installationsdateien, Instanzen, die Datenbankerstellung, Verbindungen zur jeweiligen Datenbank sowie der Zustand von Umgebungen mit partitionierten Datenbanken überprüft.

Weitere Informationen finden Sie in „Prüfen der DB2-Kopie". Der Status einer Umgebung mit partitionierten Datenbanken wird nur geprüft, wenn mindestens zwei Knoten vorhanden sind. Um zu prüfen, ob Ihre Installation des DB2-Datenbankservers erfolgreich war, erstellen Sie darüber hinaus eine Beispieldatenbank und führen Sie SQL-Befehle aus, um Beispieldaten abzurufen und um zu prüfen, ob die Daten an alle zugehörigen Datenbankpartitionsserver der Installation verteilt wurden.

Vorbereitende Schritte

Vergewissern Sie sich vor dem Ausführen dieser Schritte, dass alle Installationsschritte vollständig ausgeführt wurden.

Vorgehensweise

Um eine Beispieldatenbank (`SAMPLE`) zu erstellen, gehen Sie wie folgt vor:

1. Melden Sie sich am Primärcomputer (ServerA) als Instanzeigner an. Im vorliegenden Beispiel ist der Benutzer 'db2inst1' der Instanzeigner.
2. Geben Sie den Befehl **db2sampl** ein, um die Beispieldatenbank (`SAMPLE`) zu erstellen. Standardmäßig wird die Beispieldatenbank im Ausgangsverzeichnis des Instanzeigners erstellt. Im vorliegenden Beispiel ist `/db2home/db2inst1/` das Ausgangsverzeichnis des Instanzeigners. Dieses Ausgangsverzeichnis ist der Standarddatenbankpfad.

   Die Verarbeitung dieses Befehls kann einige Minuten in Anspruch nehmen. Es gibt keine Abschlussnachricht. Wenn die Eingabeaufforderung wieder angezeigt wird, ist die Verarbeitung abgeschlossen.

Die Datenbank SAMPLE wird beim Erstellen automatisch mit dem Aliasnamen SAMPLE katalogisiert.

3. Starten Sie den Datenbankmanager durch Eingabe des Befehls **db2start**.
4. Geben Sie in einem DB2-Befehlsfenster die nachstehenden DB2-Befehle ein, um eine Verbindung zur Datenbank SAMPLE herzustellen und eine Liste aller Mitarbeiter ('staff') in Abteilung ('dept') 20 abzurufen:

   ```
   db2 connect to sample
   db2 "select * from staff where dept = 20"
   ```

5. Um zu prüfen, ob diese Daten an alle Datenbankpartitionsserver verteilt wurden, geben Sie die folgenden Befehle über ein DB2-Befehlsfenster ein:

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   In der Ausgabe dieses Befehls werden die von der Tabelle employee verwendeten Datenbankpartitionen aufgelistet. Die resultierende Ausgabe hängt von den folgenden Faktoren ab:

   - Anzahl der Partitionen in der Datenbank
   - Anzahl der Datenbankpartitionen in der Datenbankpartitionsgruppe, die vom Tabellenbereich verwendet wird, in dem die Tabelle employee erstellt wurde

Nächste Schritte

Nachdem Sie die Installation geprüft haben, können Sie die Datenbank SAMPLE löschen, um Plattenspeicherplatz freizugeben. Geben Sie den Befehl **db2 drop database sample** ein, um die Datenbank SAMPLE zu löschen.

Teil 3. Implementierung und Pflege

Kapitel 9. Vorbereitung zum Erstellen einer Datenbank

Einrichten von Umgebungen mit partitionierten Datenbanken

Die Entscheidung, eine Mehrpartitionsdatenbank zu erstellen, muss vor der Erstellung der betreffenden Datenbank getroffen werden. Im Rahmen Ihrer Entscheidungen beim Datenbankentwurf müssen Sie festlegen, ob die Leistungsverbesserungen genutzt werden sollen, die eine Datenbankpartitionierung zu bieten hat.

Informationen zu diesem Vorgang

Auch in einer Umgebung mit partitionierten Datenbanken verwenden Sie den Befehl **CREATE DATABASE** oder die Funktion 'sqlecrea()' zum Erstellen einer Datenbank. Unabhängig von der verwendeten Methode kann die Anforderung über eine beliebige der Partitionen erfolgen, die in der Datei `db2nodes.cfg` aufgelistet sind. Die Datei `db2nodes.cfg` ist die Konfigurationsdatei des Datenbankpartitionsservers.

Außer in der Windows-Betriebssystemumgebung kann jeder Editor zum Anzeigen und Ändern des Inhalts der Konfigurationsdatei des Datenbankpartitionsservers (`db2nodes.cfg`) verwendet werden. In der Windows-Betriebssystemumgebung müssen Sie die Befehle **db2ncrt** und **db2nchg** zum Erstellen und Ändern der Konfigurationsdatei des Datenbankpartitionsservers verwenden.

Vor der Erstellung einer Mehrpartitionsdatenbank müssen Sie die Datenbankpartition auswählen, die als Katalogknoten für die Datenbank fungieren soll. Anschließend können Sie die Datenbank direkt von dieser Datenbankpartition oder von einem fernen Client aus erstellen, der mit dieser Datenbankpartition verbunden ist. Die Datenbankpartition, zu der Sie die Verbindung (mit ATTACH) herstellen, um den Befehl **CREATE DATABASE** auszuführen, wird zur *Katalogpartition* für diese spezielle Datenbank.

Die Katalogpartition ist die Datenbankpartition, in der alle Systemkatalogtabellen gespeichert werden. Jeglicher Zugriff auf die Systemtabellen muss über diese Datenbankpartition erfolgen. Alle Objekte föderierter Datenbanken (z. B. Wrapper, Server und Kurznamen) werden in den Systemkatalogtabellen in dieser Datenbankpartition gespeichert.

Wenn möglich, sollten Sie jede Datenbank in einer getrennten Instanz erstellen. Falls dies nicht möglich ist (d. h., Sie müssen mehr als eine Datenbank pro Instanz erstellen), sollten Sie die Katalogpartitionen auf die verfügbaren Datenbankpartitionen verteilen. Dadurch verringern sich Konkurrenzsituationen beim Abrufen von Katalogdaten in einer einzelnen Datenbankpartition.

Anmerkung: Sie sollten regelmäßig ein Backup der Katalogpartition durchführen und nach Möglichkeit vermeiden, Benutzerdaten in ihr zu speichern, da diese Daten die für das Backup benötigte Zeit verlängern.

Wenn Sie eine Datenbank erstellen, wird sie automatisch in allen Datenbankpartitionen erstellt, die in der Datei `db2nodes.cfg` definiert sind.

Wenn die erste Datenbank im System erstellt wird, wird ein Systemdatenbankverzeichnis gebildet. An dieses werden Informationen zu anderen Datenbanken, die Sie erstellen, angehängt. Wenn Sie mit UNIX arbeiten, heißt das Systemdatenbank-

verzeichnis sqldbdir und befindet sich im Verzeichnis sqllib unter Ihrem Ausgangsverzeichnis bzw. unter dem Verzeichnis, in dem die DB2-Datenbank installiert wurde. Dieses Verzeichnis muss sich unter UNIX in einem gemeinsam genutzten Dateisystem (z. B. NFS auf UNIX-Plattformen) befinden, weil es nur ein Systemdatenbankverzeichnis für alle Datenbankpartitionen gibt, die zu einer Umgebung mit partitionierten Datenbanken gehören. Unter Windows befindet sich das Systemdatenbankverzeichnis im Instanzverzeichnis.

Ebenfalls im Verzeichnis sqldbdir befindet sich die Systemintentionsdatei. Sie hat den Namen sqldbins und stellt sicher, dass die Datenbankpartitionen synchronisiert bleiben. Diese Datei muss ebenfalls in einem gemeinsam genutzten Dateisystem gespeichert sein, da es innerhalb aller Datenbankpartitionen nur ein Verzeichnis gibt. Die Datei wird von allen Datenbankpartitionen, die die Datenbank bilden, gemeinsam genutzt.

Zur Nutzung der Datenbankpartitionierung müssen Konfigurationsparameter geändert werden. Mithilfe der Befehle **GET DATABASE CONFIGURATION** und **GET DATABASE MANAGER CONFIGURATION** können Sie die Werte für einzelne Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers ermitteln. Zur Änderung einzelner Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers werden die Befehle **UPDATE DATABASE CONFIGURATION** bzw. **UPDATE DATABASE MANAGER CONFIGURATION** verwendet.

Zu den Konfigurationsparametern des Datenbankmanagers, die sich auf eine Umgebung mit partitionierten Datenbanken auswirken, gehören **conn_elapse**, **fcm_num_buffers**, **fcm_num_channels**, **max_connretries**, **max_coordagents**, **max_time_diff**, **num_poolagents** und **start_stop_time**.

Erstellen von Knotenkonfigurationsdateien

Wenn Ihre Datenbank in einer partitionierten Datenbankumgebung arbeiten soll, müssen Sie eine Knotenkonfigurationsdatei mit dem Namen db2nodes.cfg erstellen.

Informationen zu diesem Vorgang

Um eine Datenbankpartitionierung zu ermöglichen, muss sich die Datei db2nodes.cfg im Unterverzeichnis sqllib des Ausgangsverzeichnisses (Home) für die Instanz befinden, bevor Sie den Datenbankmanager starten. Diese Datei enthält Konfigurationsdaten für alle Datenbankpartitionen einer Instanz und wird von allen Datenbankpartitionen für diese Instanz gemeinsam genutzt.

Hinweise für Windows

Wenn Sie DB2 Enterprise Server Edition unter Windows verwenden, wird die Knotenkonfigurationsdatei beim Erstellen der Instanz erstellt. Sie sollten nicht versuchen, eine Knotenkonfigurationsdatei manuell zu erstellen oder manuell zu ändern. Mit Hilfe des Befehls **db2ncrt** können Sie einer Instanz einen Datenbankpartitionsserver hinzufügen. Mit Hilfe des Befehls **db2ndrop** können Sie einen Datenbankpartitionsserver aus einer Instanz löschen. Mit Hilfe des Befehls **db2nchg** können Sie die Konfiguration der Datenbankpartitionsserver ändern. Dies umfasst das Versetzen des Datenbankpartitionsservers von einem Computer auf einen anderen, das Ändern des TCP/IP-Hostnamens oder das Auswählen eines anderen logischen Portnamens oder Netznamens.

Anmerkung: Erstellen Sie keine anderen als die vom Datenbankmanager erstellten Dateien oder Verzeichnisse unter dem Unterverzeichnis `sqllib`, um Datenverluste zu vermeiden, wenn eine Instanz gelöscht wird. Es gibt jedoch zwei Ausnahmen. Wenn Ihr System gespeicherte Prozeduren unterstützt, stellen Sie die gespeicherten Prozeduranwendungen in das Unterverzeichnis 'function' im Unterverzeichnis `sqllib`. Die andere Ausnahme betrifft eventuell erstellte benutzerdefinierte Funktionen (UDFs). Benutzerdefinierte Funktionen können im selben Verzeichnis gespeichert werden.

Die Datei enthält eine Zeile für jede Datenbankpartition, die zu einer Instanz gehört. Jede Zeile hat folgendes Format:

dbpartitionsnummer hostname [*logischer-port* [*netzname*]]

Die Token einer Zeile sind durch Leerzeichen voneinander getrennt. Die Variablen sind:

dbpartitionsnummer
: Die Datenbankpartitionsnummer (mögliche Werte: `0` - `999`) definiert eine Datenbankpartition eindeutig. Datenbankpartitionsnummern müssen in aufsteigender Reihenfolge angegeben sein. Es dürfen Sprünge in der Folge der Nummern auftreten.

  Wenn eine Datenbankpartitionsnummer einmal zugeordnet ist, kann sie nicht mehr geändert werden. (Ansonsten könnten die Informationen in der Verteilungszuordnung, die bestimmt, wie Daten verteilt werden, inkonsistent werden.)

  Wenn Sie eine Datenbankpartition löschen, kann ihre Datenbankpartitionsnummer für jede neue Datenbankpartition, die Sie hinzufügen, wieder verwendet werden.

  Die Datenbankpartitionsnummer wird zur Generierung eines Datenbankpartitionsnamens im Datenbankverzeichnis verwendet. Er hat folgendes Format:

  `NODE` *nnnn*

  Die Ziffernfolge *nnnn* ist die Datenbankpartitionsnummer, die links mit Nullen aufgefüllt wird. Diese Datenbankpartitionsnummer wird außerdem durch die Befehle **`CREATE DATABASE`** und **`DROP DATABASE`** verwendet.

hostname
: Der Hostname der IP-Adresse für die partitionsübergreifende Kommunikation. Verwenden Sie den vollständig qualifizierten Namen für den Hostnamen. In der Datei `/etc/hosts` sollte ebenfalls der vollständig qualifizierte Name verwendet werden. Wenn der vollständig qualifizierte Name in der Datei `db2nodes.cfg` und in der Datei `/etc/hosts` nicht verwendet wird, empfangen Sie eventuell die Fehlernachricht SQL30082N RC=3.

  (Es gibt eine Ausnahme, wenn der Netzname angegeben wird. In diesem Fall wird der Netzname für den Großteil der Kommunikation verwendet, während der Hostname nur für die Befehle **`db2start`**, **`db2stop`** und **`db2_all`** verwendet wird.)

logischer-port
: Dieser Parameter ist optional und gibt die logische Portnummer für die Datenbankpartition an. Diese Nummer wird mit dem Instanznamen des Datenbankmanagers verwendet, um einen TCP/IP-Servicenamenseintrag in der Datei `etc/services` anzugeben.

Die Kombination aus IP-Adresse und logischem Port wird als allgemein bekannte Adresse verwendet und muss für alle Anwendungen eindeutig sein, um die Verbindungen zur Kommunikation zwischen Datenbankpartitionen zu unterstützen.

Für jeden Hostnamen muss ein *logischer-port* entweder `0` (null) oder leer sein (was standardmäßig dem Wert `0` entspricht). Die Datenbankpartition, der dieser *logische-port* zugeordnet ist, ist der Standardknoten auf dem Host, zum dem Clients die Verbindung herstellen. Dieses Verhalten kann mithilfe der Umgebungsvariablen **DB2NODE** im Script **db2profile** oder mit der API sqlesetc() überschrieben werden.

netzname
Dieser Parameter ist optional und wird zur Unterstützung eines Hosts verwendet, der über mehr als eine aktive TCP/IP-Schnittstelle verfügt, von denen jede ihren eigenen Hostnamen hat.

Das folgende Beispiel zeigt eine mögliche Knotenkonfigurationsdatei für ein System, auf dem SP2EN1 mehrere TCP/IP-Schnittstellen und zwei logische Datenbankpartitionen hat und SP2SW1 als DB2-Datenbankschnittstelle verwendet. Es zeigt außerdem die Datenbankpartitionsnummern, beginnend bei 1 (und nicht bei 0), sowie einen Sprung in der Folge der *dbpartitionsnummern*:

Tabelle 12. Eine Tabelle mit einem Beispiel für Datenbankpartitionsnummern

dbpartitions-nummer	*hostname*	*logischer-port*	*netzname*
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

Die Datei `db2nodes.cfg` kann mit einem beliebigen Editor aktualisiert werden. (Ausnahme: Unter Windows sollte kein Editor verwendet werden.) Sie müssen jedoch sorgfältig auf den Schutz der Integrität der Daten in der Datei achten, da die Datenbankpartitionierung erfordert, dass die Knotenkonfigurationsdatei gesperrt wird, wenn der Befehl **START DBM** ausgeführt wird, und entsperrt wird, wenn der Befehl **STOP DBM** den Datenbankmanager beendet hat. Der Befehl **START DBM** kann die Datei bei Bedarf aktualisieren, während sie gesperrt ist. Sie können beispielsweise den Befehl **START DBM** mit der Option **RESTART** oder der Option **ADD DBPARTITIONNUM** ausführen.

Anmerkung: Wenn der Befehl **STOP DBM** nicht erfolgreich ausgeführt und die Knotenkonfigurationsdatei nicht entsperrt wird, führen Sie den Befehl **STOP DBM FORCE** aus, um sie zu entsperren.

Format der DB2-Knotenkonfigurationsdatei

Mithilfe der Datei `db2nodes.cfg` werden die Datenbankpartitionsserver definiert, die einer DB2-Instanz angehören. Außerdem wird über die Datei `'db2nodes.cfg'` die IP-Adresse bzw. der Hostname einer Hochgeschwindigkeitsverbindung angegeben, falls Sie für die Kommunikation zwischen den Datenbankpartitionsservern eine Hochgeschwindigkeitsverbindung verwenden wollen.

Die Datei `db2nodes.cfg` unter Linux- und UNIX-Betriebssystemen hat das folgende Format:

dbpartitionsnum hostname logischer_port netzname ressourcengruppenname

dbpartitionsnum, *hostname*, *logischer_port*, *netzname* und *ressourcengruppenname* sind im folgenden Abschnitt definiert.

Die Datei `db2nodes.cfg` unter Windows-Betriebssystemen hat das folgende Format:
dbpartitionsnum hostname computername logischer_port netzname ressourcengruppenname

Unter Windows-Betriebssystemen werden diese Einträge mit dem Befehl **db2ncrt** oder START DBM ADD DBPARTITIONNUM zur Datei `'db2nodes.cfg'` hinzugefügt. Die Einträge können auch mit dem Befehl **db2nchg** geändert werden. Sie sollten diese Zeilen weder direkt hinzufügen noch diese Datei bearbeiten.

dbpartitionsnum
: Eine eindeutige Nummer zwischen `0` und `999`, die einen Datenbankpartitionsserver in einem partitionierten Datenbanksystem identifiziert.

  Wenn Sie das partitionierte Datenbanksystem skalieren möchten, fügen Sie der Datei `db2nodes.cfg` für jeden Datenbankpartitionsserver jeweils einen Eintrag hinzu. Die Werte für *dbpartitionsnum*, die Sie für weitere Datenbankpartitionsserver auswählen, müssen aufsteigend, aber nicht direkt aufeinanderfolgend sein. Sie können zwischen den Werten von *dbpartitionsnum* beispielsweise eine Lücke lassen, wenn Sie später logische Partitionsserver hinzufügen und für die Knoten eine logische Gruppierung in dieser Datei beibehalten möchten.

  Dieser Eintrag ist erforderlich.

hostname
: Der TCP/IP-Hostname des Datenbankpartitionsservers zur Verwendung durch FCM (Fast Communications Manager). Dieser Eintrag ist erforderlich. Es wird *dringend* empfohlen, einen kanonischen Hostnamen zu verwenden.

  Wenn in der Datei `db2nodes.cfg` Hostnamen anstelle von IP-Adressen enthalten sind, versucht der Datenbankmanager, die Hostnamen dynamisch aufzulösen. Die Auflösung kann entweder lokal oder durch die Suche auf registrierten Domänennamensservern (Domain Name Servers, DNS) erfolgen - abhängig von den Betriebssystemeinstellungen auf der Maschine.

  Ab DB2 Version 9.1 wird sowohl das Protokoll TCP/IPv4 als auch das Protokoll TCP/IPv6 unterstützt. Seither hat sich das Verfahren zur Auflösung der Hostnamen geändert.

  Während in den Releases vor Version 9.1 die in Datei `db2nodes.cfg` definierte Zeichenfolge aufgelöst wurde, wird bei dem Verfahren ab Version 9.1 versucht, die vollständig qualifizierten Domänennamen (Fully Qualified Domain Names, FQDN) aufzulösen, wenn in Datei `db2nodes.cfg` Kurznamen definiert sind. Werden anstelle der vollständig qualifizierten Hostnamen die in der Konfigurationsdatei definierten Kurznamen verwendet, kann dies zu unnötigen Verzögerungen bei Prozessen führen, bei denen Hostnamen aufgelöst werden.

  Um Verzögerungen bei der Verarbeitung von DB2-Befehlen zu vermeiden, die die Auflösung von Hostnamen voraussetzen, verwenden Sie eines der folgenden Verfahren, um das Problem zu umgehen:

  1. Wenn in der Datei `db2nodes.cfg` und in der Datei 'hosts' des Betriebssystems Kurznamen angegeben sind, geben Sie den Kurznamen und den vollständig qualifizierten Domänennamen für den Hostnamen in der Datei 'hosts' des Betriebssystems an.

2. Wenn Sie nur IPv4-Adressen verwenden wollen, weil Sie wissen, dass der DB2-Server an einem IPv4-Port empfangsbereit ist, geben Sie folgenden Befehl ein:

```
db2 catalog tcpip4
    node db2tcp2 remote 192.0.32.67
    server db2inst1 with "Look up IPv4 address from 192.0.32.67"
```

3. Wenn Sie nur IPv6-Adressen verwenden wollen, weil Sie wissen, dass der DB2-Server an einem IPv6-Port empfangsbereit ist, geben Sie folgenden Befehl ein:

```
db2 catalog tcpip6
    node db2tcp3 1080:0:0:0:8:800:200C:417A
    server 50000
    with "Look up IPv6 address from 1080:0:0:0:8:800:200C:417A"
```

logischer_port
Gibt die logische Portnummer für den Datenbankpartitionsserver an. Dieses Feld wird verwendet, um einen bestimmten Datenbankpartitionsserver auf einer Workstation anzugeben, auf der logische Datenbankpartitionsserver ausgeführt werden.

Zum Zeitpunkt der Installation reserviert DB2 einen Portbereich (z. B. 60000 - 60003) in der Datei `'/etc/services'` für die Kommunikation zwischen den Partitionen. Das Feld für den logischen Port (*logischer_port*) in der Datei `'db2nodes.cfg'` gibt an, welcher Port im Bereich einem bestimmten logischen Partitionsserver zugeordnet werden soll.

Wenn dieses Feld keinen Eintrag enthält, ist die Standardeinstellung `0`. Wenn Sie jedoch einen Eintrag für das Feld *netzname* hinzufügen, müssen Sie eine Nummer für das Feld *logischer_port* angeben.

Wenn Sie logische Datenbankpartitionen verwenden, *muss* der von Ihnen angegebene Wert für *logischer_port* bei `0` beginnen und in aufsteigender Reihenfolge fortgesetzt werden (zum Beispiel `0,1,2`).

Weiterhin gilt: Wenn Sie für einen Datenbankpartitionsserver einen Eintrag für *logischer_port* angeben, müssen Sie für jeden Datenbankpartitionsserver, der in der Datei `db2nodes.cfg` aufgelistet ist, einen *logischen_port* angeben.

Jeder physische Server muss über einen logischen Knoten `0` verfügen.

Dieses Feld ist nur dann optional, wenn Sie *keine* logischen Datenbankpartitionen oder Hochgeschwindigkeitsverbindung verwenden.

netzname
Gibt den Hostnamen oder die IP-Adresse der Hochgeschwindigkeitsverbindung für die FCM-Kommunikation an.

Ist für dieses Feld ein Eintrag vorhanden, erfolgt die gesamte Kommunikation zwischen den Datenbankpartitionsservern (mit Ausnahme der Kommunikation als Ergebnis der Befehle **db2start**, **db2stop** und **db2_all**) über die Hochgeschwindigkeitsverbindung.

Dieser Parameter ist nur dann erforderlich, wenn Sie für die Kommunikation zwischen Datenbankpartitionen eine Hochgeschwindigkeitsverbindung verwenden.

ressourcengruppenname
Der Ressourcengruppenname (*ressourcengruppenname*) definiert die Betriebssystemressource, in der der Knoten gestartet werden soll. Der Resourcengruppenname dient zur Unterstützung der Prozessaffinität, die für MLNs

(Multiple Logical Nodes) verwendet wird. Diese Unterstützung wird durch ein Feld vom Typ 'Zeichenfolge' bereitgestellt, das früher als 'quadname' bezeichnet wurde.

Dieser Parameter wird nur unter AIX-, HP-UX- und Solaris-Betriebssystemen unterstützt.

Unter AIX wird dieses Konzept als 'Ressourcengruppen' und im Solaris-Betriebssystem als 'Projekte' bezeichnet. Weitere Informationen zum Ressourcenmanagement enthält die Dokumentation zum betreffenden Betriebssystem.

Unter HP-UX ist der Parameter für den Ressourcengruppennamen (*ressourcengruppenname*) ein Name der PRM-Gruppe. Weitere Informationen finden Sie in der Dokumentation 'HP-UX Process Resource Manager.User Guide. (B8733-90007)' von HP.

Unter Windows-Betriebssystemen kann die Prozessaffinität für einen logischen Knoten über die Registrierdatenbankvariable **DB2PROCESSORS** definiert werden.

Unter Linux-Betriebssystemen definiert die Spalte für den Ressourcengruppennamen (*ressourcengruppenname*) eine Nummer, die einem NUMA-Knoten (Non-Uniform Memory Access, NUMA) im System entspricht. Das Systemdienstprogramm **numactl** muss zusätzlich zu einem Kernel Version 2.6 mit Unterstützung für NUMA-Richtlinien verfügbar sein.

Der Parameter für den Netznamen (*netzname*) muss angegeben werden, wenn der Parameter für den Ressourcengruppennamen (*ressourcengruppenname*) verwendet wird.

Beispielkonfigurationen

Anhand der folgenden Beispielkonfigurationen können Sie die geeignete Konfiguration für Ihre Umgebung ermitteln.

Ein Computer, vier Datenbankpartitionsserver

Wenn Sie keine Clusterumgebung verwenden und vier Datenbankpartitionsserver auf einer physischen Workstation namens `ServerA` ausführen wollen, müssen Sie die Datei `db2nodes.cfg` wie folgt aktualisieren:

```
0          ServerA        0
1          ServerA        1
2          ServerA        2
3          ServerA        3
```

Zwei Computer, ein Datenbankpartitionsserver pro Computer

Wenn Ihr partitioniertes Datenbanksystem zwei physische Workstations namens `ServerA` und `ServerB` enthalten soll, müssen Sie die Datei `db2nodes.cfg` wie folgt aktualisieren:

```
0          ServerA        0
1          ServerB        0
```

Zwei Computer, drei Datenbankpartitionsserver auf einem Computer

Wenn Ihr partitioniertes Datenbanksystem zwei physische Workstations namens `ServerA` und `ServerB` enthalten soll und wenn auf `ServerA` 3 Datenbankpartitionsserver ausgeführt werden sollen, aktualisieren Sie die Datei `db2nodes.cfg` wie folgt:

```
4          ServerA        0
6          ServerA        1
8          ServerA        2
9          ServerB        0
```

Zwei Computer, drei Datenbankpartitionsserver mit Hochgeschwindigkeitsschaltern Wenn Ihr partitioniertes Datenbanksystem zwei Computer namens `ServerA` und `ServerB` enthalten soll (wobei auf `ServerB` zwei Datenbankpartitionsserver ausgeführt werden), und Sie eine Hochgeschwindigkeitsverbindung namens `switch1` und `switch2` verwenden wollen, aktualisieren Sie die Datei `db2nodes.cfg` wie folgt:

```
0          ServerA        0              switch1
1          ServerB        0              switch2
2          ServerB        1              switch2
```

Beispiele für die Verwendung von 'ressourcengruppenname'

Diese Einschränkungen gelten für folgende Beispiele:

- Dieses Beispiel zeigt, wie *ressourcengruppenname* verwendet wird, wenn die Konfiguration keine Hochgeschwindigkeitsverbindung umfasst.
- Der Netzname (*netzname*) steht in der vierten Spalte und ein Hostname (*hostname*) kann ebenfalls für diese Spalte angegeben werden, wenn kein Schaltername vorhanden ist und *ressourcengruppenname* verwendet werden soll. Der fünfte Parameter ist *ressourcengruppenname*, sofern er definiert ist. Die Ressourcengruppenspezifikation kann nur als fünfte Spalte in der Datei `'db2nodes.cfg'` angezeigt werden. Das heißt, dass Sie eine vierte Spalte eingeben müssen, um eine Ressourcengruppe angeben zu können. Die vierte Spalte ist für einen Hochgeschwindigkeitsschalter vorgesehen.
- Wenn Sie keinen Hochgeschwindigkeitsschalter haben oder diesen nicht verwenden wollen, müssen Sie den Hostnamen (*hostname*) eingeben (derselbe Name wie in der zweiten Spalte). Ein DB2-Datenbankmanagementsystem unterstützt demnach keine Spaltenabstände (und den Austausch derselben) in `'db2nodes.cfg'`-Dateien. Diese Einschränkung gilt bereits für die ersten drei Spalten und erstreckt sich nun auf alle fünf Spalten.

Beispiel für AIX

Das folgende Beispiel zeigt, wie die Ressourcengruppe für AIX-Betriebssysteme eingerichtet wird.

In diesem Beispiel gibt es einen (1) physischen Knoten mit 32 Prozessoren und 8 logischen Datenbankpartitionen (MLNs). Es wird gezeigt, wie Prozessaffinität für jeden dieser MLNs zur Verfügung gestellt wird.

1. Definieren Sie die Ressourcengruppen in `'/etc/rset'` wie folgt:

```
DB2/MLN1:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000,sys/cpu.00001,sys/cpu.00002,sys/cpu.00003

DB2/MLN2:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004,sys/cpu.00005,sys/cpu.00006,sys/cpu.00007

DB2/MLN3:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00008,sys/cpu.00009,sys/cpu.00010,sys/cpu.00011

DB2/MLN4:
```

```
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00012,sys/cpu.00013,sys/cpu.00014,sys/cpu.00015

DB2/MLN5:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00016,sys/cpu.00017,sys/cpu.00018,sys/cpu.00019

DB2/MLN6:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00020,sys/cpu.00021,sys/cpu.00022,sys/cpu.00023

DB2/MLN7:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00024,sys/cpu.00025,sys/cpu.00026,sys/cpu.00027

DB2/MLN8:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00028,sys/cpu.00029,sys/cpu.00030,sys/cpu.00031
```

2. Aktivieren Sie die Speicheraffinität, indem Sie den folgenden Befehl eingeben:

```
vmo -p -o memory_affinity=1
```

3. Erteilen Sie Instanzberechtigungen zur Verwendung von Ressourcengruppen:

```
chuser capabilities=
    CAP_BYPASS_RAC_VMM,CAP_PROPAGATE,CAP_NUMA_ATTACH  db2inst1
```

4. Fügen Sie den Ressourcengruppennamen als fünfte Spalte in `'db2nodes.cfg'` hinzu:

```
1 regatta 0 regatta DB2/MLN1
2 regatta 1 regatta DB2/MLN2
3 regatta 2 regatta DB2/MLN3
4 regatta 3 regatta DB2/MLN4
5 regatta 4 regatta DB2/MLN5
6 regatta 5 regatta DB2/MLN6
7 regatta 6 regatta DB2/MLN7
8 regatta 7 regatta DB2/MLN8
```

Beispiel für HP-UX

Das folgende Beispiel zeigt, wie PRM-Gruppen für gemeinsam genutzte CPU-Kapazitäten auf einer Maschine mit 4 CPUs und 4 MLNs verwendet werden, wobei für jeden MLN 24 % der gemeinsamen CPU-Kapazität eingestellt werden sollen, sodass 4 % für andere Anwendungen übrig bleiben. Der Name der DB2-Instanz lautet 'db2inst1'.

1. Bearbeiten Sie den Abschnitt GROUP von `'/etc/prmconf'` wie folgt:

```
 OTHERS:1:4::
db2prm1:50:24::
 db2prm2:51:24::
  db2prm3:52:24::
 db2prm4:53:24::
```

2. Fügen Sie den Eintrag des Instanzeigners für `'/etc/prmconf'` wie folgt hinzu:

```
db2inst1::::OTHERS,db2prm1,db2prm2,db2prm3,db2prm4
```

3. Initialisieren Sie die Gruppen und aktivieren Sie den CPU-Manager durch Eingabe des folgenden Befehls:

   ```
   prmconfig -i
   prmconfig -e CPU
   ```

4. Fügen Sie die PRM-Gruppennamen als fünfte Spalte in `'db2nodes.cfg'` wie folgt hinzu:

   ```
   1 voyager 0 voyager db2prm1
   2 voyager 1 voyager db2prm2
   3 voyager 2 voyager db2prm3
   4 voyager 3 voyager db2prm4
   ```

Die PRM-Konfiguration (Schritte 1-3) kann mithilfe des interaktiven GUI-Tools **'xprm'** erfolgen.

Beispiel für Linux

Unter Linux-Betriebssystemen definiert die Spalte für den Ressourcengruppennamen (*ressourcengruppenname*) eine Nummer, die einem NUMA-Knoten (Non-Uniform Memory Access, NUMA) im System entspricht. Das Systemdienstprogramm **'numactl'** muss zusätzlich zu einem Kernel Version 2.6 mit Unterstützung für NUMA-Richtlinien verfügbar sein. Weitere Informationen zur NUMA-Unterstützung unter Linux-Betriebssystemen finden Sie auf der Man-Page für **numactl**.

In diesem Beispiel wird erläutert, wie ein NUMA-Computer mit vier Knoten konfiguriert wird, wobei jedem logischen Knoten ein NUMA-Knoten zugeordnet ist.

1. Stellen Sie sicher, dass NUMA-Funktionen auf dem System vorhanden sind.
2. Setzen Sie den folgenden Befehl ab:

   ```
   $ numactl --hardware
   ```

   Die Ausgabe sieht etwa wie folgt aus:

   ```
   available: 4 nodes (0-3)
   node 0 size: 1901 MB
   node 0 free: 1457 MB
   node 1 size: 1910 MB
   node 1 free: 1841 MB
   node 2 size: 1910 MB
   node 2 free: 1851 MB
   node 3 size: 1905 MB
   node 3 free: 1796 MB
   ```

3. In diesem Beispiel sind vier NUMA-Knoten auf dem System vorhanden. Bearbeiten Sie die Datei `'db2nodes.cfg'` wie angegeben, sodass jedem MLN auf dem System ein NUMA-Knoten zugeordnet ist:

   ```
   0 hostname 0 hostname 0
   1 hostname 1 hostname 1
   2 hostname 2 hostname 2
   3 hostname 3 hostname 3
   ```

Beispiel für Solaris

Das folgende Beispiel zeigt, wie das Projekt für Solaris Version 9 eingerichtet wird.

In diesem Beispiel gibt es einen (1) physischen Knoten mit acht (8) Prozessoren: Eine CPU wird für das Standardprojekt verwendet, drei (3) CPUs werden vom Anwendungsserver verwendet und vier (4) CPUs werden für DB2 verwendet. Der Instanzname lautet 'db2inst1'.

1. Erstellen Sie mithilfe eines Editors eine Konfigurationsdatei für den Ressourcenpool. In diesem Beispiel wird die Datei 'pool.db2' genannt. Sie hat folgenden Inhalt:

```
create system hostname
create pset pset_default (uint pset.min = 1)
create pset db0_pset (uint pset.min = 1; uint pset.max = 1)
create pset db1_pset (uint pset.min = 1; uint pset.max = 1)
create pset db2_pset (uint pset.min = 1; uint pset.max = 1)
create pset db3_pset (uint pset.min = 1; uint pset.max = 1)
create pset appsrv_pset (uint pset.min = 3; uint pset.max = 3)
create pool pool_default (string pool.scheduler="TS";
     boolean pool.default = true)
create pool db0_pool (string pool.scheduler="TS")
create pool db1_pool (string pool.scheduler="TS")
create pool db2_pool (string pool.scheduler="TS")
create pool db3_pool (string pool.scheduler="TS")
create pool appsrv_pool (string pool.scheduler="TS")
associate pool pool_default (pset pset_default)
associate pool db0_pool (pset db0_pset)
associate pool db1_pool (pset db1_pset)
associate pool db2_pool (pset db2_pset)
associate pool db3_pool (pset db3_pset)
associate pool appsrv_pool (pset appsrv_pset)
```

2. Bearbeiten Sie die Datei '/etc/project', indem Sie die DB2-Projekte und das Projekt 'appsrv' wie folgt hinzufügen:

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
appsrv:4000:App Serv project:root::project.pool=appsrv_pool
db2proj0:5000:DB2 Node 0 project:db2inst1,root::project.pool=db0_pool
db2proj1:5001:DB2 Node 1 project:db2inst1,root::project.pool=db1_pool
db2proj2:5002:DB2 Node 2 project:db2inst1,root::project.pool=db2_pool
db2proj3:5003:DB2 Node 3 project:db2inst1,root::project.pool=db3_pool
```

3. Erstellen Sie den Ressourcenpool wie folgt: # poolcfg -f pool.db2.
4. Activate the resource pool: # pooladm -c.
5. Fügen Sie den Projektnamen als fünfte Spalte in der Datei 'db2nodes.cfg' wie folgt hinzu:

```
0 hostname 0 hostname db2proj0
1 hostname 1 hostname db2proj1
2 hostname 2 hostname db2proj2
3 hostname 3 hostname db2proj3
```

Angeben der Liste von Systemen in einer Umgebung mit partitionierten Datenbanken

Standardmäßig wird die Liste der Computer der Datei für die Datenbankpartitionskonfiguration db2nodes.cfg entnommen.

Informationen zu diesem Vorgang

Anmerkung: Unter Windows sollten Sie die Datei für die Datenbankpartitionskonfiguration auf *keinen* Fall manuell bearbeiten, um keine Inkonsistenzen zu verursachen. Zum Abrufen der Liste der Computer in der Instanz verwenden Sie den Befehl **db2nlist**.

Vorgehensweise

Gehen Sie wie folgt vor, um die Liste der Computer in db2nodes.cfg zu überschreiben:

- Geben Sie einen Pfadnamen zu der Datei an, die die Liste der Computer enthält, indem Sie die Umgebungsvariable **RAHOSTFILE** exportieren (auf Linux- und UNIX-Betriebssystemen) oder definieren (unter Windows).
- Geben Sie die Liste explizit als Zeichenfolge durch Leerzeichen getrennter Namen an, indem Sie die Umgebungsvariable **RAHOSTLIST** exportieren (auf Linux- und UNIX-Betriebssystemen) oder definieren (unter Windows).

  Anmerkung: Wenn beide dieser Umgebungsvariablen angegeben werden, erhält **RAHOSTLIST** den Vorrang.

Eliminieren doppelter Einträge aus einer Liste von Systemen in einer Umgebung mit partitionierten Datenbanken

Wenn Sie mit mehreren logischen Datenbankpartitionsservern auf einem Computer arbeiten, enthält Ihre Datei db2nodes.cfg mehrere Einträge für diesen Computer.

Informationen zu diesem Vorgang

In diesem Fall muss dem Befehl **rah** mitgeteilt werden, ob der Befehl nur einmal auf jedem Computer oder einmal für jede in der Datei db2nodes.cfg aufgeführte logische Datenbankpartition ausgeführt werden soll. Verwenden Sie den Befehl **rah**, wenn Sie Computer angeben möchten. Verwenden Sie den Befehl **db2_all**, wenn Sie logische Datenbankpartitionen angeben möchten.

Anmerkung: Wenn Sie auf Linux- und UNIX-Betriebssystemen Computer angeben, entfernt **rah** normalerweise doppelte Einträge aus der Computerliste, jedoch mit folgender Ausnahme: Wenn Sie logische Datenbankpartitionen angeben, setzt der Befehl **db2_all** Ihrem Befehl die folgende Zuweisung voran:

```
export DB2NODE=nnn (für Korn-Shell-Syntax)
```

Dabei ist *nnn* die Datenbankpartitionsnummer, die der entsprechenden Zeile in der Datei db2nodes.cfg entnommen ist, sodass der Befehl an den gewünschten Datenbankpartitionsserver weitergeleitet wird.

Bei der Angabe logischer Datenbankpartitionen können Sie die Liste mithilfe der Präfixsequenzen <<-*nnn*< und <<+*nnn*< auf alle logischen Datenbankpartitionen außer einer beschränken bzw. nur eine angeben. Diese Möglichkeit kann nützlich sein, wenn Sie einen Befehl ausführen möchten, um zuerst die Datenbankpartition zu katalogisieren, und nach Abschluss dieses Befehls denselben Befehl in allen anderen Datenbankpartitionsservern eventuell parallel ausführen möchten. Normalerweise ist diese Methode für die Ausführung des Befehls **RESTART DATABASE** erforderlich. Für diesen Fall müssen Sie die Datenbankpartitionsnummer der Katalogpartition kennen.

Wenn Sie den Befehl **RESTART DATABASE** mithilfe des Befehls **rah** ausführen, werden doppelte Einträge aus der Liste der Computer eliminiert. Wenn Sie jedoch das Präfix " angeben, werden doppelte Einträge nicht eliminiert, da angenommen wird, dass die Verwendung des Präfixes " das Senden der Befehle an alle Datenbankpartitionsserver und nicht an alle Computer impliziert.

Aktualisieren der Knotenkonfigurationsdatei (Linux und UNIX)

In einer DB2-Umgebung mit partitionierten Datenbanken beschreibt die vorliegende Task die erforderlichen Schritte zum Aktualisieren der Datei db2nodes.cfg, um Einträge für die zugehörigen Computer hinzuzufügen.

Vorbereitende Schritte

- Das DB2-Datenbankprodukt muss auf allen zugehörigen Computern installiert sein.
- Auf dem Primärcomputer muss eine DB2-Instanz vorhanden sein.
- Sie müssen als Benutzer über die Berechtigung SYSADM verfügen.
- Prüfen Sie die Konfigurationsbeispiele und Informationen zum Dateiformat im Abschnitt über das Format der DB2-Knotenkonfigurationsdatei, wenn eine der folgenden Bedingungen zutrifft:
  - Sie beabsichtigen, einen Hochgeschwindigkeitsschalter für die Kommunikation zwischen Datenbankpartitionsservern zu verwenden
  - Ihre partitionierte Konfiguration soll mehrere logische Partitionen enthalten

Informationen zu diesem Vorgang

Die Knotenkonfigurationsdatei (db2nodes.cfg) im Ausgangsverzeichnis des Instanzeigners enthält Konfigurationsdaten, mit deren Hilfe das DB2-Datenbanksystem ermittelt, welche Server einer Instanz der Umgebung mit partitionierten Datenbanken angehören. Für jede Instanz in einer Umgebung mit partitionierten Datenbanken ist eine Datei db2nodes.cfg vorhanden.

Die Datei db2nodes.cfg muss für jeden Server, der der Instanz angehört, jeweils einen Eintrag enthalten. Bei der Erstellung einer Instanz wird die Datei db2nodes.cfg automatisch erstellt und für den Server, der Instanzeigner ist, wird ein Eintrag hinzugefügt.

Wenn Sie beispielsweise die DB2-Instanz mithilfe des **DB2-Installationsassistenten** auf dem Server ServerA (Instanzeigner) erstellt haben, wird die Datei db2nodes.cfg wie folgt aktualisiert:

```
0     ServerA     0
```

Einschränkungen

Bei den Hostnamen, die in den Schritten im Abschnitt 'Vorgehensweise' verwendet werden, muss es sich um vollständig qualifizierte Hostnamen handeln.

Vorgehensweise

Um die Datei db2nodes.cfg zu aktualisieren, gehen Sie wie folgt vor:

1. Melden Sie sich als Instanzeigner an. In diesen Schritten ist beispielsweise 'db2inst1' der Instanzeigner.
2. Stoppen Sie die DB2-Instanz, indem Sie folgenden Befehl eingeben:
   INSTANZAUSGANGSVERZEICHNIS/sqllib/adm/db2stop

   Dabei ist *INSTANZAUSGANGSVERZEICHNIS* das Ausgangsverzeichnis des Instanzeigners. (Ist die Instanz aktiv, wird die Datei db2nodes.cfg gesperrt. Sie kann erst nach dem Stoppen der Instanz editiert werden.)

   Lautet das Ausgangsverzeichnis der Instanz beispielsweise /db2home/db2inst1, geben Sie folgenden Befehl ein:

```
/db2home/db2inst1/sqllib/adm/db2stop
```

3. Fügen Sie der Datei `.rhosts` für jeder DB2-Instanz einen Eintrag hinzu. Aktualisieren Sie die Datei, indem Sie Folgendes hinzufügen:

   ```
   hostname db2-instanz
   ```

   Dabei ist *hostname* der TCP/IP-Hostname des Datenbankservers und *db2-instanz* der Name der Instanz, die Sie für den Zugriff auf den Datenbankserver verwenden.

4. Fügen Sie der Datei `db2nodes.cfg` jedes zugehörigen Servers einen Eintrag hinzu. Wenn Sie die Datei `db2nodes.cfg` zum ersten Mal anzeigen, sollte sie einen ähnlichen Eintrag wie den folgenden enthalten:

   ```
   0     ServerA     0
   ```

   Dieser Eintrag enthält die Nummer des Datenbankpartitionsservers (Knotennummer), den TCP/IP-Hostnamen des Servers, auf dem sich der Datenbankpartitionsserver befindet, und die logische Portnummer des Datenbankpartitionsservers.

   Wenn Sie beispielsweise auf jedem der Computer eine partitionierte Konfiguration mit vier Computern und einem Datenbankpartitionsserver installieren, sollte die aktualisierte Datei `db2nodes.cfg` etwa wie folgt aussehen:

   ```
   0     ServerA     0
   1     ServerB     0
   2     ServerC     0
   3     ServerD     0
   ```

5. Geben Sie nach abgeschlossener Aktualisierung der Datei `db2nodes.cfg` den Befehl *INSTANZAUSGANGSVERZEICHNIS*`/sqllib/adm/db2start` ein, wobei *INSTANZAUSGANGSVERZEICHNIS* das Ausgangsverzeichnis des Instanzeigners ist. Lautet das Ausgangsverzeichnis der Instanz beispielsweise `/db2home/db2inst1`, geben Sie folgenden Befehl ein:

   ```
   /db2home/db2inst1/sqllib/adm/db2start
   ```

6. Melden Sie sich ab.

Einrichten mehrerer logischer Partitionen

Es gibt verschiedene Fälle, in denen es günstiger ist, wenn auf einem Computer mehrere Datenbankpartitionsserver ausgeführt werden.

Dies bedeutet, dass die Konfiguration eine höhere Anzahl von Datenbankpartitionen als Computer umfassen kann. In diesen Fällen werden auf dem betreffenden Computer *mehrere logische Partitionen* ausgeführt, wenn diese dieselbe Instanz verwenden. Andernfalls werden auf dem Computer nicht mehrere logische Partitionen ausgeführt.

Durch die Unterstützung mehrerer logischer Partitionen können drei verschiedene Konfigurationsarten ausgewählt werden:

- Eine Standardkonfiguration, bei der jeder Computer über nur einen Datenbankpartitionsserver verfügt
- Eine Konfiguration mit mehreren logischen Partitionen, bei der ein Computer über mehrere Datenbankpartitionsserver verfügt
- Eine Konfiguration, bei der mehrere logische Partitionen auf mehreren Computern ausgeführt werden

Konfigurationen mit mehreren logischen Partitionen eignen sich gut, wenn mit dem System Abfragen auf einem Computer mit SMP-Architektur (SMP = Symmetric Multiprocessor) ausgeführt werden. Die Möglichkeit, mehrere logische Partitio-

nen auf einem Computer zu konfigurieren, ist auch im Falle eines Computerausfalls von Vorteil. Wenn ein Computer (und damit auch der bzw. die auf diesem Computer ausgeführte(n) Datenbankpartitionsserver) ausfällt, können Sie den/die Datenbankpartitionsserver mit dem Befehl **START DBM DBPARTITIONNUM** auf einem anderen Computer neu starten. Dies stellt die kontinuierliche Verfügbarkeit der Benutzerdaten sicher.

Ein weiterer Vorteil bei mehreren logischen Partitionen besteht darin, dass diese die SMP-Hardwarekonfigurationen nutzen können. Darüber hinaus lassen sich durch die kleineren Datenbankpartitionen bessere Leistungswerte beim Backup und Restore von Datenbankpartitionen und Tabellenbereichen sowie bei der Indexerstellung erzielen.

Konfigurieren mehrerer logischer Partitionen

Es gibt zwei Methoden zur Konfiguration mehrerer logischer Partitionen.

Informationen zu diesem Vorgang

- Konfigurieren der logischen Partitionen (Datenbankpartitionen) in der Datei `db2nodes.cfg`. Anschließend können Sie alle logischen und fernen Partitionen mit dem Befehl **db2start** bzw. der entsprechenden API starten.

  Anmerkung: Für Windows müssen Sie zum Hinzufügen einer Datenbankpartition den Befehl **db2ncrt** verwenden, wenn auf dem System keine Datenbank vorhanden ist. Wenn bereits eine oder mehrere Datenbanken vorhanden sind, muss der Befehl **db2start addnode** verwendet werden. Unter Windows sollte die Datei `db2nodes.cfg` nie manuell bearbeitet werden.
- Neustarten einer logischen Partition auf einem anderen Prozessor, auf dem bereits andere logische Partitionen ausgeführt werden. Mit dieser Methode können Sie den Hostnamen und die Portnummer, die in der Datei `db2nodes.cfg` für die logische Partition angegeben sind, außer Kraft setzen.

Zur Konfiguration einer logischen Datenbankpartition in der Datei `db2nodes.cfg` müssen Sie einen Eintrag in der Datei vornehmen, um der Datenbankpartition eine logische Portnummer zuzuordnen. Dabei gilt die folgende Syntax:

```
knotennummer hostname logischer-port netzname
```

Für das IBM DB2 pureScale Feature müssen Sie sicherstellen, dass ein Member mit der Knotennummer "0" vorhanden ist.

Anmerkung: Für Windows müssen Sie zum Hinzufügen einer Datenbankpartition den Befehl **db2ncrt** verwenden, wenn auf dem System keine Datenbank vorhanden ist. Wenn bereits eine oder mehrere Datenbanken vorhanden sind, muss der Befehl **db2start addnode** verwendet werden. Unter Windows sollte die Datei `db2nodes.cfg` nie manuell bearbeitet werden.

Das Format der Datei `db2nodes.cfg` unter Windows unterscheidet sich vom Format dieser Datei unter UNIX. Unter Windows gilt folgendes Spaltenformat:

```
knotennummer hostname computername logischer_port netzname
```

Verwenden Sie den vollständig qualifizierten Namen für den Hostnamen. In der Datei `/etc/hosts` sollte ebenfalls der vollständig qualifizierte Name verwendet werden. Wenn der vollständig qualifizierte Name in der Datei `db2nodes.cfg` und in der Datei `/etc/hosts` nicht verwendet wird, empfangen Sie eventuell die Fehlernachricht `SQL30082N RC=3`.

Sie müssen sicherstellen, dass in der Datei `services` des Verzeichnisses `etc` genügend Ports für die FCM-Kommunikation definiert sind.

Aktivieren der partitionsübergreifenden Abfrageparallelität

Partitionsübergreifende Parallelität tritt entsprechend der Anzahl von Datenbankpartitionen und der Verteilung von Daten auf diese Datenbankpartitionen automatisch auf.

Informationen zu diesem Vorgang

Zur Nutzung der Parallelität innerhalb einer Datenbankpartition oder einer nicht partitionierten Datenbank müssen Sie die Konfigurationsparameter ändern. Durch die partitionsinterne Parallelität können Sie zum Beispiel mehrere Prozessoren einer SMP-Maschine (SMP = Symmetric Multiprocessor) nutzen.

Vorgehensweise

- Für die Aktivierung der Parallelität beim Laden von Daten gilt:

  Das Dienstprogramm LOAD nutzt die Parallelität automatisch, und Sie können die folgenden Parameter im Befehl **LOAD** verwenden:

  – **CPU_PARALLELISM**

  – **DISK_PARALLELISM**

  In einer Umgebung mit partitionierten Datenbanken erfolgt die partitionsübergreifende Parallelität zum Laden von Daten automatisch, wenn die Zieltabelle für mehrere Datenbankpartitionen definiert ist. Die partitionsübergreifende Parallelität zum Laden von Daten kann durch die Angabe von **OUTPUT_DBPARTNUMS** überschrieben werden. Das Dienstprogramm LOAD macht außerdem intelligenten Gebrauch von der Parallelität durch die Datenbankpartitionierung in Abhängigkeit von der Größe der Zieldatenbankpartitionen. Der Parameter **MAX_NUM_PART_AGENTS** kann zur Steuerung des maximalen Grads der Parallelität verwendet werden, der vom Dienstprogramm LOAD ausgewählt wird. Die Parallelität durch Datenbankpartitionierung kann durch Angeben von **PARTITIONING_DBPARTNUMS** überschrieben werden, wenn außerdem **ANYORDER** angegeben wird.
- Für die Aktivierung der Parallelität bei der Erstellung eines Index gilt:
  – Die Tabelle muss ausreichend groß sein, um die Parallelität vorteilhaft nutzen zu können.
  – Auf einem SMP-Computer müssen mehrere Prozessoren aktiviert sein.
- Gehen Sie wie folgt vor, um die E/A-Parallelität für das Backup einer Datenbank oder eines Tabellenbereichs zu aktivieren:
  – Verwenden Sie mehr als einen Zieldatenträger.
  – Konfigurieren Sie Tabellenbereiche für parallele E/A, indem Sie mehrere Container definieren, oder verwenden Sie einen einzigen Container mit mehreren Platten, und verwenden Sie die Registrierdatenbankvariable **DB2_PARALLEL_IO** entsprechend. Wenn Sie eine parallele E/A nutzen wollen, müssen Sie bereits vor dem Definieren von Containern die erforderlichen Vorbedingungen berücksichtigen und entsprechend vorgehen. Diese Vorbedingungen können nicht erst bei Erkennung eines Bedarfs erfüllt werden, sondern müssen eingeplant werden, bevor Sie den Punkt erreichen, an dem Sie Ihre Datenbank oder Ihren Tabellenbereich sichern müssen.
  – Geben Sie mit dem Parameter **PARALLELISM** im Befehl **BACKUP** den Grad der Parallelität an.

– Verwenden Sie den Parameter **WITH** anzahl-puffer **BUFFERS** im Befehl **BACKUP**, um sicherzustellen, dass genügend Puffer für den Grad der Parallelität verfügbar sind. Die Anzahl der Puffer sollte der Summe aus der Anzahl Ihrer Zieldatenträger und dem ausgewählten Grad der Parallelität plus einigen zusätzlichen Puffern entsprechen.

  Verwenden Sie ferner einen Backup-Puffer, für dessen Größe Folgendes gilt:

  - So groß wie möglich. 4 MB oder 8 MB (1024 oder 2048 Seiten) ist eine zweckmäßige Faustregel.
  - Mindestens so groß wie das größte Produkt (EXTENTSIZE * Anzahl der Container) der zu sichernden Tabellenbereiche.

- Gehen Sie wie folgt vor, um die E/A-Parallelität für den Restore einer Datenbank oder eines Tabellenbereichs zu aktivieren:
  – Verwenden Sie mehr als einen Quellendatenträger.
  – Konfigurieren Sie Tabellenbereiche für eine parallele E/A. Sie müssen die Entscheidung für die Verwendung dieser Option treffen, bevor Sie Ihre Container definieren. Die entsprechenden Vorbedingungen können nicht erst bei Erkennung eines Bedarfs erfüllt werden, sondern müssen eingeplant werden, bevor Sie den Punkt erreichen, an dem Sie Ihre Datenbank oder Ihren Tabellenbereich wiederherstellen müssen.
  – Geben Sie mit dem Parameter **PARALLELISM** im Befehl **RESTORE** den Grad der Parallelität an.
  – Verwenden Sie den Parameter **WITH** anzahl-puffer **BUFFERS** im Befehl **RESTORE**, um sicherzustellen, dass genügend Puffer für den Grad der Parallelität verfügbar sind. Die Anzahl der Puffer sollte der Summe aus der Anzahl Ihrer Zieldatenträger und dem ausgewählten Grad der Parallelität plus einigen zusätzlichen Puffern entsprechen.

    Verwenden Sie ferner einen Restorepuffer, für dessen Größe Folgendes gilt:

    - So groß wie möglich. 4 MB oder 8 MB (1024 oder 2048 Seiten) ist eine zweckmäßige Faustregel.
    - Mindestens so groß wie das größte Produkt (EXTENTSIZE * Anzahl der Container) der wiederherzustellenden Tabellenbereiche.
    - Dieselbe Größe oder ein gerades Vielfaches der Größe des Backup-Puffers.

Aktivieren der partitionsinternen Parallelität für Abfragen

Zur Aktivierung der partitionsinternen Abfrageparallelität müssen Sie einen oder mehrere Konfigurationsparameter der Datenbank bzw. des Datenbankmanagers, Vorkompilierungs- und Bindeoptionen oder ein Sonderregister ändern. Alternativ dazu können Sie die Option MAXIMUM DEGREE der Anweisung CREATE WORKLOAD oder ALTER WORKLOAD oder die Prozedur ADMIN_SET_INTRA-_PARALLEL verwenden, um die partitionsinterne Parallelität auf Transaktionsebene zu aktivieren bzw. zu inaktivieren.

Vorbereitende Schritte

Verwenden Sie die folgenden Steuerelemente, um den Grad der partitionsinternen Parallelität anzugeben, den das Optimierungsprogramm verwenden soll.

- Sonderregister CURRENT DEGREE (für dynamisches SQL)
- Bindeoption DEGREE (für statisches SQL)
- Konfigurationsparameter **dft_degree** des Datenbankmanagers (gibt den Standardwert für die beiden vorherigen Parameter an)

Verwenden Sie die folgenden Steuerelemente, um den Grad der partitionsinternen Parallelität während der Laufzeit zu begrenzen. Die Laufzeiteinstellungen überschreiben die Einstellungen des Optimierungsprogramms.

- Konfigurationsparameter **max_querydegree** des Datenbankmanagers
- Befehl SET RUNTIME DEGREE
- Workloadoption MAXIMUM DEGREE

Verwenden Sie die folgenden Steuerelemente, um die partitionsinterne Parallelität zu aktivieren oder zu inaktivieren:

- Konfigurationsparameter **intra_parallel** des Datenbankmanagers
- Gespeicherte Prozedur ADMIN_SET_INTRA_PARALLEL
- Workloadoption MAXIMUM DEGREE (1 definiert)

Informationen zu diesem Vorgang

Verwenden Sie zum Ermitteln der Werte für individuelle Einträge in einer bestimmten Datenbank oder der Konfigurationsdatei der Instanz die Befehle **GET DATABASE CONFIGURATION** und **GET DATABASE MANAGER CONFIGURATION**. Zur Änderung eines oder mehrerer dieser Einträge wird der Befehl **UPDATE DATABASE CONFIGURATION** oder der Befehl **UPDATE DATABASE MANAGER CONFIGURATION** verwendet.

intra_parallel
Dieser Konfigurationsparameter des Datenbankmanagers gibt an, ob der Datenbankmanager partitionsinterne Parallelität verwenden kann. Der Standardwert ist NO, d. h., Anwendungen in dieser Instanz werden ohne partitionsinterne Parallelität ausgeführt. Beispiel:

```
update dbm cfg using intra_parallel yes;
get dbm cfg;
```

max_querydegree
Dieser Konfigurationsparameter des Datenbankmanagers gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf dieser Instanz ausgeführt werden. Eine SQL-Anweisung verwendet keinen höheren Wert als diesen für die Ausführung paralleler Operationen innerhalb einer Datenbankpartition. Der Standardwert ist -1. Dies bedeutet, dass das System den vom Optimierungsprogramm festgelegten Grad partitionsinterner Parallelität verwendet, nicht den benutzerdefinierten Wert. Beispiel:

```
update dbm cfg using max_querydegree any;
get dbm cfg;
```

Der Konfigurationsparameter **intra_parallel** des Datenbankmanagers muss zudem auf den Wert YES gesetzt sein, damit der Wert des Parameters **max_querydegree** verwendet wird.

dft_degree
Ein Datenbankkonfigurationsparameter, der den Standardwert für die Vorkompilierungs- bzw. Bindeoption **DEGREE** und das Sonderregister CURRENT DEGREE angibt. Der Standardwert lautet 1. Der Wert -1 (oder ANY) bedeutet, dass das System den Grad partitionsinterner Parallelität verwendet, der durch das Optimierungsprogramm festgelegt wird. Beispiel:

```
connect to sample;
update db cfg using dft_degree -1;
get db cfg;
connect reset;
```

DEGREE Vorkompilierungs- oder Bindeoption, die den Grad partitionsinterner Parallelität für die Ausführung von statischen SQL-Anweisungen auf einem SMP-System (SMP = symmetrischer Mehrprozessorbetrieb) angibt. Beispiel:

```
connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...
```

CURRENT DEGREE

Ein Sonderregister, das den Grad partitionsinterner Parallelität für die Ausführung von dynamischen SQL-Anweisungen angibt. Verwenden Sie die Anweisung SET CURRENT DEGREE, um dem Sonderregister CURRENT DEGREE einen Wert zuzuordnen. Beispiel:

```
connect to sample;
set current degree = '1';
connect reset;
```

Der Konfigurationsparameter **intra_parallel** des Datenbankmanagers muss zudem auf den Wert YES gesetzt sein, damit die partitionsinterne Parallelität verwendet werden kann. Wird NO definiert, wird der Wert dieses Sonderregisters ignoriert und die Anweisung verwendet keine partitionsinterne Parallelität. Der Wert für den Konfigurationsparameter **intra_parallel** des Datenbankmanagers und der Wert für das Sonderregister CURRENT DEGREE können für eine Workload überschrieben werden, indem das Workloadattribut MAXIMUM DEGREE definiert wird.

MAXIMUM DEGREE

Eine Option der Anweisung CREATE WORKLOAD (oder der Anweisung ALTER WORKLOAD), die den maximalen Laufzeitgrad der Parallelität für eine Workload angibt.

Beispiel: Angenommen, bank_trans ist eine Standardsoftware, die hauptsächlich kurze OLTP-Transaktionen ausführt, und bank_report ist eine weitere Standardsoftware, die komplexe Abfragen zur Generierung eines Business-Intelligence-Berichts (BI-Berichts) ausführt. Keine der beiden Anwendungen kann modifiziert werden und für beide besteht eine Datenbankbindung mit dem Wert 4 für den Grad partitionsinterner Parallelität. Während bank_trans ausgeführt wird, erfolgt eine Zuordnung zur Workload trans, wodurch die partitionsinterne Parallelität inaktiviert wird. Diese OLTP-Anwendung wird ohne die Leistungseinbußen ausgeführt, die in Verbindung mit dem Aufwand für die partitionsinterne Parallelität auftreten. Während bank_report ausgeführt wird, erfolgt eine Zuordnung zur Workload bi. Hierdurch wird die partitionsinterne Parallelität aktiviert und der Wert 8 für den maximalen Laufzeitgrad angegeben. Da der für diese Software kompilierte Grad 4 lautet, werden die statischen SQL-Anweisungen in dieser Anwendung nur mit dem Grad 4 ausgeführt. Wenn diese BI-Anwendung dynamische SQL-Anweisungen enthält und für das Sonderregister CURRENT DEGREE der Wert 16 definiert ist, werden diese Anweisungen mit dem Grad 8 ausgeführt.

```
connect to sample;

create workload trans
  applname('bank_trans')
  maximum degree 1
  enable;

create workload bi
  applname('bank_report')
```

```
  maximum degree 8
  enable;

connect reset;
```

ADMIN_SET_INTRA_PARALLEL

Eine Prozedur, die die partitionsinterne Parallelität für eine Datenbankanwendung aktiviert oder inaktiviert. Die Prozedur wird zwar in der aktuellen Transaktion aufgerufen, wird jedoch erst mit Beginn der nächsten Transaktion wirksam. Beispiel: Angenommen, der nachfolgend aufgeführte Code ist Teil der Anwendung `demoapp`, die die Prozedur ADMIN_SET_INTRA_PARALLEL sowohl mit statischen als auch mit dynamischen SQL-Anweisungen verwendet:

```
EXEC SQL CONNECT TO prod;

// Partitionsinterne Parallelität inaktivieren:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('NO');
// Commit durchführen, damit dieser Aufruf von der
// nächsten Anweisung an wirksam wird:
EXEC SQL COMMIT;

// Alle Anweisungen in den nächsten zwei Transaktionen
// werden ohne partitionsinterne Parallelität ausgeführt:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR rstmt;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :deptname;
EXEC SQL CLOSE c1;
...
// Neuer Abschnitt für diese statische Anweisung:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;

// Partitionsinterne Parallelität aktivieren:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('YES');
// Commit durchführen, damit dieser Aufruf von der
// nächsten Anweisung an wirksam wird:
EXEC SQL COMMIT;

strcpy(stmt, "SET CURRENT DEGREE='4'");
// Grad der Parallelität auf 4 setzten:
EXEC SQL EXECUTE IMMEDIATE :stmt;

// Alle dynamischen Anweisungen in den nächsten zwei Transaktionen
// werden mit partitionsinterner Parallelität und Grad 4 ausgeführt:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
EXEC SQL DECLARE c2 CURSOR FOR rstmt;
EXEC SQL OPEN c2;
EXEC SQL FETCH c2 INTO :deptname;
EXEC SQL CLOSE c2;...
// Alle statischen Anweisungen in den nächsten zwei Transaktionen
// werden mit partitionsinterner Parallelität und Grad 2 ausgeführt:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;
```

Der Grad partitionsinterner Parallelität für dynamische SQL-Anweisungen wird über das Sonderregister CURRENT DEGREE angegeben; für statische SQL-Anweisungen wird er über die Bindeoption DEGREE angegeben. Die folgenden Befehle werden zur Vorbereitung und Bindung der Anwendung `demoapp` verwendet:

```
connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...
```

Verwaltung der Datenserverkapazität

Wenn die Kapazität des Datenservers nicht Ihren gegenwärtigen oder zukünftigen Anforderungen genügt, können Sie die Kapazität des Datenservers erweitern, indem Sie Plattenspeicherplatz hinzufügen und weitere Container erstellen oder Hauptspeicher hinzufügen. Wenn diese einfachen Strategien nicht die erforderliche Kapazität erbringen, können Sie auch das Hinzufügen von Prozessoren oder physischen Partitionen in Betracht ziehen.

Wenn Sie Ihr System durch eine Änderung der Umgebung skalieren, sollten Sie sich über die Auswirkungen dieser Änderung auf die Prozeduren in Ihrer Datenbank wie zum Beispiel auf das Laden von Daten oder das Backup oder den Restore von Datenbanken im Klaren sein.

Hinzufügen von Prozessoren

Wenn eine Konfiguration mit Einzelpartitionsdatenbanken und einem Einzelprozessor bis zur maximalen Kapazitätsgrenze ausgelastet wird, können Sie entweder Prozessoren oder logische Partitionen hinzufügen. Der Vorteil hinzugefügter Prozessoren liegt in der größeren Verarbeitungskapazität. Bei einer Konfiguration mit einer Einzelpartitionsdatenbank mit mehreren Prozessoren (SMP) werden Speicher und Speichersystemressourcen von den Prozessoren gemeinsam genutzt. Alle Prozessoren befinden sich in einem einzigen System, sodass die Auslastung nicht durch Kommunikation oder die Koordination von Tasks die Workload erhöht wird. Dienstprogramme, z. B. Programme zum Laden, Backup und Restore, können die zusätzlichen Prozessoren vorteilhaft nutzen.

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

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

- Standardgrad (**dft_degree**)
- Maximaler Grad der Parallelität bei Abfragen (**max_querydegree**)
- Partitionsinterne Parallelität aktivieren (**intra_parallel**)

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

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

Hinzufügen zusätzlicher Computer

Wenn Sie eine vorhandene Umgebung mit partitionierten Datenbanken haben, können Sie die Verarbeitungskapazität und die Datenspeicherkapazität erhöhen, indem Sie der Umgebung zusätzliche Computer (entweder jeweils mit einem oder mit mehreren Prozessoren) sowie Speicherressourcen hinzufügen. Die Hauptspeicher-

und Massenspeicherressourcen werden unter den Computern nicht gemeinsam genutzt. Diese Option hat den Vorteil, dass Daten und Benutzerzugriffe auf Speichereinheiten und Computer verteilt werden.

Nach dem Hinzufügen der neuen Computer und Speichereinheiten verwenden Sie den Befehl **START DATABASE MANAGER**, um den neuen Computern neue Datenbankpartitionsserver hinzuzufügen. Für jede Datenbank in der Instanz auf jedem neuen Datenbankpartitionsserver, den Sie hinzufügen, wird eine neue Datenbankpartition erstellt und konfiguriert. In den meisten Fällen brauchen Sie die Instanz nach dem Hinzufügen der neuen Datenbankpartitionsserver nicht erneut zu starten.

Fast Communications Manager

Fast Communications Manager (Windows)

In Umgebungen mit mehreren Membern hat jedes Member ein Paar FCM-Dämonen zur Unterstützung der mit Agentenanforderungen verbundenen Kommunikation zwischen Membern. Der eine Dämon dient zum Senden von Übertragungen, der andere zum Empfangen. Diese Dämonen und die unterstützende Infrastruktur werden aktiviert, wenn eine Instanz gestartet wird. Die FCM-Kommunikation wird außerdem für Agenten verwendet, die innerhalb desselben Members aktiv sind. Dieser Typ von Kommunikation wird auch als memberinterne Kommunikation bezeichnet.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter **fcm_num_buffers** des Datenbankmanagers festlegen. Sie können die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter **fcm_num_channels** des Datenbankmanagers festlegen. Standardmäßig sind die Konfigurationsparameter **fcm_num_buffers** und **fcm_num_channels** des Datenbankmanagers auf den Wert AUTOMATIC gesetzt. Bei der Einstellung AUTOMATIC, die auch die empfohlene Einstellung ist, überwacht FCM die Ressourcennutzung und passt Ressourcen an den Workloadbedarf an.

Fast Communications Manager (Linux und UNIX)

Fast Communications Manager (FCM) stellt die Kommunikationsunterstützung für partitionierte Datenbankumgebungen zur Verfügung.

In Umgebungen mit mehreren Membern hat jedes Member ein Paar FCM-Dämonen zur Unterstützung der mit Agentenanforderungen verbundenen Kommunikation zwischen Membern. Der eine Dämon dient zum Senden von Übertragungen, der andere zum Empfangen. Diese Dämonen und die unterstützende Infrastruktur werden aktiviert, wenn eine Instanz gestartet wird. Die FCM-Kommunikation wird außerdem für Agenten verwendet, die innerhalb desselben Members aktiv sind. Dieser Typ von Kommunikation wird auch als memberinterne Kommunikation bezeichnet.

Der FCM-Dämon erfasst Informationen zu Kommunikationsaktivitäten. Mithilfe des Datenbanksystemmonitors können Sie Informationen zur FCM-Kommunikation abrufen. Schlägt die Kommunikation zwischen Membern fehl oder wird die Kommunikation zwischen Membern wiederhergestellt, aktualisieren die FCM-Dämonen Monitorelemente mit diesen Informationen. Die FCM-Dämonen können auch die entsprechende Aktion für dieses Ereignis auslösen. Ein Beispiel für eine entsprechende Aktion ist die Rollback-Operation für eine betroffene Transaktion. Sie können den Datenbanksystemmonitor verwenden, um Unterstützung beim Einstellen der FCM-Konfigurationsparameter zu erhalten.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter **fcm_num_buffers** des Datenbankmanagers festlegen. Sie können die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter **fcm_num_channels** des Datenbankmanagers festlegen. Standardmäßig sind die Konfigurationsparameter **fcm_num_buffers** und **fcm_num_channels** des Datenbankmanagers auf den Wert AUTOMATIC gesetzt. Bei der Einstellung AUTOMATIC, die auch die empfohlene Einstellung ist, überwacht FCM die Ressourcennutzung und passt Ressourcen an den Workloadbedarf an.

Aktivieren der Kommunikation zwischen Datenbankpartitionen durch FCM-Kommunikation

In einer partitionierten Datenbankumgebung wird ein Großteil der Kommunikation zwischen Datenbankpartitionen von FCM (Fast Communication Manager) verarbeitet.

Zur Aktivierung von FCM in einer Datenbankpartition und zur Ermöglichung der Kommunikation mit anderen Datenbankpartitionen müssen Sie einen Serviceeintrag in der Datei services des Verzeichnisses etc der Datenbankpartition erstellen. Die Vorgehensweise wird in diesem Abschnitt erläutert. FCM verwendet den angegebenen Port für die Kommunikation. Wenn mehrere Datenbankpartitionen auf demselben Host definiert sind, müssen Sie einen Portbereich definieren, wie später in diesem Abschnitt erläutert.

Bevor Sie versuchen, Speicher für FCM manuell zu konfigurieren, wird empfohlen, mit der Einstellung AUTOMATIC für die Anzahl von FCM-Puffern (**fcm_num_buffers**) und die Anzahl von FCM-Kanälen (**fcm_num_channels**) zu beginnen, die gleichzeitig die Standardeinstellung ist. Stellen Sie mithilfe der Systemmonitordaten für die FCM-Aktivität fest, ob diese Einstellung geeignet ist.

Windows-Aspekte

Der TCP/IP-Portbereich wird durch folgende Programme automatisch der Datei 'services' hinzugefügt:

- Das Installationsprogramm, wenn es die Instanz erstellt oder eine neue Datenbankpartition hinzufügt.
- Das Dienstprogramm **db2icrt**, wenn es eine neue Instanz erstellt.
- Das Dienstprogramm **db2ncrt**, wenn es die erste Datenbankpartition auf dem Computer hinzufügt.

Der Serviceeintrag hat folgende Syntax:

```
DB2_instanz port/tcp #kommentar
```

DB2_*instanz*

Der Wert für *instanz* ist der Name der Datenbankmanagerinstanz. Alle Zeichen des Namens müssen in Kleinschreibung angegeben sein. Wenn der Instanzname DB2PUSER lautet, geben Sie DB2_db2puser an.

port/**tcp**

Der TCP/IP-Port, den Sie für die Datenbankpartition reservieren wollen.

#*kommentar*

Ein beliebiger Kommentar, den Sie dem Eintrag hinzufügen wollen. Dem Kommentar muss ein Nummernzeichen (#) vorangestellt werden.

Wenn die Datei services des Verzeichnisses etc gemeinsam genutzt wird, müssen Sie sicherstellen, dass die Anzahl der in der Datei zugeordneten Ports größer oder gleich der größten Anzahl mehrerer Datenbankpartitionen in dieser Instanz ist.

Stellen Sie bei der Zuordnung von Ports außerdem sicher, dass Sie jeden möglichen Prozessor, der als Ersatz verwendet werden kann, mit berücksichtigen.

Wenn die Datei services im Verzeichnis etc nicht gemeinsam genutzt wird, gelten dieselben Überlegungen, jedoch mit einem zusätzlichen Gesichtspunkt: Sie müssen sicherstellen, dass die für die DB2-Datenbankinstanz definierten Einträge in allen Dateien services des Verzeichnisses etc die gleichen sind (obwohl andere Einträge, die sich nicht auf Ihre partitionierte Datenbankumgebung beziehen, nicht gleich sein müssen).

Wenn Sie mehrere Datenbankpartitionen auf demselben Host in einer Instanz haben, müssen Sie mehr als einen Port für die Verwendung durch FCM definieren. Dazu müssen zwei Zeilen in die Datei services des Verzeichnisses etc eingefügt werden, die den Portbereich angeben, den Sie zuordnen. Die erste Zeile gibt den ersten Port an und die zweite Zeile das Ende des Portblocks. Im folgenden Beispiel werden fünf Ports für die Instanz SALES zugeordnet. Dies bedeutet, dass kein Prozessor in der Instanz mehr als fünf Datenbankpartitionen hat. Beispiel:

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

Anmerkung: Die Zeichenfolge END muss unbedingt durchgehend in Großschreibung angegeben werden. Außerdem müssen unbedingt beide Unterstreichungszeichen (_) vorhanden sein.

Aktivieren der Kommunikation zwischen Datenbankpartitionsservern (Linux und UNIX)

Diese Task beschreibt, wie die Kommunikation zwischen den Datenbankpartitionsservern aktiviert wird, die Ihrem partitionierten Datenbanksystem angehören.

Die Kommunikation zwischen Datenbankpartitionsservern wird von Fast Communications Manager (FCM) gesteuert. Zum Aktivieren von FCM muss auf jedem Computer im partitionierten Datenbanksystem ein Port oder Portbereich in der Datei /etc/services reserviert sein.

Vorbereitende Schritte

Sie müssen über eine Benutzer-ID mit Rootberechtigung verfügen.

Sie müssen diese Task auf allen Computern ausführen, die an der Instanz beteiligt sind.

Informationen zu diesem Vorgang

Die Anzahl der für FCM zu reservierenden Ports entspricht der maximalen Anzahl von Datenbankpartitionen, die von einem der Computer in der Instanz per Hosting bereitgestellt wird oder potenziell bereitgestellt werden kann.

Die Datei db2nodes.cfg im folgenden Beispiel enthält diese Einträge:

```
0 server1 0
1 server1 1
2 server2 0
3 server2 1
4 server2 2
```

```
5 server3 0
6 server3 1
7 server3 2
8 server3 3
```

Im Beispiel wird davon ausgegangen, dass die FCM-Ports beginnend mit 60000 nummeriert sind. In dieser Situation gilt Folgendes:

- server1 verwendet zwei Ports (60000, 60001) für seine zwei Datenbankpartitionen
- server2 verwendet drei Ports (60000, 60001, 60002) für seine drei Datenbankpartitionen
- server3 verwendet vier Ports (60000, 60001, 60002, 60003) für seine vier Datenbankpartitionen

Alle Computer müssen die Ports 60000, 60001, 60002 und 60003 reservieren, da diese den größten Portbereich bilden, der für einen der Computer in der Instanz erforderlich ist.

Wenn Sie eine Hochverfügbarkeitslösung wie Tivoli System Automation oder IBM PowerHA SystemMirror for AIX verwenden, um Datenbankpartitionen durch einen Failover von einem Computer auf einen anderen zu übertragen, müssen Sie die möglichen Portanforderungen berücksichtigen. Wenn ein Computer zum Beispiel normalerweise vier Datenbankpartitionen per Hosting bereitstellt, er jedoch potenziell zwei Datenbankpartitionen eines anderen Computers übernehmen kann, müssen für diesen Computer sechs Ports geplant werden.

Wenn Sie eine Instanz erstellen, wird auf dem Primärcomputer ein Portbereich reserviert. Der Primärcomputer wird auch als Instanzeigner bezeichnet. Wenn der Portbereich, der der `/etc/services`-Datei ursprünglich hinzugefügt wurde, für Ihren Bedarf jedoch nicht ausreichend ist, müssen Sie den Bereich der reservierten Ports durch manuelles Hinzufügen weiterer Einträge erweitern.

Vorgehensweise

Gehen Sie wie folgt vor, um die Kommunikation zwischen Servern in einer Umgebung mit partitionierten Datenbanken unter Verwendung von `/etc/services` zu aktivieren:

1. Melden Sie sich am Primärcomputer (dem Computer, dem die Instanz gehört) als Benutzer mit Rootberechtigung an.
2. Erstellen Sie eine Instanz.
3. Zeigen Sie den Standardportbereich an, der in der Datei `/etc/services` reserviert wurde. Zusätzlich zur Basiskonfiguration sollten die FCM-Ports etwa wie folgt aussehen:

   ```
   db2c_db2inst1         50000/tcp
   #Add FCM port information
   DB2_db2inst1        60000/tcp
   DB2_db2inst1_1        60001/tcp
   DB2_db2inst1_2        60002/tcp
   DB2_db2inst1_END      60003/tcp
   ```

   Standardmäßig ist der erste Port (50000) für Verbindungsanforderungen reserviert. Die ersten vier verfügbaren Ports über 60000 sind standardmäßig für die FCM-Kommunikation reserviert. Ein Port ist für den Datenbankpartitionsserver reserviert, der als Instanzeigner fungiert. Drei Ports sind für Server logischer Datenbankpartitionen reserviert, die Sie nach Abschluss der Installation möglicherweise dem Computer hinzufügen möchten.

Der Portbereich muss einen Start- und einen Endeintrag aufweisen. Zwischeneinträge sind optional. Das explizite Aufnehmen von Zwischenwerten kann nützlich sein, um die Verwendung dieser Ports durch andere Anwendungen zu verhindern. Diese Einträge werden vom Datenbankmanager jedoch nicht geprüft.

Die DB2-Porteinträge verwenden das folgende Format:

```
DB2_instanzname_suffix  portnummer/tcp  # kommentar
```

Dabei gilt Folgendes:

- *instanzname* ist der Name der partitionierten Instanz.
- *suffix* wird für den ersten FCM-Port nicht verwendet. Zwischeneinträge sind die Einträge zwischen dem niedrigsten und dem höchsten Port. Wenn Sie die Zwischeneinträge zwischen dem ersten und dem letzten FCM-Port einfügen, besteht das *suffix* aus einer ganzen Zahl, die für jeden weiteren Port jeweils um eins erhöht wird. Auf diese Weise wird der zweite Port beispielsweise mit der Zahl 1 nummeriert, der dritte Port mit der Zahl 2 und so weiter, um die Eindeutigkeit zu gewährleisten. Für den letzten Eintrag muss das Wort `END` als *suffix* verwendet werden.
- *portnummer* ist die Portnummer, die Sie für die Kommunikation zwischen Datenbankpartitionsservern reservieren.
- *kommentar* ist ein optionaler Kommentar, der einen Eintrag beschreibt.

4. Stellen Sie sicher, dass für die FCM-Kommunikation ausreichend Ports reserviert sind. Wenn der Bereich der reservierten Ports nicht ausreichend ist, fügen Sie der Datei neue Einträge hinzu.
5. Melden Sie sich nacheinander jeweils an allen zur Instanz zugehörigen Computern als Benutzer mit Rootberechtigung an und fügen Sie der Datei `/etc/services` identische Einträge hinzu.

Kapitel 10. Erstellen und Verwalten von Umgebungen mit partitionierten Datenbanken

Verwalten von Datenbankpartitionen

Sie können Partitionen starten oder stoppen, Partitionen löschen oder für diese einen Trace durchführen.

Vorbereitende Schritte

Zur Arbeit mit Datenbankpartitionen müssen Sie über die Berechtigung zum Herstellen der Verbindung (ATTACH) zu einer Instanz verfügen. Jeder Benutzer mit der Berechtigung SECADM oder ACCESSCTRL kann Ihnen die Berechtigung zum Zugriff auf eine bestimmte Instanz erteilen.

Vorgehensweise

- Verwenden Sie den Befehl **START DATABASE MANAGER** oder den Befehl **STOP DATABASE MANAGER** mit dem Parameter **DBPARTITIONNUM**, um eine bestimmte Datenbankpartition zu starten oder zu stoppen.
- Verwenden Sie den Befehl **STOP DATABASE MANAGER** mit dem Parameter **DROP DBPARTITIONNUM**, um eine bestimmte Datenbankpartition aus der Konfigurationsdatei `db2nodes.cfg` zu löschen. Vor der Verwendung des Parameters **DROP DBPARTITIONNUM** müssen Sie den Befehl **DROP DBPARTITIONNUM VERIFY** ausführen, um sicherzustellen, dass keine Benutzerdaten in dieser Datenbankpartition vorhanden sind.
- Verwenden Sie die vom IBM Support angegebenen Optionen, wenn Sie einen Trace der Aktivitäten auf einer Datenbankpartition erstellen möchten.

  Achtung: Verwenden Sie das Tracedienstprogramm nur dann, wenn Sie vom IBM Support oder von einem Beauftragten der technischen Unterstützung dazu aufgefordert werden.
  Das Tracedienstprogramm zeichnet Daten zu DB2 for Linux, UNIX and Windows-Operationen auf und formatiert sie. Weitere Einzelheiten finden Sie im Abschnitt zum „Tracebefehl 'db2trc'".

Hinzufügen von Datenbankpartitionen in Umgebungen mit partitionierten Datenbanken

Sie können einem System mit einer partitionierten Datenbank Datenbankpartitionen hinzufügen, wenn es aktiv ist oder wenn es gestoppt ist. Da das Hinzufügen eines neuen Servers sehr zeitaufwändig sein kann, ist es vielleicht wünschenswert, dies zu tun, wenn der Datenbankmanager bereits aktiv ist.

Verwenden Sie den Befehl **ADD DBPARTITIONNUM**, um einem System eine Datenbankpartition hinzuzufügen. Dieser Befehl kann folgendermaßen aufgerufen werden:

- Als Option im Befehl **START DBM**
- Mit dem Befehl **ADD DBPARTITIONNUM**
- Mit der API 'sqleaddn'
- Mit der API 'sqlepstart'

Wenn Ihr System gestoppt ist, verwenden Sie den Befehl **START DBM**. Wenn es aktiv ist, können Sie eine der anderen Möglichkeiten verwenden.

Wenn Sie dem System mit dem Befehl **ADD DBPARTITIONNUM** eine neue Datenbankpartition hinzufügen, werden alle in dieser Instanz vorhandenen Datenbanken auf die neue Datenbankpartition erweitert. Sie können auch angeben, welche Container für temporäre Tabellenbereiche für die Datenbanken verwendet werden sollen. Für die Container gibt es folgende Möglichkeiten:

- Es sind dieselben Container wie die, die für die Katalogpartition der jeweiligen Datenbanken definiert sind (Standardeinstellung).
- Es sind dieselben Container wie die für eine andere Datenbankpartition definierten.
- Sie wurden noch nicht erstellt. Sie müssen jeder Datenbank mithilfe der Anweisung ALTER TABLESPACE Tabellenbereichscontainer für temporäre Tabellen hinzufügen, bevor die Datenbank verwendet werden kann.

Anmerkung: Nicht katalogisierte Datenbanken werden beim Hinzufügen einer neuen Datenbankpartition nicht erkannt. Die nicht katalogisierte Datenbank ist dann in der neuen Datenbankpartition nicht enthalten. Beim Versuch, eine Verbindung zu der Datenbank in der neuen Datenbankpartition herzustellen, wird die Fehlernachricht SQL1013N zurückgegeben.

Sie können eine Datenbank in der neuen Datenbankpartition nicht zur Speicherung von Daten verwenden, bevor nicht mindestens eine Datenbankpartitionsgruppe zur Aufnahme der neuen Datenbankpartition geändert wurde.

Sie können nicht von einer Einzelpartitionsdatenbank zu einer Mehrpartitionsdatenbank wechseln, indem Sie Ihrem System eine Partition hinzufügen, da für die Umverteilung von Daten zwischen Datenbankpartitionen für jede betroffene Tabelle ein entsprechender Verteilungsschlüssel erforderlich ist. Beim Erstellen einer Tabelle in einer Mehrpartitionsdatenbank werden die Verteilungsschlüssel automatisch generiert. In einer Einzelpartitionsdatenbank können Verteilungsschlüssel mithilfe der SQL-Anweisungen CREATE TABLE oder ALTER TABLE explizit erstellt werden.

Anmerkung: Wenn keine Datenbanken im System definiert sind und Sie Enterprise Server Edition unter einem UNIX-Betriebssystem ausführen, fügen Sie durch Bearbeiten der Datei `db2nodes.cfg` eine neue Definition für eine Datenbankpartition hinzu. Verwenden Sie keine der beschriebenen Prozeduren, da sie nur gelten, wenn eine Datenbank vorhanden ist.

Für Windows ist Folgendes zu beachten: Wenn Sie Enterprise Server Edition unter einem Windows-Betriebssystem verwenden und keine Datenbanken in der Instanz definiert haben, verwenden Sie den Befehl **db2ncrt**, um das Datenbanksystem zu skalieren. Wenn bereits Datenbanken vorhanden sind, verwenden Sie den Befehl **START DBM ADD DBPARTITIONNUM**, um sicherzustellen, dass beim Skalieren des Systems für jede vorhandene Datenbank eine Datenbankpartition erstellt wird. Bearbeiten Sie unter Windows-Betriebssystemen die Datei für die Datenbankpartitionskonfiguration (`db2nodes.cfg`) nicht manuell, da dies zu Inkonsistenzen in der Datei führen kann.

Hinzufügen einer Onlinedatenbankpartition

Sie können neue Datenbankpartitionen, die online sind, einer Umgebung mit partitionierten Datenbanken hinzufügen, während diese aktiv ist und Anwendungen mit Datenbanken verbunden sind.

Vorgehensweise

Gehen Sie wie folgt vor, um unter Verwendung der Befehlszeile einem aktiven Datenbankmanager eine Onlinedatenbankpartition hinzuzufügen:

1. Führen Sie den Befehl **START DBM** in jeder vorhandenen Datenbankpartition aus.

   Geben Sie auf allen Plattformen die neuen Datenbankpartitionswerte für die Parameter **DBPARTITIONNUM**, **ADD DBPARTITIONNUM**, **HOSTNAME**, **PORT** und **NETNAME** an. Auf der Windows-Plattform geben Sie außerdem die Parameter **COMPUTER**, **USER** und **PASSWORD** an.

   Sie können auch die Quelle für die Definitionen von Tabellenbereichscontainern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichscontainer für temporäre Tabellen von der Katalogpartition der einzelnen Datenbanken abgerufen.

   Mit den folgenden Befehlen werden einer vorhandenen Datenbank beispielsweise drei neue Datenbankpartitionen hinzugefügt:

   ```
   START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
    PORT PORT3;
   START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
    PORT PORT4;
   START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
    PORT PORT5;
   ```

2. Optional: Ändern Sie die Datenbankpartitionsgruppe, um die neue Datenbankpartition einzubeziehen. Diese Aktion kann auch erwogen werden, wenn die Daten auf die neue Datenbankpartition umverteilt werden.
3. Optional: Verteilen Sie Daten auf die neue Datenbankpartition um. Diese Aktion sollte aber eigentlich in jedem Fall ausgeführt werden, wenn Sie von den neuen Datenbankpartitionen profitieren wollen. Sie können die Option zum Ändern der Datenbankpartitionsgruppe auch als Teil der Umverteilungsoperation einbeziehen. Andernfalls muss das Ändern der Datenbankpartitionsgruppe zur Einbeziehung der neuen Datenbankpartitionen als separate Aktion ausgeführt werden, bevor die Daten auf die neuen Datenbankpartitionen umverteilt werden.
4. Optional: Führen Sie ein Backup aller Datenbanken in der neuen Datenbankpartition durch. Diese Aktion ist optional, doch sind Backups der neuen und besonders der anderen Datenbankpartitionen sinnvoll, wenn Sie die Daten über die alten und neuen Datenbankpartitionen umverteilt haben.

Einschränkungen beim Onlineverfahren zum Hinzufügen einer Datenbankpartition

Der Status, in dem sich die neue Datenbankpartition befindet, nachdem Sie zur Instanz hinzugefügt wurde, hängt vom Status der ursprünglichen Datenbankpartition ab. Nachdem die neue Datenbankpartition zur Instanz hinzugefügt wurde, kann ihr Vorhandensein Anwendungen, die WITH HOLD-Cursor verwenden, bekannt sein oder auch nicht.

Wenn Sie eine neue Datenbankpartition zu einer Instanz einer Einzelpartitionsdatenbank hinzufügen, gibt es folgende Möglichkeiten:

- Wenn die ursprüngliche Datenbankpartition aktiv ist, während die Datenbankpartition hinzugefügt wird, ist die neue Datenbankpartition nach Abschluss des Hinzufügens der Datenbankpartition inaktiv.

- Wenn die ursprüngliche Datenbankpartition inaktiv ist, während die Datenbankpartition hinzugefügt wird, ist die neue Datenbankpartition nach Abschluss des Hinzufügens der Datenbankpartition aktiv.

Den Anwendungen, die WITH HOLD-Cursor verwenden und die vor dem Hinzufügen der Datenbankpartition gestartet wurden, ist die neue Datenbankpartition nach Abschluss des Hinzufügens der Datenbankpartition nicht bekannt. Wenn die WITH HOLD-Cursor geschlossen werden, bevor die Datenbankpartition hinzugefügt wird, ist die neue Datenbankpartition den Anwendungen nach Abschluss des Hinzufügens der Datenbankpartition bekannt.

Hinzufügen einer Datenbankpartition offline (Windows)

Sie können einem partitionierten Datenbanksystem neue Datenbankpartitionen hinzufügen, während es gestoppt ist. Die neu hinzugefügte Datenbankpartition wird erst nach einem Neustart des Datenbankmanagers für alle Datenbanken verfügbar.

Vorbereitende Schritte

- Sie müssen den neuen Server installieren, bevor Sie eine Datenbankpartition auf diesem Server erstellen können.
- Setzen Sie den Standardwert der Registrierdatenbankvariablen **DB2_FORCE_OFFLINE_ADD_PARTITION** auf TRUE, um zu erzwingen, dass alle hinzugefügten Datenbankpartitionen offline sind.

Vorgehensweise

Gehen Sie wie folgt vor, um unter Verwendung der Befehlszeile einem gestoppten partitionierten Datenbankserver eine Datenbankpartition hinzuzufügen:

1. Setzen Sie den Befehl **STOP DBM** ab, um alle Datenbankpartitionen zu stoppen.
2. Führen Sie den Befehl **ADD DBPARTITIONNUM** auf dem neuen Server aus.

   Eine Datenbankpartition wird lokal für jede Datenbank erstellt, die bereits im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, sodass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.
3. Führen Sie den Befehl **START DBM** aus, um das Datenbanksystem zu starten. Beachten Sie, dass die Datei für die Datenbankpartitionskonfiguration bereits bei der Installation des neuen Servers durch den Datenbankmanager aktualisiert wurde, um den neuen Server aufzunehmen.
4. Aktualisieren Sie die Konfigurationsdatei in der neuen Datenbankpartition wie folgt:
   a. Führen Sie den Befehl **START DBM** in jeder vorhandenen Datenbankpartition aus.

      Geben Sie die neuen Datenbankpartitionswerte für die Parameter **DBPARTITIONNUM**, **ADD DBPARTITIONNUM**, **HOSTNAME**, **PORT** und **NETNAME** sowie für die Parameter **COMPUTER**, **USER** und **PASSWORD** an.

      Sie können auch die Quelle für die Definitionen von Tabellenbereichscontainern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichscontainer für temporäre Tabellen von der Katalogpartition der einzelnen Datenbanken abgerufen.

      Mit den folgenden Befehlen werden einer vorhandenen Datenbank beispielsweise drei neue Datenbankpartitionen hinzugefügt:

```
START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
 PORT PORT3;
START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
 PORT PORT4;
START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
 PORT PORT5;
```

Wenn der Befehl **START DBM** ausgeführt wurde, wird der neue Server gestoppt.

b. Stoppen Sie den Datenbankmanager durch Ausführen des Befehls **STOP DBM**.

Wenn Sie alle Datenbankpartitionen im System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neuen Datenbankpartitionen. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn **STOP DBM** ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl **ADD DBPARTITIONNUM**, der aufgerufen wird, wenn Sie den Parameter **ADD DBPARTITIONNUM** im Befehl **START DBM** angeben, in den richtigen Datenbankpartitionen ausgeführt wird. Wenn das Dienstprogramm beendet ist, werden die neuen Serverpartitionen gestoppt.

5. Starten Sie den Datenbankmanager durch Ausführen des Befehls **START DBM**.

   Nun werden die hinzugefügten Datenbankpartitionen mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl **START DBM** zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.

6. Optional: Ändern Sie die Datenbankpartitionsgruppe, um die neue Datenbankpartition einzubeziehen. Diese Aktion kann auch erwogen werden, wenn die Daten auf die neue Datenbankpartition umverteilt werden.
7. Optional: Verteilen Sie Daten auf die neue Datenbankpartition um. Diese Aktion sollte aber eigentlich in jedem Fall ausgeführt werden, wenn Sie von der neuen Datenbankpartition profitieren wollen. Sie können die Option zum Ändern der Datenbankpartitionsgruppe auch als Teil der Umverteilungsoperation einbeziehen. Andernfalls muss das Ändern der Datenbankpartitionsgruppe zur Einbeziehung der neuen Datenbankpartition als separate Aktion ausgeführt werden, bevor die Daten auf die neue Datenbankpartition umverteilt werden.
8. Optional: Führen Sie ein Backup aller Datenbanken in der neuen Datenbankpartition durch. Diese Aktion ist optional, doch sind Backups der neuen und besonders der anderen Datenbankpartitionen sinnvoll, wenn Sie die Daten über die alten und neuen Datenbankpartitionen umverteilt haben.

Hinzufügen einer Datenbankpartition offline (Linux und UNIX)

Sie können einem partitionierten Datenbanksystem neue Datenbankpartitionen hinzufügen, die offline sind. Die neu hinzugefügte Datenbankpartition wird erst nach einem Neustart des Datenbankmanagers für alle Datenbanken verfügbar.

Vorbereitende Schritte

- Falls noch nicht vorhanden, installieren Sie den neuen Server, bevor Sie eine Datenbankpartition auf diesem Server erstellen können.
- Richten Sie den Zugriff auf die ausführbaren Dateien durch Anhängen gemeinsamer Dateisysteme oder lokale Kopien ein.

- Synchronisieren Sie die Betriebssystemdateien mit denen auf vorhandenen Prozessoren.
- Stellen Sie sicher, dass der Zugriff auf das Verzeichnis `sqllib` als gemeinsam genutztes Dateisystem möglich ist.
- Stellen Sie sicher, dass die richtigen Werte für die relevanten Betriebssystemparameter (z. B. die maximale Anzahl von Prozessen) gesetzt sind.
- Registrieren Sie den Hostnamen auf dem Namensserver oder in der Datei `hosts` im Verzeichnis `/etc` auf allen Datenbankpartitionen. Der Hostname des Computers muss in `.rhosts` registriert sein, damit Befehle mit **rsh** oder **rah** fern ausgeführt werden können.
- Setzen Sie den Standardwert der Registrierdatenbankvariablen **DB2_FORCE_OFFLINE_ADD_PARTITION** auf `TRUE`, um zu erzwingen, dass alle hinzugefügten Datenbankpartitionen offline sind.

Vorgehensweise

- Gehen Sie wie folgt vor, um unter Verwendung der Befehlszeile einem gestoppten partitionierten Datenbankserver eine Datenbankpartition hinzuzufügen:
  1. Setzen Sie den Befehl **STOP DBM** ab, um alle Datenbankpartitionen zu stoppen.
  2. Führen Sie den Befehl **ADD DBPARTITIONNUM** auf dem neuen Server aus.

     Eine Datenbankpartition wird lokal für jede Datenbank erstellt, die im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, sodass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.
  3. Führen Sie den Befehl **START DBM** aus, um das Datenbanksystem zu starten. Beachten Sie, dass die Datei für die Datenbankpartitionskonfiguration (`db2nodes.cfg`) bereits bei der Installation des neuen Servers durch den Datenbankmanager aktualisiert wurde, um den neuen Server aufzunehmen.
  4. Aktualisieren Sie die Konfigurationsdatei in der neuen Datenbankpartition wie folgt:
     a. Führen Sie den Befehl **START DBM** in jeder vorhandenen Datenbankpartition aus.

        Geben Sie die neuen Datenbankpartitionswerte für die Parameter **DBPARTITIONNUM**, **ADD DBPARTITIONNUM**, **HOSTNAME**, **PORT** und **NETNAME** sowie für die Parameter **COMPUTER**, **USER** und **PASSWORD** an.

        Sie können auch die Quelle für die Definitionen von Tabellenbereichscontainern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichscontainer für temporäre Tabellen aus der Katalogpartition der einzelnen Datenbanken abgerufen.

        Mit den folgenden Befehlen werden einer vorhandenen Datenbank beispielsweise drei neue Datenbankpartitionen hinzugefügt:

        ```
        START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
         PORT PORT3;
        START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
         PORT PORT4;
        START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
         PORT PORT5;
        ```

        Wenn der Befehl **START DBM** ausgeführt wurde, wird der neue Server gestoppt.

b. Stoppen Sie den gesamten Datenbankmanager durch Ausführen des Befehls **STOP DBM**.

   Wenn Sie alle Datenbankpartitionen im System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neuen Datenbankpartitionen. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn **STOP DBM** ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl **ADD DBPARTITIONNUM**, der aufgerufen wird, wenn Sie den Parameter **ADD DBPARTITIONNUM** im Befehl **START DBM** angeben, in der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm beendet ist, werden die neuen Serverpartitionen gestoppt.

5. Starten Sie den Datenbankmanager durch Ausführen des Befehls **START DBM**.

   Nun wird die hinzugefügte Datenbankpartition mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl **START DBM** zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.

6. Optional: Ändern Sie die Datenbankpartitionsgruppe, um die neue Datenbankpartition einzubeziehen. Diese Aktion kann auch erwogen werden, wenn die Daten auf die neue Datenbankpartition umverteilt werden.
7. Optional: Verteilen Sie Daten auf die neue Datenbankpartition um. Diese Aktion sollte aber eigentlich in jedem Fall ausgeführt werden, wenn Sie von der neuen Datenbankpartition profitieren wollen. Sie können die Option zum Ändern der Datenbankpartitionsgruppe auch als Teil der Umverteilungsoperation einbeziehen. Andernfalls muss das Ändern der Datenbankpartitionsgruppe zur Einbeziehung der neuen Datenbankpartition als separate Aktion ausgeführt werden, bevor die Daten auf die neue Datenbankpartition umverteilt werden.
8. Optional: Führen Sie ein Backup aller Datenbanken in der neuen Datenbankpartition durch. Diese Aktion ist optional, doch sind Backups der neuen und besonders der anderen Datenbankpartitionen sinnvoll, wenn Sie die Daten über die alten und neuen Datenbankpartitionen umverteilt haben.

- Sie können die Konfigurationsdatei auch wie folgt manuell aktualisieren:
  1. Bearbeiten Sie die Datei `db2nodes.cfg` und fügen Sie die neue Datenbankpartition ein.
  2. Setzen Sie den folgenden Befehl ab, um die neue Datenbankpartition zu starten: `START DBM DBPARTITIONNUM` *`partitionsnummer`*

     Geben Sie als Wert für *partitionsnummer* die Nummer an, die Sie der neuen Datenbankpartition zuordnen.
  3. Wenn der neue Server eine logische Partition sein soll (d. h. nicht die Datenbankpartition 0), verwenden Sie den Befehl **db2set**, um die Registrierdatenbankvariable **DBPARTITIONNUM** zu aktualisieren. Geben Sie die Nummer der Datenbankpartition an, die Sie hinzufügen.
  4. Führen Sie den Befehl **ADD DBPARTITIONNUM** in der neuen Datenbankpartition aus.

     Dieser Befehl erstellt lokal eine Datenbankpartition für jede Datenbank, die im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankparti-

tion bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, sodass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.

5. Wenn der Befehl **ADD DBPARTITIONNUM** ausgeführt wurde, setzen Sie den Befehl **START DBM** ab, um die anderen Datenbankpartitionen des Systems zu starten.

   Führen Sie keine Aktionen aus, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank, bis alle Datenbankpartitionen erfolgreich gestartet wurden.

Fehlerbehebung beim Hinzufügen von Datenbankpartitionen

Das Hinzufügen von Datenbankpartitionen schlägt nicht aufgrund nicht vorhandener Pufferpools fehl, weil der Datenbankmanager Systempufferpools erstellt, um eine automatische Standardunterstützung für alle Pufferpoolseitengrößen bereitzustellen.

Wenn allerdings einer dieser Systempufferpools verwendet wird, kann die Leistung erheblich beeinträchtigt werden, da diese Pufferpools sehr klein sind. Wenn ein Systempufferpool verwendet wird, wird eine Nachricht in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben. Systempufferpools werden in Szenarios, in denen Datenbankpartitionen hinzugefügt werden, unter den folgenden Umständen verwendet:

- Sie fügen Datenbankpartitionen einer Umgebung mit partitionierten Datenbanken hinzu, die mindestens einen Tabellenbereich für temporäre Systemtabellen mit einer anderen Seitengröße als die Standardgröße von 4 KB besitzt. Wenn eine Datenbankpartition erstellt wird, ist nur der Pufferpool IBMDEFAULTDP vorhanden und dieser Pufferpool hat eine Seitengröße von 4 KB.

  Betrachten Sie die folgenden Beispiele:

  1. Sie verwenden den Befehl **START DBM**, um der aktuellen Mehrpartitionsdatenbank eine Datenbankpartition hinzuzufügen:

     ```
     START DBM DBPARTITIONNUM 2 ADD DBPARTITIONNUM HOSTNAME newhost PORT 2
     ```

  2. Sie verwenden den Befehl **ADD DBPARTITIONNUM**, nachdem Sie die Datei `db2nodes.cfg` manuell mit der neuen Datenbankpartitionsbeschreibung aktualisiert haben.

  Eine Möglichkeit zur Vermeidung dieses Problems ist die Angabe der Klausel WITHOUT TABLESPACES im Befehl **ADD DBPARTITIONNUM** oder **START DBM**. Verwenden Sie anschließend die Anweisung CREATE BUFFERPOOL mit den entsprechenden Werten für SIZE und PAGESIZE und ordnen Sie dem Pufferpool die Tabellenbereiche für temporäre Systemtabellen mit der Anweisung ALTER TABLESPACE zu.

- Sie fügen Datenbankpartitionen einer vorhandenen Datenbankpartitionsgruppe hinzu, die mindestens einen Tabellenbereich mit einer anderen Seitengröße als der Standardseitengröße von 4 KB hat. Dies geschieht, weil die in der neuen Datenbankpartition erstellten Pufferpools mit einer vom Standard abweichenden Seitengröße für die Tabellenbereiche nicht aktiviert wurden.

  Anmerkung: In früheren Versionen arbeitete dieser Befehl mit dem Schlüsselwort NODEGROUP anstelle der Schlüsselwörter DATABASE PARTITION GROUP.

  Betrachten Sie das folgende Beispiel:

  – Sie verwenden die Anweisung ALTER DATABASE PARTITION GROUP, um einer Datenbankpartitionsgruppe wie folgt eine Datenbankpartition hinzuzufügen:

```
START DBM
CONNECT TO mpp1
ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (2)
```

Eine Möglichkeit, dieses Problems zu vermeiden, besteht darin, Pufferpools für jede Seitengröße zu erstellen und die Verbindung zur Datenbank wieder herzustellen, bevor Sie die folgende Anweisung ALTER DATABASE PARTITION GROUP absetzen:

```
START DBM
CONNECT TO mpp1
CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
CONNECT RESET
CONNECT TO mpp1
ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (2)
```

Anmerkung: Wenn die Datenbankpartitionsgruppe Tabellenbereiche mit der Standardseitengröße hat, wird die Nachricht SQL1759W zurückgegeben.

Löschen von Datenbankpartitionen

Sie können eine Datenbankpartition löschen, die von keiner Datenbank genutzt wird, und den Computer für andere Zwecke freigeben.

Vorbereitende Schritte

Vergewissern Sie sich, dass die Datenbankpartition nicht in Gebrauch ist, indem Sie den Befehl **DROP DBPARTITIONNUM VERIFY** oder die API sqledrpn ausführen.

- Wenn Sie die Nachricht SQL6034W (Die Datenbankpartition wird von keiner Datenbank verwendet) empfangen, können Sie die Datenbankpartition löschen.
- Wenn Sie die Nachricht SQL6035W (Datenbankpartition wird von Datenbank verwendet) empfangen, verwenden Sie den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP**, um die Daten von der Datenbankpartition, die Sie löschen wollen, auf andere Datenbankpartitionen vom Aliasnamen der Datenbank umzuverteilen.

Stellen Sie außerdem sicher, dass alle Transaktionen, die von der betreffenden Datenbankpartition koordiniert wurden, erfolgreich mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht wurden. Dazu ist möglicherweise eine Recovery nach einem Systemabsturz auf anderen Servern erforderlich. Wenn Sie zum Beispiel die Koordinatorpartition löschen und vor dem Löschen der Koordinatorpartition eine andere an einer Transaktion beteiligte Datenbankpartition abgestürzt war, ist die abgestürzte Datenbankpartition nicht in der Lage, die Koordinatorpartition nach dem Ergebnis unbestätigter Transaktionen abzufragen.

Vorgehensweise

Geben Sie in der Befehlszeile Folgendes ein, um eine Datenbankpartition zu löschen:

Zum Löschen der Datenbankpartition geben Sie den Befehl **STOP DBM** mit dem Parameter **DROP DBPARTITIONNUM** ein.
Nach erfolgreicher Ausführung des Befehls wird das System gestoppt. Starten Sie anschließend den Datenbankmanager mit dem Befehl **START DBM**.

Auflisten der Datenbankpartitionsserver einer Instanz (Windows)

Unter Windows können Sie mit dem Befehl **db2nlist** eine Liste der Datenbankpartitionsserver abrufen, die an einer Instanz beteiligt sind.

Informationen zu diesem Vorgang

Der Befehl ist wie folgt anzugeben:

```
db2nlist
```

Wenn Sie den Befehl wie angegeben verwenden, wird die aktuelle Instanz (wie in der Umgebungsvariablen **DB2INSTANCE** festgelegt) als Standardinstanz verwendet. Sie können in dem Befehl auch eine bestimmte Instanz angeben:

```
db2nlist /i:instName
```

Dabei ist *instName* der Name der gewünschten Instanz.

Mit dem folgenden Befehl können Sie außerdem (optional) den Status jedes Datenbankpartitionsservers abfragen:

```
db2nlist /s
```

Jeder Datenbankpartitionsserver kann einen der folgenden Statuswerte aufweisen: Starten, Ausführen, Stoppen, Gestoppt.

Hinzufügen von Datenbankpartitionsservern zu einer Instanz (Windows)

Unter Windows können Sie mit dem Befehl **db2ncrt** einer Instanz einen Datenbankpartitionsserver hinzufügen.

Informationen zu diesem Vorgang

Anmerkung: Wenn die Instanz bereits Datenbanken enthält, darf der Befehl **db2ncrt** nicht verwendet werden. Verwenden Sie in diesem Fall stattdessen den Befehl **START DBM ADD DBPARTITIONNUM**. Dadurch wird sichergestellt, dass die Datenbank dem neuen Datenbankpartitionsserver korrekt hinzugefügt wird. **Editieren Sie AUF KEINEN FALL** die Datei db2nodes.cfg! Dies kann zu Inkonsistenzen in der Umgebung mit partitionierten Datenbanken führen.

Der Befehl hat folgende erforderliche Parameter:

```
db2ncrt /n:partitionsnummer
        /u:benutzername,kennwort
        /p:logischer_port
```

/n:partitionsnummer
Die eindeutige Datenbankpartitionsnummer zum Identifizieren des Datenbankpartitionsservers. Die Nummer kann aus den Werten 1 bis 999 (in aufsteigender Reihenfolge) bestehen.

/u:benutzername,kennwort
Der Name des Anmeldekontos und das Kennwort des DB2-Service.

/p:logischer_port
Die logische Portnummer, die für den Datenbankpartitionsserver verwendet wird, wenn der logische Port ungleich null (0) ist. Wird hier kein Wert angegeben, wird die logische Portnummer '0' zugeordnet.

Die logische Portnummer ist nur dann optional, wenn die erste Datenbankpartition auf einem Computer erstellt wird. Beim Erstellen einer logischen Datenbankpartition müssen Sie diesen Parameter angeben und eine logische Portnummer auswählen, die noch nicht verwendet wird. Es gelten verschiedene Einschränkungen:

- Auf jedem Computer muss ein Datenbankpartitionsserver mit dem logischen Port '0' definiert sein.
- Die Portnummer darf den für die FCM-Kommunikation reservierten Portbereich in der Servicedatei im Verzeichnis %SystemRoot%\system32\drivers\etc nicht überschreiten. Wenn Sie zum Beispiel einen Bereich von vier Ports für die aktuelle Instanz reservieren, ist die höchste zulässige Portnummer 3 (Ports 1, 2 und 3; Port 0 ist der logischen Standarddatenbankpartition zugeordnet). Der Portbereich wird definiert, wenn der Befehl **db2icrt** mit dem Parameter '/r:basisport,endport' angegeben wird.

Außerdem gibt es folgende optionale Parameter:

/g:netzname
Gibt den Netznamen für den Datenbankpartitionsserver an. Wird dieser Parameter nicht angegeben, verwendet DB2 die erste IP-Adresse, die auf dem System festgestellt wird.

Verwenden Sie diesen Parameter, wenn auf einem Computer mehrere IP-Adressen definiert sind und wenn Sie eine bestimmte IP-Adresse für den Datenbankpartitionsserver angeben möchten. Sie können für den Parameter 'netzname' den Netznamen oder die IP-Adresse der gewünschten Einheit angeben.

/h:hostname
Gibt den TCP/IP-Hostnamen an, der von FCM für die interne Kommunikation verwendet wird, wenn der Hostname nicht der lokale Hostname ist. Dieser Parameter ist erforderlich, wenn Sie den Datenbankpartitionsserver auf einem fernen Computer hinzufügen möchten.

/i:instanzname
Gibt den Instanznamen an. Der Standardwert ist die aktuelle Instanz.

/m:computername
Der Computername der Windows-Workstation, auf der sich die Datenbankpartition befindet (der Standardname ist der Computername des lokalen Computers).

/o:instanzeignercomputer
Der Name des Computers, der Instanzeigner ist (der Standardname ist der Name des lokalen Computers). Dieser Parameter ist erforderlich, wenn der Befehl **db2ncrt** auf einem Computer aufgerufen wird, der nicht der Instanzeignercomputer ist.

Wenn Sie z. B. der Instanz TESTMPP einen neuen Datenbankpartitionsserver auf dem Instanzeignercomputer MYMACHIN hinzufügen möchten (um mehrere logische Datenbankpartitionen auszuführen) und diese neue Datenbankpartition als Datenbankpartition 2 mit dem logischen Port 1 definieren möchten, müssen Sie Folgendes eingeben:

```
db2ncrt /n:2 /p:1 /u:meine_id,mein_kwort /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

Ändern von Datenbankpartitionen (Windows)

Unter Windows können Datenbankpartitionen mit dem Befehl **db2nchg** geändert werden.

Informationen zu diesem Vorgang

- Versetzen der Datenbankpartition von einem Computer auf einen anderen
- Ändern des TCP/IP-Hostnamens des Computers

  Wenn Sie mit mehreren Netzadaptern arbeiten wollen, müssen Sie mit diesem Befehl die TCP/IP-Adresse für das Feld 'netzname' in der Datei `db2nodes.cfg` angeben.
- Verwenden einer anderen logischen Portnummer
- Verwenden eines anderen Namens für den Datenbankpartitionsserver

Der Befehl hat den folgenden erforderlichen Parameter:

```
db2nchg /n:knotennummer
```

Der Parameter **/n:** gibt die Nummer des Datenbankpartitionsservers an, die geändert werden soll. Dieser Parameter ist obligatorisch.

Der Befehl unterstützt die folgenden optionalen Parameter:

/i:*instanzname*
: Gibt die Instanz an, in der sich der aktuelle Datenbankpartitionsserver befindet. Wird dieser Parameter nicht angegeben, wird standardmäßig die aktuelle Instanz verwendet.

/u:*benutzername*,*kennwort*
: Ändert den Namen des Anmeldekontos und das Kennwort für den DB2-Datenbankservice. Wird dieser Parameter nicht angegeben, bleiben das Anmeldekonto und das Kennwort unverändert.

/p:*logischer_port*
: Ändert den logischen Port für den Datenbankpartitionsserver. Dieser Parameter muss angegeben werden, wenn Sie den Datenbankpartitionsserver auf einen anderen Computer versetzen wollen. Wird dieser Parameter nicht angegeben, bleibt die logische Portnummer unverändert.

/h:*hostname*
: Ändert den TCP/IP-Hostnamen, der von FCM für die interne Kommunikation verwendet wird. Wird dieser Parameter nicht angegeben, bleibt der Hostname unverändert.

/m:*computername*
: Versetzt den Datenbankpartitionsserver auf einen anderen Computer. Der Datenbankpartitionsserver kann nur dann versetzt werden, wenn die Instanz keine bereits definierten Datenbanken enthält.

/g:*netzname*
: Ändert den Netznamen des Datenbankpartitionsservers.

  Verwenden Sie diesen Parameter, wenn auf einem Computer mehrere IP-Adressen definiert sind und wenn Sie eine spezifische IP-Adresse für den Datenbankpartitionsserver verwenden wollen. Sie können für den Parameter *netzname* den Netznamen oder die IP-Adresse verwenden.

Wenn zum Beispiel der logische Port für Datenbankpartition 2 geändert werden soll, der an der Instanz TESTMPP beteiligt ist, um den logischen Port 3 zu verwenden, müssen Sie folgenden Befehl eingeben:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

Der DB2-Datenbankmanager ermöglicht Ihnen den Zugriff auf Registrierdatenbankvariablen des DB2-Datenbanksystems auf der Instanzebene auf einem fernen

Computer. Gegenwärtig werden Registrierdatenbankvariablen des DB2-Datenbanksystems auf drei verschiedenen Ebenen gespeichert: Computerebene (globale Ebene), Instanzebene und Datenbankpartitionsebene. Die auf Instanzebene (einschließlich der auf Datenbankpartitionsebene) gespeicherten Registrierdatenbankvariablen können mit **DB2REMOTEPREG** an einen anderen Computer umgeleitet werden. Wenn **DB2REMOTEPREG** definiert ist, greift der DB2-Datenbankmanager auf die Registrierdatenbankvariablen des DB2-Datenbanksystems auf dem Computer zu, der in **DB2REMOTEPREG** angegeben ist. Der Befehl **db2set** sähe folgendermaßen aus:

```
db2set DB2REMOTEPREG=ferne_workstation
```

Dabei ist *ferne_workstation* der Name der fernen Workstation.

Anmerkung:

- Bei der Definition dieser Option ist Vorsicht geboten, da alle DB2-Datenbankinstanzprofile und Instanzlisten unter dem angegebenen fernen Computernamen lokalisiert werden.
- Wenn in Ihrer Umgebung Benutzer aus Domänen vorhanden sind, stellen Sie sicher, dass das Anmeldekonto, das dem DB2-Instanzservice zugeordnet ist, ein Domänenkonto ist. Dadurch wird gewährleistet, dass die DB2-Instanz die richtigen Zugriffsrechte zur Aufzählung von Gruppen auf der Domänenebene besitzt.

Diese Einrichtung kann in Kombination mit der Einstellung von **DBINSTPROF** dazu verwendet werden, ein fernes LAN-Laufwerk auf demselben Computer anzugeben, der auch die Registrierdatenbank enthält.

Hinzufügen von Containern zu SMS-Tabellenbereichen in Datenbankpartitionen

Sie können einen Container nur einem SMS-Tabellenbereich in einer Datenbankpartition hinzufügen, der zurzeit keine Container besitzt.

Vorgehensweise

Geben Sie in der Befehlszeile Folgendes ein, um einem SMS-Tabellenbereich einen Container hinzuzufügen:

```
ALTER TABLESPACE name
   ADD ('pfad')
   ON DBPARTITIONNUM (datenbankpartitionsnummer)
```

Die durch die Nummer angegebene Datenbankpartition und jede Partition in dem Bereich der Datenbankpartitionen muss in der Datenbankpartitionsgruppe vorhanden sein, in der der Tabellenbereich definiert ist. Eine Datenbankpartitionsnummer kann möglicherweise nur explizit oder innerhalb eines Bereichs in genau einer Klausel ON DBPARTITIONNUM für die Anweisung angegeben werden.

Beispiel

Das folgende Beispiel zeigt, wie ein neuer Container der Datenbankpartitionsnummer 3 der Datenbankpartitionsgruppe hinzugefügt wird, die vom Tabellenbereich „plans” auf einem UNIX-Betriebssystem verwendet wird:

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON DBPARTITIONNUM (3)
```

Löschen einer Datenbankpartition aus einer Instanz (Windows)

Unter Windows können Sie mit dem Befehl **db2ndrop** einen Datenbankpartitionsserver aus einer Instanz löschen, in der keine Datenbanken definiert sind. Wenn Sie einen Datenbankpartitionsserver löschen, kann dessen Datenbankpartitionsnummer für einen neuen Datenbankpartitionsserver verwendet werden.

Informationen zu diesem Vorgang

Gehen Sie beim Löschen von Datenbankpartitionsservern aus einer Instanz mit besonderer Sorgfalt vor. Wenn Sie den Instanzeigner-Datenbankpartitionsserver (0) aus der Instanz entfernen, wird die Instanz dadurch unbrauchbar. Wenn Sie die Instanz löschen wollen, verwenden Sie den Befehl **db2idrop**.

Anmerkung: Wenn die Instanz Datenbanken enthält, darf der Befehl **db2ndrop** nicht verwendet werden. Verwenden Sie in diesem Fall stattdessen den Befehl **STOP DBM DROP DBPARTITIONNUM**. Dadurch wird sichergestellt, dass die Datenbank in der Datenbankpartition korrekt gelöscht wird. **Editieren Sie AUF KEINEN FALL** die Datei `db2nodes.cfg`! Dies kann zu Inkonsistenzen in der Umgebung mit partitionierten Datenbanken führen.

Wenn Sie eine Datenbankpartition löschen möchten, der durch einen Computer mit mehreren logischen Datenbankpartitionen der logische Port 0 zugeordnet ist, müssen Sie alle anderen Datenbankpartitionen löschen, die den anderen logischen Ports zugeordnet sind, bevor Sie die Datenbankpartition mit dem logischen Port 0 löschen können. Jeder Datenbankpartitionsserver muss über eine Datenbankpartition verfügen, der der logische Port 0 zugeordnet ist.

Der Befehl hat folgende Parameter:

```
db2ndrop /n:dbpartitionsnummer /i:instanzname
```

/n:*dbpartitionsnummer*
: Die eindeutige Datenbankpartitionsnummer (*dbpartitionsnummer*)zum Identifizieren des Datenbankpartitionsservers. Dieser Parameter ist erforderlich. Die Nummer kann ein Wert aus dem Bereich von null (`0`) bis `999` in aufsteigender Reihenfolge sein. Beachten Sie, dass die Datenbankpartition null (`0`) der Instanzeignercomputer ist.

/i:*instanzname*
: Der Instanzname (*instanzname*). Dies ist ein optionaler Parameter. Wird dieser Parameter nicht angegeben, wird standardmäßig die (mit der Registrierdatenbankvariablen **DB2INSTANCE** definierte) aktuelle Instanz verwendet.

Szenario: Umverteilen von Daten in neue Datenbankpartitionen

Dieses Szenario zeigt, wie neue Datenbankpartitionen einer Datenbank hinzugefügt und die Daten auf die neuen Datenbankpartitionen umverteilt werden. Die Veranschaulichung des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** bildet einen Teil der Erklärung, wie Daten auf unterschiedliche Tabellengruppen innerhalb einer Datenbankpartitionsgruppe umverteilt werden.

Informationen zu diesem Vorgang

Szenario:
: Eine Datenbankpartitionsgruppe DBPG_1 hat zwei Datenbankpartitionen, die als (0, 1) angegeben werden, und eine Datenbankpartitionsgruppendefinition (0, 1).

Die folgenden Tabellenbereiche sind in der Datenbankpartitionsgruppe DBPG_1 definiert:

- Tabellenbereich TS1: Dieser Tabellenbereich enthält die beiden Tabellen T1 und T2.
- Tabellenbereich TS2: Dieser Tabellenbereich enthält die drei definierten Tabellen T3, T4 und T5.

Ab Version 9.7 können Sie Datenbankpartitionen hinzufügen, während die Datenbank aktiv ist und Anwendungen mit ihr verbunden sind. Die Operation kann in diesem Szenario offline durchgeführt werden, indem der Standardwert der Registry-Variablen **DB2_FORCE_OFFLINE_ADD_PARTITION** in TRUE geändert wird.

Vorgehensweise

Gehen Sie wie folgt vor, um Daten auf die Datenbankpartitionen in DBPG_1 umzuverteilen:

1. Ermitteln Sie die Objekte, die inaktiviert oder entfernt werden müssen, bevor die Umverteilung erfolgt.
   a. Replizierte MQTs: Dieser Typ von MQT (Materialized Query Table) wird als Teil der Umverteilungsoperation nicht unterstützt. Solche MQTs müssen vor der Ausführung der Umverteilung gelöscht und anschließend erneut erstellt werden.

   ```
   SELECT tabschema, tabname
     FROM syscat.tables
     WHERE partition_mode = 'R'
   ```

   b. Ereignismonitore mit der Klausel WRITE TO TABLE (WTT-Ereignismonitore): Sie müssen alle automatisch aktivierten WTT-Ereignismonitore inaktivieren, die eine Tabelle beinhalten, die sich in der Datenbankpartitionsgruppe, die umverteilt werden soll, befindet.

   ```
   SELECT distinct evmonname FROM syscat.eventtables E
     JOIN syscat.tables T on T.tabname = E.tabname AND T.tabschema = E.tabschema
     JOIN syscat.tablespaces S on S.tbspace = T.tbspace
       AND S.ngname = 'DBPG_1'
   ```

   c. EXPLAIN-Tabellen: Es wird empfohlen, die EXPLAIN-Tabellen in nur einer Datenbankpartitionsgruppe zu erstellen. Wenn sie jedoch in einer Datenbankpartitionsgruppe definiert werden, die eine Umverteilung erfordert, und die bisher generierten Daten nicht beibehalten werden müssen, sollten Sie sie löschen. Die EXPLAIN-Tabellen können neu definiert werden, sobald die Umverteilung abgeschlossen ist.
   d. Zugriffsmodus und Status von Tabellen: Stellen Sie sicher, dass sich alle Tabellen in den umzuverteilenden Datenbankpartitionsgruppen im Vollzugriffsmodus und in normalen Tabellenstatus befinden.

   ```
   SELECT DISTINCT TRIM(T.OWNER) || \'.\' || TRIM(T.TABNAME)
     AS NAME, T.ACCESS_MODE, A.LOAD_STATUS
    FROM SYSCAT.TABLES T, SYSCAT.DBPARTITIONGROUPS
     N, SYSIBMADM.ADMINTABINFO A
    WHERE T.PMAP_ID = N.PMAP_ID
     AND A.TABSCHEMA = T.OWNER
     AND A.TABNAME = T.TABNAME
     AND N.DBPGNAME = 'DBPG_1'
     AND (T.ACCESS_MODE <> 'F' OR A.LOAD_STATUS IS NOT NULL)
   ```

   e. Statistikprofile: Wenn für die Tabelle ein Statistikprofil definiert wurde, können Tabellenstatistiken im Rahmen des Umverteilungsprozesses aktualisiert werden. Wenn Sie das Dienstprogramm für die Umverteilung zur Aktualisierung der Statistiken einer Tabelle verwenden, sinkt das benötigte

E/A-Volumen, weil alle Daten für die Umverteilung durchsucht werden und kein zusätzliches Durchsuchen der Daten für das Dienstprogramm **RUNSTATS** erforderlich ist.

```
RUNSTATS on table schema.tabelle USE PROFILE runstats-profil SET PROFILE ONLY
```

2. Überprüfen Sie die Datenbankkonfiguration. Der Wert des Parameters **util_heap_sz** ist für die Verarbeitung der Datenversetzung zwischen Datenbankpartitionen von kritischer Bedeutung. Ordnen Sie durch den Parameter **util_heap_sz** eine möglichst hohe Speicherkapazität für die Dauer der Umverteilung zu. Wenn im Rahmen der Umverteilung ein Indexrebuild erfolgt, ist genügend Sortierspeicher (Parameter **sortheap**) erforderlich. Erhöhen Sie die Werte der Parameter **util_heap_sz** und **sortheap** nach Bedarf, um die Umverteilungsleistung zu verbessern.
3. Rufen Sie die Datenbankkonfigurationseinstellungen ab, die für die neuen Datenbankpartitionen verwendet werden sollen. Wenn Datenbankpartitionen hinzugefügt werden, wird eine Standarddatenbankkonfiguration verwendet. Daher ist es wichtig, die Datenbankkonfiguration in den neuen Datenbankpartitionen zu aktualisieren, bevor der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** ausgeführt wird. Durch diese Reihenfolge der Ereignisse wird eine gleichmäßige Verteilung der Konfiguration sichergestellt.

```
SELECT name,
  CASE WHEN deferred_value_flags = 'AUTOMATIC'
    THEN deferred_value_flags
    ELSE substr(deferred_value,1,20)
                          END
  AS deferred_value
  FROM sysibmadm.dbcfg
  WHERE dbpartitionnum = vorhandener_knoten
    AND deferred_value != ''
    AND name NOT IN ('hadr_local_host','hadr_local_svc','hadr_peer_window',
                'hadr_remote_host','hadr_remote_inst','hadr_remote_svc',
                'hadr_syncmode','hadr_timeout','backup_pending','codepage',
                'codeset','collate_info','country','database_consistent',
                'database_level','hadr_db_role','log_retain_status',
                'loghead','logpath','multipage_alloc','numsegs','pagesize',
                'release','restore_pending','restrict_access',
                'rollfwd_pending','territory','user_exit_status',
                'number_compat','varchar2_compat','database_memory')
```

4. Führen Sie ein Backup der Datenbank (oder der Tabellenbereiche in der relevanten Datenbankpartitionsgruppe) aus, bevor Sie den Umverteilungsprozess starten. Durch diese Aktion wird ein aktueller Wiederherstellungspunkt sichergestellt.
5. Fügen Sie der Datenbank drei neue Datenbankpartitionen hinzu. Setzen Sie die folgenden Befehle ab:

```
START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
 PORT PORT3 WITHOUT TABLESPACES;

START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
 PORT PORT4 WITHOUT TABLESPACES;

START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
 PORT PORT5 WITHOUT TABLESPACES;
```

   Wenn der Parameter **DB2_FORCE_OFFLINE_ADD_PARTITION** auf TRUE gesetzt ist, werden neue Datenbankpartitionen erst dann für die Instanz sichtbar, nachdem diese heruntergefahren und neu gestartet wurde. Beispiel:

```
STOP DBM;START DBM;
```

6. Definieren Sie Tabellenbereichscontainer für temporäre Systemtabellen in den neu definierten Datenbankpartitionen.

```
ALTER TABLESPACE tablespace_name
  ADD container_information
  ON dbpartitionnums (3 to 5)
```

7. Fügen Sie die neuen Datenbankpartitionen der Datenbankpartitionsgruppen hinzu. Der folgende Befehl ändert die Definition der Datenbankpartitionsgruppe DBPG_1 von (0, 1) in (0, 1, 3, 4, 5):

   ```
   ALTER DATABASE PARTITION GROUP DBPG_1
     ADD dbpartitionnums (3 to 5)
     WITHOUT TABLESPACES
   ```

8. Definieren Sie Tabellenbereichscontainer für permanente Daten in den neu definierten Datenbankpartitionen.

   ```
   ALTER TABLESPACE tablespace_name
     ADD container_information
     ON dbpartitionnums (3 to 5)
   ```

9. Wenden Sie die Datenbankkonfigurationseinstellungen auf die neuen Datenbankpartitionen an (oder führen Sie einen einzelnen Befehl **UPDATE DB CFG** für alle Datenbankpartitionen aus).

10. Erfassen Sie die Definitionen aller replizierten MQTs, die in den Datenbankpartitionsgruppen vorhanden sind, die umverteilt werden sollen, und löschen Sie diese.

    ```
    db2look -d DBPG1 -e -z
      schema -t replizierte_mqt-namen
      -o repMQTs.clp
    ```

11. Inaktivieren Sie alle Ereignismonitore mit der Klausel WRITE TO TABLE, die in den Datenbankpartitionsgruppen, die umverteilt werden sollen, vorhanden sind.

    ```
    SET EVENT MONITOR monitorname STATE 0
    ```

12. Führen Sie das Dienstprogramm für die Umverteilung aus, um Daten gleichmäßig auf alle Datenbankpartitionen umzuverteilen.

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
     UNIFORM STOP AT 2006-03-10-07.00.00.000000;
    ```

    In diesem Szenario wird angenommen, dass der Befehl für die Tabellen T1, T2 und T3 erfolgreich ausgeführt und dann infolge der Angabe der Option **STOP AT** gestoppt wurde.

    Wenn die Datenumverteilung für die Datenbankpartitionsgruppe abgebrochen und die ausgeführten Änderungen an den Tabellen T1, T2 und T3 rückgängig gemacht werden sollen, kann der folgende Befehl ausgeführt werden:

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
    NOT ROLLFORWARD RECOVERABLE ABORT;
    ```

    Sie können die Datenumverteilung abbrechen, wenn ein Fehler oder eine Unterbrechung aufgetreten ist und Sie die Umverteilungsoperation nicht fortsetzen möchten. Für dieses Szenario soll angenommen werden, dass dieser Befehl erfolgreich ausgeführt und die Tabellen T1 und T2 in ihren ursprünglichen Status zurückversetzt wurden.

    Zur Umverteilung der Tabellen T5 und T4 mit nur 5000 4-KB-Seiten als Datenpuffer (**DATA BUFFER**) kann der folgende Befehl ausgeführt werden:

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
     UNIFORM TABLE (T5, T4) ONLY DATA BUFFER 5000;
    ```

    Wenn der Befehl erfolgreich ausgeführt wurde, wurden die Daten in den Tabellen T4 und T5 erfolgreich umverteilt.

    Mit dem folgenden Befehl wird die Umverteilung der Daten in den Tabellen T1, T2 und T3 in einer angegebenen Reihenfolge abgeschlossen:

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
     CONTINUE TABLE (T1) FIRST;
    ```

Die Angabe **TABLE (T1) FIRST** veranlasst den Datenbankmanager, die Tabelle T1 zuerst zu verarbeiten, sodass sie vor den anderen Tabellen wieder online (schreibgeschützt) verfügbar wird. Alle anderen Tabellen werden in einer vom Datenbankmanager festgelegten Reihenfolge verarbeitet.

Anmerkung:

- Der Parameter **ADD DBPARTITIONNUM** kann im Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** als Alternative zum Ausführen der Anweisungen ALTER DATABASE PARTITION GROUP und ALTER TABLESPACE in den Schritten 7 auf Seite 187 und 8 auf Seite 187 angegeben werden. Wenn eine Datenbankpartition mit diesem Befehlsparameter hinzugefügt wird, basieren die Container für Tabellenbereiche auf den Containern des entsprechenden Tabellenbereichs in der vorhandenen Partition mit der niedrigsten Nummer in der Datenbankpartitionsgruppe.
- Der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** in diesem Beispiel kann nicht durch eine aktualisierende Recovery wiederholt werden.
- Nach Abschluss des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** befinden sich alle Tabellenbereiche, auf die zugegriffen wurde, im Status 'Backup anstehend'. Für solche Tabellenbereiche muss ein Backup durchgeführt werden, bevor auf die Tabellen, die in ihnen enthalten sind, für Schreiboperationen zugegriffen werden kann.

Weitere Einzelheiten enthalten die Informationen zum „Befehl REDISTRIBUTE DATABASE PARTITION GROUP".

Sie sollten auch eine Tabellenliste als Eingabe für den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** angeben, um die Reihenfolge, in der die Tabellen verarbeitet werden, durchzusetzen. Das Dienstprogramm für die Umverteilung versetzt die (komprimierten und zusammengefassten) Daten. Optional werden Indizes erneut erstellt und Statistiken aktualisiert, wenn Statistikprofile definiert sind. Daher kann anstelle des vorherigen Befehls das folgende Script ausgeführt werden:

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
  NOT ROLLFORWARD RECOVERABLE uniform
  TABLE (t1, t2,...) FIRST;
```

Absetzen von Befehlen in Umgebungen mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken kann es wünschenswert sein, Befehle absetzen zu können, die auf Computern in der Instanz oder auf Datenbankpartitionsservern ausgeführt werden. Sie haben diese Möglichkeit mithilfe des Befehls **rah** oder des Befehls **db2_all**. Der Befehl **rah** ermöglicht Ihnen das Absetzen von Befehlen, die Sie auf Computern in der Instanz ausführen wollen.

Wenn Sie die Befehle auf Datenbankpartitionsservern in der Instanz ausführen wollen, verwenden Sie den Befehl **db2_all**. Dieser Abschnitt enthält eine Übersicht über diese Befehle. Die folgenden Informationen beziehen sich ausschließlich auf Umgebungen mit partitionierten Datenbanken.

Unter Windows müssen Sie mit einem Benutzerkonto angemeldet sein, das zur Administratorgruppe gehört, um die Befehle **rah** oder **db2_all** ausführen zu können.

Auf Linux- und UNIX-Betriebssystemen kann Ihre Anmeldeshell eine Korn-Shell oder eine beliebige andere Shell sein. Allerdings gibt es Unterschiede in der Art, wie verschiedene Shells Befehle behandeln, die Sonderzeichen enthalten.

Darüber hinaus verwendet der Befehl **rah** auf Linux- und UNIX-Betriebssystemen das Programm für die ferne Shell, das durch die Registrierdatenbankvariable **DB2RSHCMD** angegeben wird. Sie können zwischen den beiden Programmen für die ferne Shell wählen: 'ssh' (für zusätzliche Sicherheit) oder 'rsh' (bzw. 'remsh' für HP-UX). Wenn **DB2RSHCMD** nicht definiert ist, wird 'rsh' (bzw. 'remsh' für HP-UX) verwendet. Das Programm 'ssh' für die ferne Shell wird verwendet, um die Übertragung von unverschlüsselten Kennwörtern in UNIX-Betriebssystemumgebungen zu verhindern.

Wenn ein Befehl nur auf einem Datenbankpartitionsserver ausgeführt wird und Sie ihn auf allen von ihnen ausführen wollen, verwenden Sie dazu den Befehl **db2_all**. Eine Ausnahme bildet der Befehl **db2trc**, der auf allen logischen Datenbankpartitionsservern auf einem Computer ausgeführt wird. Wenn Sie den Befehl **db2trc** auf allen logischen Datenbankpartitionsservern aller Computer ausführen möchten, verwenden Sie dazu den Befehl **rah**.

Anmerkung: Der Befehl **db2_all** unterstützt keine Befehle, die eine interaktive Benutzereingabe erfordern.

Übersicht über die Befehle rah und db2_all

Die Befehle können sequenziell auf einem Datenbankpartitionsserver nach dem anderen oder parallel ausgeführt werden.

Wenn Sie die Befehle auf Linux- und UNIX-Betriebssystemen parallel ausführen, können Sie wählen, ob die Ausgabe an einen Puffer gesendet werden soll, in dem sie zur späteren Anzeige gesammelt wird (Standardmodus), oder ob die Ausgabe auf dem Computer angezeigt werden soll, auf dem der Befehl abgesetzt wird. Unter Windows wird die Ausgabe bei paralleler Ausführung der Befehle auf dem Computer angezeigt, auf dem der Befehl abgesetzt wurde.

Zur Verwendung des Befehls **rah** geben Sie Folgendes ein:

```
rah befehl
```

Zur Verwendung des Befehls **db2_all** geben Sie Folgendes ein:

```
db2_all befehl
```

Hilfe zur Syntax von **rah** erhalten Sie durch folgende Eingabe:

```
rah "?"
```

Als Befehl kann fast alles angegeben werden, was in eine interaktive Eingabeaufforderung eingegeben werden kann. Dazu gehören zum Beispiel auch mehrere Befehle, die nacheinander ausgeführt werden. Auf Linux- und UNIX-Betriebssystemen werden mehrere Befehle durch ein Semikolon (;) voneinander getrennt. Unter Windows werden mehrere Befehle durch ein Et-Zeichen (&) voneinander getrennt. Das Trennzeichen darf jedoch nicht nach dem letzten Befehl verwendet werden.

Das folgende Beispiel zeigt die Verwendung des Befehls **db2_all**, um die Datenbankkonfiguration in allen Datenbankpartitionen zu ändern, die in der Datei für

die Datenbankpartitionskonfiguration angegeben sind. Da das Semikolon (;) in die doppelten Anführungszeichen mit eingeschlossen ist, wird die Anforderung parallel ausgeführt.

```
db2_all ";DB2 UPDATE DB CFG FOR sample USING LOGFILSIZ 100"
```

Anmerkung: Der Befehl **db2_all** unterstützt keine Befehle, die eine interaktive Benutzereingabe erfordern.

Angeben der Befehle 'rah' und 'db2_all'

Sie können den Befehl **rah** über die Befehlszeile als Parameter oder als Antwort auf die Eingabeaufforderung angeben, wenn Sie keine Parameter angeben.

Verwenden Sie die Angabe über die Eingabeaufforderung, wenn der Befehl die folgenden Sonderzeichen enthält:

```
| & ; < > ( )  { } [ ] nicht ersetztes $
```

Wenn Sie den Befehl als Parameter in der Befehlszeile angeben, müssen Sie ihn in doppelte Anführungszeichen setzen, wenn er eines oder mehrere der aufgeführten Sonderzeichen enthält.

Anmerkung: Auf Linux- und UNIX-Betriebssystemen wird der Befehl Ihrem Befehlsprotokoll so hinzugefügt, als hätten Sie ihn in die Eingabeaufforderung eingegeben.

Alle Sonderzeichen im Befehl können normal eingegeben werden (d. h. ohne Anführungszeichen, außer dem umgekehrten Schrägstrich \). Wenn Sie einen umgekehrten Schrägstrich (\) in Ihrem Befehl benötigen, müssen Sie zwei umgekehrte Schrägstriche eingeben (\\).

Anmerkung: Auf Linux- und UNIX-Betriebssystemen können alle Sonderzeichen im Befehl normal eingegeben werden (d. h. ohne Anführungszeichen, außer den Zeichen ", \, nicht ersetztes $ und einfaches Anführungszeichen (')), wenn Sie keine Korn-Shell verwenden. Wenn Sie eines dieser Zeichen in Ihrem Befehl benötigen, müssen Sie dem Zeichen drei umgekehrte Schrägstriche (\\\) voranstellen. Wenn Sie beispielsweise einen umgekehrten Schrägstrich (\) in Ihrem Befehl benötigen, müssen Sie vier umgekehrte Schrägstriche (\\\\) eingeben.

Wenn Sie ein Anführungszeichen (") in Ihrem Befehl benötigen, müssen Sie ihm drei umgekehrte Schrägstriche voranstellen (Beispiel: \\\").

Anmerkung:

1. Auf Linux- und UNIX-Betriebssystemen dürfen Sie kein einfaches Anführungszeichen (') im Befehl angeben, sofern Ihre Befehlsshell nicht irgendeine Möglichkeit zur Eingabe eines einfachen Anführungszeichens innerhalb einer in einfache Anführungszeichen gesetzten Zeichenfolge bereitstellt.
2. Unter Windows dürfen Sie kein einfaches Anführungszeichen (') im Befehl angeben, sofern Ihr Befehlsfenster nicht irgendeine Möglichkeit zur Eingabe eines einfachen Anführungszeichens innerhalb einer in einfache Anführungszeichen gesetzten Zeichenfolge bereitstellt.

Wenn Sie ein beliebiges Shell-Script der Korn-Shell ausführen, das Logik enthält, die von 'stdin' im Hintergrund gelesen wird, leiten Sie 'stdin' ausdrücklich zu einer Quelle um, an der der Prozess den Lesevorgang ohne Unterbrechungen am Terminal durchführen kann (Nachricht SIGTTIN). Zum Umleiten von 'stdin' können Sie ein Script im folgenden Format ausführen:

```
shell_script </dev/null &
```

wenn keine Eingabe bereitgestellt werden muss.

Geben Sie in ähnlicher Weise stets </dev/null an, wenn **db2_all** im Hintergrund ausgeführt wird. Beispiel:

```
db2_all ";diesen_befehl_ausführen" </dev/null &
```

Auf diese Weise können Sie 'stdin' umleiten und Unterbrechungen am Terminal vermeiden.

Eine Alternative zu dieser Methode, die Sie anwenden können, wenn Sie die Ausgabe von einem fernen Befehl nicht stört, ist die Verwendung der Option „daemonize" im Präfix für **db2_all**:

```
db2_all ";daemonize_diesen_befehl" &
```

Paralleles Ausführen von Befehlen (Linux, UNIX)

Standardmäßig wird der Befehl sequenziell auf jedem Computer ausgeführt. Jedoch können Sie mithilfe von rshell-Hintergrundprozessen die Befehle parallel ausführen, indem Sie den Befehl mit bestimmten Präfixsequenzen versehen. Wenn rshell im Hintergrund ausgeführt wird, schreibt jeder Befehl die Ausgabe in eine Pufferdatei auf dem jeweiligen fernen Computer.

Anmerkung: Die Informationen dieses Abschnitts beziehen sich ausschließlich auf Linux- und UNIX-Betriebssysteme.

Dieser Prozess ruft die Ausgabe in zwei Teilen ab:

1. Nach Beendigung des fernen Befehls
2. Nach Beendigung der rshell, was später geschehen kann, wenn einige Prozesse andauern

Der Name der Pufferdatei ist standardmäßig /tmp/$USER/rahout. Er kann aber durch die Umgebungsvariable **RAHBUFDIR** oder $**RAHBUFNAME** definiert werden.

Wenn Sie angeben, dass die Befehle gleichzeitig ausgeführt werden sollen, schaltet dieses Script standardmäßig einen weiteren Befehl vor den Befehl, der an alle Hosts gesendet wird, um zu überprüfen, ob die Umgebungsvariablen $**RAHBUFDIR** und $**RAHBUFNAME** für die Pufferdatei verwendbar sind. Das in $**RAHBUFDIR** definierte Verzeichnis wird erstellt. Sie können dies unterdrücken, indem Sie eine Umgebungsvariable RAHCHECKBUF=no exportieren. Dies dient der Zeitersparnis, wenn Sie bereits wissen, dass das Verzeichnis existiert und verwendbar ist.

Bevor Sie **rah** verwenden, um einen Befehl gleichzeitig auf mehreren Computern auszuführen, sollten Sie folgende Schritte ausführen:

- Stellen Sie sicher, dass ein Verzeichnis /tmp/$USER für Ihre Benutzer-ID auf jedem Computer vorhanden ist. Zur Erstellung eines Verzeichnisses, falls es noch nicht vorhanden ist, führen Sie folgenden Befehl aus:

  ```
  rah ")mkdir /tmp/$USER"
  ```

- Fügen Sie die folgende Zeile in Ihre Datei .kshrc (für Korn-Shellsyntax) oder .profile ein, und geben Sie sie auch in Ihre aktuelle Sitzung ein:

  ```
  export RAHCHECKBUF=no
  ```

- Stellen Sie sicher, dass jede Computer-ID, auf der Sie den fernen Befehl ausführen, in der zugehörigen Datei .rhosts über einen Eintrag für die ID verfügt, unter der der Befehl **rah** ausgeführt wird, und dass die ID, unter der der Befehl

rah ausgeführt wird, in der zugehörigen Datei `.rhosts` über einen Eintrag für die einzelnen Computer-IDs verfügt, auf denen Sie den fernen Befehl ausführen.

Erweiterung des Befehls 'rah' zur Verwendung von Baumstrukturlogik (AIX und Solaris)

Zur Leistungsverbesserung wurde 'rah' erweitert, um eine Baumstrukturlogik für große Systeme zu verwenden. Das heißt, 'rah' überprüft, wie viele Datenbankpartitionen in der Liste enthalten sind. Wenn diese Anzahl einen Schwellenwert überschreitet, erstellt 'rah' eine Teilmenge der Liste und sendet einen rekursiven Aufruf von sich selbst an diese Datenbankpartitionen.

In diesen Datenbankpartitionen wendet das rekursiv aufgerufene 'rah' dieselbe Logik an, bis die Liste so klein ist, dass mit der Standardlogik (nun die Logik "leaf-of-tree") der Befehl an alle Datenbankpartitionen in der Liste versendet werden kann. Der Schwellenwert kann mit der Umgebungsvariable **RAHTREETHRESH** angegeben werden, oder er nimmt standardmäßig den Wert 15 an.

Bei einem System mit mehreren logischen Datenbankpartitionen für eine physische Datenbankpartition sendet **db2_all** den rekursiven Aufruf bevorzugt an bestimmte physische Datenbankpartitionen, die dann 'rsh' an die anderen logischen Datenbankpartitionen auf derselben physischen Datenbankpartition senden, sodass auch die Übertragungen zwischen den physischen Datenbankpartitionen reduziert werden. (Dies gilt nur für **db2_all**, nicht für 'rah', da 'rah' immer nur an bestimmte physische Datenbankpartitionen sendet.)

Befehle 'rah' und 'db2_all'

Dieser Abschnitt enthält Beschreibungen der Befehle **rah** und **db2_all**.

Befehl **Beschreibung**

rah Führt den Befehl auf allen Computern aus.

db2_all
Führt einen Befehl auf allen Datenbankpartitionsservern aus, die Sie angeben. Der Befehl **db2_all** unterstützt keine Befehle, die eine interaktive Benutzereingabe erfordern.

db2_kill
Stoppt abrupt alle Prozesse, die auf mehreren Datenbankpartitionsservern ausgeführt werden, und bereinigt alle Ressourcen auf allen Datenbankpartitionsservern. Durch diesen Befehl werden Ihre Datenbanken inkonsistent. Führen Sie diesen Befehl *nicht* ohne entsprechende Anweisung durch den IBM Software Support oder entsprechend der Anweisung für die Recovery nach einem erhaltenen Trap aus.

db2_call_stack
Auf Linux- und UNIX-Betriebssystemen werden alle Prozesse, die auf allen Datenbankpartitionsservern ausgeführt werden, veranlasst, ein Traceback für Aufrufe in das Systemprotokoll zu schreiben.

Auf Linux- und UNIX-Betriebssystemen führen diese Befehle **rah** mit bestimmten impliziten Einstellungen wie zum Beispiel folgenden aus:

- Parallele Ausführung auf allen Computern
- Puffern der Befehlsausgabe in `/tmp/$USER/db2_kill` bzw. `/tmp/$USER/db2_call_stack`

Der Befehl **db2_call_stack** steht unter Windows *nicht* zur Verfügung. Verwenden Sie stattdessen den Befehl **db2pd -stack**.

Präfixsequenzen für die Befehle 'rah' und 'db2_all'

Eine Präfixsequenz besteht aus einem oder mehreren Sonderzeichen.

Eine oder mehrere Präfixsequenzen müssen unmittelbar vor den Zeichen des Befehls ohne zwischengeschaltete Leerzeichen eingegeben werden. Wenn Sie mehr als eine Sequenz angeben wollen, können Sie sie in beliebiger Reihenfolge eingeben. Zeichen innerhalb einer Mehrzeichensequenz müssen jedoch in der richtigen Reihenfolge eingegeben werden. Wenn Sie Präfixsequenzen eingeben, müssen Sie den gesamten Befehl, einschließlich der Präfixsequenzen, in doppelte Anführungszeichen setzen, wie in folgenden Beispielen gezeigt:

- Auf Linux- und UNIX-Betriebssystemen:

```
rah "};ps -F pid,ppid,etime,args -u $USER"  db2_all "};ps -F pid,ppid,etime,args
-u $USER"
```

- Auf Windows-Betriebssystemen:

```
rah "||db2 get db cfg for sample"  db2_all "||db2 get db cfg for sample"
```

Folgende Präfixsequenzen sind möglich:

Sequenz	Zweck
`|`	Führt die Befehle nacheinander im Hintergrund aus.
`|&`	Führt die Befehle nacheinander im Hintergrund aus und beendet den Befehl, wenn alle fernen Befehle ausgeführt sind, auch wenn einige Prozesse noch aktiv sind. Diese Aktion kann später stattfinden, wenn zum Beispiel untergeordnete Prozesse (auf Linux- und UNIX-Betriebssystemen) oder Hintergrundprozesse (auf Windows-Betriebssystemen) noch aktiv sind. In diesem Fall startet der Befehl einen getrennten Hintergrundprozess, um sämtliche fernen Ausgaben abzurufen, die nach Beendigung des Befehls generiert wurden, und schreibt diese wieder in den Ursprungscomputer. **Anmerkung:** Auf Linux- und UNIX-Betriebssystemen wird durch die Angabe von `&` die Leistung beeinträchtigt, da mehr **rsh**-Befehle erforderlich werden.
`||`	Führt die Befehle parallel im Hintergrund aus.
`||&`	Führt die Befehle parallel im Hintergrund aus und beendet den Befehl, wenn alle fernen Befehle ausgeführt sind, wie oben für die Sequenz `|&` beschrieben. **Anmerkung:** Auf Linux- und UNIX-Betriebssystemen wird durch die Angabe von `&` die Leistung beeinträchtigt, da mehr **rsh**-Befehle erforderlich werden.
`;`	Hat die gleiche Funktion wie `||&`. Dies ist eine kürzere alternative Form. **Anmerkung:** Auf Linux- und UNIX-Betriebssystemen wird durch die Angabe von `;` die Leistung im Vergleich zu `||` herabgesetzt, da mehr **rsh**-Befehle erforderlich sind.
`]`	Schaltet die Punktdateiausführung des Profils des Benutzers vor die Ausführung des Befehls. **Anmerkung:** Nur für Linux- und UNIX-Betriebssysteme verfügbar.
`}`	Schaltet die Punktdateiausführung der in $**RAHENV** definierten Datei (wahrscheinlich `.kshrc`) vor die Ausführung des Befehls.

Anmerkung: Nur für Linux- und UNIX-Betriebssysteme verfügbar.

]} Schaltet die Punktdateiausführung des Profils des Benutzers gefolgt von der Ausführung der in $**RAHENV** definierten Datei (wahrscheinlich `.kshrc`) vor die Ausführung des Befehls.

Anmerkung: Nur für Linux- und UNIX-Betriebssysteme verfügbar.

) Unterdrückt die Ausführung des Benutzerprofils und der in $**RAHENV** definierten Datei.

Anmerkung: Nur für Linux- und UNIX-Betriebssysteme verfügbar.

' Meldet den Befehlsaufruf zurück an den Computer.

< Sendet an alle Computer außer diesen.

<<-*nnn***<**

Sendet an alle außer Datenbankpartitionsserver *nnn* (d. h. alle Datenbankpartitionsserver in der Datei `db2nodes.cfg` außer dem mit der Datenbankpartitionsnummer *nnn*. Siehe den ersten Abschnitt nach der letzten Präfixsequenz in dieser Tabelle).

nnn ist die zugehörige ein-, zwei- oder dreistellige Datenbankpartitionsnummer für den Wert *nodenum* in der Datei `db2nodes.cfg`.

<<-nnn< ist nur auf **db2_all** anwendbar.

<<+*nnn***<**

Sendet nur an Datenbankpartitionsserver *nnn* (der Datenbankpartitionsserver in der Datei `db2nodes.cfg`, dessen Datenbankpartitionsnummer *nnn* lautet. Siehe erster Abschnitt nach der letzten Präfixsequenz in dieser Tabelle).

nnn ist die zugehörige ein-, zwei- oder dreistellige Datenbankpartitionsnummer für den Wert *nodenum* in der Datei `db2nodes.cfg`.

<<+nnn< ist nur auf **db2_all** anwendbar.

(Leerzeichen)
Führt den fernen Befehl im Hintergrund bei geschlossenen `stdin` (Standardeingabeeinheit), `stdout` (Standardausgabeeinheit) und `stderr` (Standardfehlereinheit) aus. Diese Option ist nur gültig, wenn der Befehl im Hintergrund ausgeführt wird, d. h. nur in einer Präfixsequenz, die auch \ oder ; enthält. Sie ermöglicht dem Befehl eine wesentlich frühere Beendigung (sobald der ferne Befehl eingeleitet wurde). Wenn Sie dieses Präfixzeichen in der **rah**-Befehlszeile angeben, setzen Sie den Befehl entweder in einfache Anführungszeichen oder setzen Sie den Befehl in doppelte Anführungszeichen und stellen dem Präfixzeichen einen umgekehrten Schrägstrich (\ ) voran. Beispiel:

```
rah '; mydaemon'
```

oder

```
rah ";\ mydaemon"
```

Bei der Ausführung als Hintergrundprozess wartet der **rah**-Befehl nie auf die Rückgabe irgendeiner Ausgabe.

> Ersetzt die Vorkommen von > durch den Computernamen.

" Ersetzt die Vorkommen von () durch den Computerindex und die Vorkommen von ## durch die Datenbankpartitionsnummer.

- Der Computerindex ist eine Nummer, die einem Computer innerhalb des Datenbanksystems zugeordnet ist. Wenn Sie nicht mit mehreren logischen Partitionen arbeiten, entspricht der Computerindex für einen Computer der Datenbankpartitionsnummer für diesen Computer in der Datei für die Datenbankpartitionskonfiguration. Der Computerindex für einen Computer in einer Umgebung mit mehreren logischen Datenbankpartitionen wird ermittelt, indem mehrere Einträge für die Computer, die mehrere logische Partitionen ausführen, nur einfach gezählt werden. Wenn zum Beispiel MACH1 zwei logische Partitionen ausführt und MACH2 ebenfalls zwei logische Partitionen ausführt, ist die Datenbankpartitionsnummer für MACH3 in der Datei für die Datenbankpartitionskonfiguration 5. Der Computerindex für MACH3 wäre jedoch 3.
  - Auf Windows-Betriebssystemen sollten Sie die Datei für die Datenbankpartitionskonfiguration nicht bearbeiten. Verwenden Sie den Befehl **db2nlist**, wenn Sie den Computerindex abrufen möchten.
- Wenn " angegeben wird, werden doppelte Vorkommen aus der Liste der Computer nicht eliminiert.

Hinweise

- Präfixsequenzen werden als Teil des Befehls behandelt. Wenn Sie Präfixsequenzen angeben, müssen Sie den gesamten Befehl, einschließlich der Präfixsequenzen, in doppelte Anführungszeichen setzen.

Steuern des Befehls 'rah'

In diesem Abschnitt werden die Umgebungsvariablen aufgeführt, mit denen der Befehl **rah** gesteuert werden kann.

Tabelle 13. Umgebungsvariablen zur Steuerung des Befehls **rah**

Name	Bedeutung	Standardwert
$**RAHBUFDIR** **Anmerkung:** Nur für Linux- und UNIX-Betriebssysteme verfügbar.	Verzeichnis für den Puffer	`/tmp/$USER`
$**RAHBUFNAME** **Anmerkung:** Nur für Linux- und UNIX-Betriebssysteme verfügbar.	Dateiname für den Puffer	rahout
$**RAHOSTFILE** (auf Linux- und UNIX-Betriebssystemen); **RAHOSTFILE** (auf Windows-Betriebssystemen)	Datei mit der Liste der Hosts	`db2nodes.cfg`
$**RAHOSTLIST** (auf Linux- und UNIX-Betriebssystemen); **RAHOSTLIST** (auf Windows-Betriebssystemen)	Liste der Hosts in Form einer Zeichenfolge	extrahiert aus $**RAHOSTFILE**

Tabelle 13. Umgebungsvariablen zur Steuerung des Befehls **rah** *(Forts.)*

Name	Bedeutung	Standardwert
$**RAHCHECKBUF** **Anmerkung:** Nur für Linux- und UNIX-Betriebssysteme verfügbar.	Wenn auf "no", Prüfungen übergehen	Nicht definiert
$**RAHSLEEPTIME** (auf Linux- und UNIX-Betriebssystemen); **RAHSLEEPTIME** (auf Windows-Betriebssystemen)	Zeit in Sekunden, die dieses Script auf die erste Ausgabe parallel ausgeführter Befehle wartet	86400 Sekunden für **db2_kill**, 200 Sekunden für alle anderen
$**RAHWAITTIME** (auf Linux- und UNIX-Betriebssystemen); **RAHWAITTIME** (auf Windows-Betriebssystemen)	Auf Windows-Betriebssystemen das Intervall in Sekunden zwischen aufeinander folgenden Prüfungen darauf, ob ferne Jobs noch aktiv sind. Auf Linux- und UNIX-Betriebssystemen das Intervall in Sekunden zwischen aufeinander folgenden Prüfungen darauf, ob ferne Jobs noch aktiv sind und rah: waiting for pid> ...-Nachrichten zurückgegeben werden. Bei allen Betriebssystemen ist eine positive ganze Zahl anzugeben. Stellen Sie dem Wert eine führende Null voran, um Nachrichten zu unterdrücken. Beispiel: export RAHWAITTIME=045. Es ist nicht notwendig, einen niedrigen Wert anzugeben, da der Befehl **rah** bei der Feststellung von Jobbeendigungen nicht auf diese Prüfungen angewiesen ist.	45 Sekunden
$**RAHENV** **Anmerkung:** Nur für Linux- und UNIX-Betriebssysteme verfügbar.	Gibt den Dateinamen an, der auszuführen ist, wenn $**RAHDOTFILES**=E oder K oder PE oder B ist.	$**ENV**
$**RAHUSER** (auf Linux- und UNIX-Betriebssystemen); **RAHUSER** (auf Windows-Betriebssystemen)	Auf Linux- und UNIX-Betriebssystemen die Benutzer-ID, unter der der ferne Befehl auszuführen ist. Auf Windows-Betriebssystemen das Anmeldekonto, das DB2 Remote Command Service zugeordnet ist.	$**USER**

Anmerkung: Auf Linux- und UNIX-Betriebssystemen wird der Wert von $**RAHENV** von dort, wo **rah** ausgeführt wird, verwendet, nicht der Wert (falls vorhanden), der von der fernen Shell definiert wurde.

Angeben der .-Dateien, die mit 'rah' ausgeführt werden (Linux und UNIX)

In diesem Abschnitt werden die Punktdateien (.-Dateien) beschrieben, die ausgeführt werden, wenn keine Präfixsequenz angegeben wird.

Anmerkung: Die Informationen in diesem Abschnitt beziehen sich ausschließlich auf Linux- und UNIX-Betriebssysteme.

P `.profile`

E Datei, die in $**RAHENV** definiert ist (wahrscheinlich `.kshrc`)

K Wie E

PE `.profile` gefolgt von der Datei, die in $**RAHENV** definiert ist (wahrscheinlich `.kshrc`)

B Wie PE

N Keine (weder noch)

Anmerkung: Wenn Ihre Anmeldeshell keine Korn-Shell ist, werden alle Punktdateien, die Sie zur Ausführung angeben, in einem Korn-Shellprozess ausgeführt und müssen daher der Korn-Shell-Syntax entsprechen. Wenn Ihre Anmeldeshell zum Beispiel eine C-Shell ist und Sie Ihre `.cshrc`-Umgebung für Befehle einrichten wollen, die von **rah** ausgeführt werden, müssen Sie eine Korn-Shell-Entsprechung der Datei *INSTHOME*`/.profile` für Ihre Datei `.cshrc` erstellen und in Ihrer Datei *INSTHOME*`/.cshrc` Folgendes angeben:

```
setenv RAHDOTFILES P
```

Oder Sie müssen eine Korn-Shell-Entsprechung der Datei *INSTHOME*`/.kshrc` für Ihre Datei `.cshrc` erstellen und in Ihrer Datei *INSTHOME*`/.cshrc` Folgendes angeben:

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

Außerdem darf Ihre Datei `.cshrc` die Ausgabe nicht in die Standardausgabe (stdout) schreiben, wenn (wie beim Aufruf durch **rsh**) kein TTY vorhanden ist. Dies können Sie sicherstellen, indem Sie alle Zeilen, die an 'stdout' schreiben, zum Beispiel folgendermaßen einschließen:

```
if { tty -s } then echo "executed .cshrc";
endif
```

Beheben von Fehlern mit 'rah' (Linux, UNIX)

Dieser Abschnitt enthält Vorschläge zur Behandlung einiger Fehler, die bei der Ausführung von **rah** möglicherweise auftreten.

Anmerkung: Die Informationen in diesem Abschnitt beziehen sich ausschließlich auf Linux- und UNIX-Betriebssysteme.

1. **rah** blockiert (oder benötigt sehr lange Zeit)

   Dieses Problem kann folgende Ursache haben:

   - **rah** hat festgestellt, dass Ausgaben gepuffert werden müssen und dass `RAHCHECKBUF=no` nicht exportiert wurde. Daher sendet **rah** vor der Ausführung Ihres Befehls einen Befehl an alle Computer, um die Existenz des Pufferverzeichnisses zu überprüfen und um es zu erstellen, falls es noch nicht vorhanden ist.
   - Mindestens ein Computer, an den Sie Ihren Befehl senden, antwortet nicht. Der Befehl **rsh** wird schließlich sein Zeitlimit überschreiten. Das Zeitlimit ist in der Regel jedoch recht lang (in der Regel ungefähr 60 Sekunden).

2. Sie haben Nachrichten wie die folgenden empfangen:
   - Anmeldung nicht korrekt (Login incorrect)
   - Berechtigung verweigert (Permission denied)

Entweder ist auf einem der Computer die ID, unter der **rah** ausgeführt wird, in der zugehörigen Datei `/etc/hosts` nicht korrekt definiert oder für die ID, unter der **rah** ausgeführt wird, ist einer der Computer in der zugehörigen Datei `.rhosts` nicht korrekt definiert. Wenn die Registrierdatenbankvariable **DB2RSHCMD** für die Verwendung von ssh konfiguriert wurde, sind möglicherweise die ssh-Clients und -Server auf dem jeweiligen Computer nicht korrekt konfiguriert.

Anmerkung: In bestimmten Fällen benötigen Sie möglicherweise einen höheren Grad an Sicherheit hinsichtlich der unverschlüsselten Übertragung von Kennwörtern zwischen Datenbankpartitionen. Dies ist von dem Programm für die ferne Shell abhängig, das Sie einsetzen. **rah** verwendet das Programm für die ferne Shell, das in der Registrierdatenbankvariablen **DB2RSHCMD** angegeben ist. Sie können zwischen den beiden Programmen für die ferne Shell wählen: ssh (für zusätzliche Sicherheit) oder rsh (bzw. remsh für HP-UX). Wenn diese Registrierdatenbankvariable nicht definiert ist, wird rsh (bzw. remsh für HP-UX) verwendet.

3. Obwohl bei der parallelen Ausführung von Befehlen mit fernen Hintergrundshells die Befehle ausgeführt und in der erwarteten abgelaufenen Zeit auf den Hosts abgeschlossen werden, benötigt **rah** viel Zeit, um dies zu erkennen und die Shelleingabeaufforderung anzuzeigen.

   Für die ID, die **rah** ausführt, ist einer der Computer nicht korrekt in der Datei `.rhosts` definiert, oder es wurden, falls die Registrierdatenbankvariable **DB2RSHCMD** für die Verwendung von 'ssh' konfiguriert wurde, die ssh-Clients und -Server auf den jeweiligen Computern möglicherweise nicht korrekt konfiguriert.

4. **rah** läuft bei Ausführung über die Shell-Befehlszeile einwandfrei. Aber wenn **rah** fern mithilfe von rsh, zum Beispiel mit folgendem Befehl ausgeführt wird:

   ```
   rsh somewher -l $USER db2_kill
   ```

   wird **rah** niemals beendet.

   Dies ist normal. Der Befehl **rah** startet Überwachungsprozesse im Hintergrund, die auch nach der Beendigung weiterhin aktiv sind. Diese Prozesse bleiben in der Regel bestehen, bis alle Prozesse, die zu dem Befehl gehören, den Sie ausgeführt haben, ihrerseits beendet sind. Im Fall von **db2_kill** bedeutet dies, dass sie bis zur Beendigung aller Datenbankmanager bestehen bleiben. Sie können die Überwachungsprozesse beenden, indem Sie den Prozess suchen, dessen Befehl **rahwaitfor** ist, und den Befehl kill *prozess-id*> ausführen. Geben Sie keine Signalnummer an. Verwenden Sie stattdessen den Standardwert (15).

5. Die Ausgabe von **rah** wird nicht korrekt angezeigt, oder **rah** meldet fälschlicherweise, dass $**RAHBUFNAME** nicht existiert, wenn mehrere Befehle von **rah** unter derselben Benutzer-ID $**RAHUSER** abgesetzt wurden.

   Dies kann eintreten, weil mehrere gleichzeitig ablaufende Ausführungen von **rah** versuchen, dieselbe Pufferdatei (z. B. $**RAHBUFDIR** oder $**RAHBUFNAME**) zu verwenden. Zur Vermeidung dieses Problems verwenden Sie für jeden gleichzeitig ablaufenden Befehl **rah** einen unterschiedlichen Namen für $**RAHBUFNAME**. Im Folgenden ist ein ksh-Beispiel aufgeführt:

   ```
   export RAHBUFNAME=rahout
   rah ";$befehl_1" &
   export RAHBUFNAME=rah2out
   rah ";$befehl_2" &
   ```

   Alternativ hierzu können Sie eine Methode verwenden, mit der die Shell automatisch einen eindeutigen Namen auswählt. Beispiel:

```
RAHBUFNAME=rahout.$$ db2_all "....."
```

Unabhängig von der verwendeten Methode müssen Sie sicherstellen, dass die Pufferdateien zu einem bestimmten Zeitpunkt bereinigt werden, wenn der Plattenspeicherplatz begrenzt ist. Der Befehl **rah** löscht keine Pufferdatei am Ende der Ausführung. Eine vorhandene Pufferdatei wird jedoch beim nächsten Mal gelöscht und wiederverwendet, wenn Sie dieselbe Pufferdatei erneut angeben.

6. Sie haben Folgendes eingegeben:

```
rah '"print from ()'
```

Daraufhin haben Sie die folgende Nachricht empfangen:

```
ksh: syntax error at line 1 : (' unexpected
(ksh: Syntaxfehler in Zeile 1 : (' unerwartet)
```

Voraussetzungen für die Ersetzung von () und ## sind:

- Es ist **db2_all** und nicht **rah** zu verwenden.
- Stellen Sie sicher, dass eine **RAHOSTFILE**-Datei verwendet wird. Dies geschieht entweder durch Exportieren der Umgebungsvariablen **RAHOSTFILE** oder durch die sich standardmäßig ergebene Verwendung der Datei `/sqllib/db2nodes.cfg`. Ohne diese Vorbedingungen lässt **rah** die Zeichenfolgen () und ## unverändert. Sie empfangen einen Fehler, weil der Befehl **print from ()** ungültig ist.

Zur Verbesserung der Leistung bei der parallelen Ausführung der Befehle sollten Sie `|` anstelle von `|&` und `||` anstelle von `||&` oder ; verwenden, es sei denn, die von & bereitgestellte Funktion wird unbedingt benötigt. Die Angabe von & erfordert eine größere Anzahl ferner Shellbefehle und führt daher zu einer Beeinträchtigung der Leistung.

Überwachen von rah-Prozessen (Linux, UNIX)

Während einige ferne Befehle immer noch aktiv sind oder die gepufferte Ausgabe noch gesammelt wird, überwachen von **rah** gestartete Prozesse die Aktivitäten, um Nachrichten an das Terminal zu schreiben und anzuzeigen, welche Befehle nicht ausgeführt wurden, sowie um die gepufferte Ausgabe abzurufen.

Informationen zu diesem Vorgang

Anmerkung: Die Informationen in diesem Abschnitt beziehen sich ausschließlich auf Linux- und UNIX-Betriebssysteme.

Die Informationsnachrichten werden in einem Intervall geschrieben, das durch die Umgebungsvariable **RAHWAITTIME** gesteuert wird. Im Hilfetext finden Sie Einzelheiten dazu, wie diese definiert wird. Alle Informationsnachrichten können durch Exportieren von `RAHWAITTIME=0` unterdrückt werden.

Der primäre Überwachungsprozess ist ein Befehl, dessen Befehlsname (wie vom Befehl **ps** angezeigt) **rahwaitfor** lautet. Die erste Informationsnachricht teilt Ihnen die Prozess-ID (pid) dieses Prozesses mit. Alle anderen Überwachungsprozesse erscheinen als **ksh**-Befehle, die das **rah**-Script (bzw. den Namen des symbolischen Links) ausführen. Falls erwünscht, können Sie alle Überwachungsprozesse durch folgenden Befehl stoppen:

```
kill pid
```

Dabei ist *pid* die Prozess-ID des primären Überwachungsprozesses. Geben Sie keine Signalnummer an. Belassen Sie den Standardwert auf 15. Dadurch werden die fernen Befehle in keiner Weise beeinflusst, sondern es wird das automatische Anzeigen der gepufferten Ausgabe verhindert. Beachten Sie, dass es während der

Dauer einer einzigen **rah**-Ausführung zwei oder mehr verschiedene Gruppen von Überwachungsprozessen geben kann, die zu verschiedenen Zeiten aktiv sind. Wenn Sie zu einem Zeitpunkt die aktuelle Gruppe von Überwachungsprozessen stoppen, werden keine weiteren mehr gestartet.

Wenn Ihre reguläre Anmeldeshell keine Korn-Shell (z. B. `/bin/ksh`) ist, können Sie **rah**, jedoch gibt es eine Reihe leicht abweichender Regeln für die Eingabe von Befehlen, die die folgenden Sonderzeichen enthalten:

```
"  nicht ersetztes $ '
```

Weitere Informationen erhalten Sie durch die Eingabe von `rah "?"`. Außerdem darf auf einem Linux- und UNIX-Betriebssystem die Anmeldeshell unter der ID, die das **rah**-Script ausführt, keine Korn-Shell sein, wenn die Anmeldeshell unter der ID, die die fernen Befehle ausführt, keine Korn-Shell ist. (**rah** entscheidet basierend auf der lokalen ID, ob die Shell unter der fernen ID eine Korn-Shell ist.) Die Shell darf keinerlei Substitution oder spezielle Verarbeitung an einer Zeichenfolge durchführen, die in einfache Anführungszeichen gesetzt ist. Die Zeichenfolge muss in der Form belassen werden, in der sie vorliegt.

Einstellen des Standardumgebungsprofils für 'rah' unter Windows

Zum Definieren des Standardumgebungsprofils für den Befehl **rah** wird eine Datei namens `db2rah.env` verwendet, die im Instanzverzeichnis zu erstellen ist.

Informationen zu diesem Vorgang

Anmerkung: Die Informationen dieses Abschnitts beziehen sich ausschließlich auf Windows.

Die Datei sollte folgendes Format haben:

```
; Dies ist eine Kommentarzeile
DB2INSTANCE=instanzname
DB2DBDFT=datenbank
; Dateiende
```

Sie können alle Umgebungsvariablen angeben, die Sie zur Initialisierung der Umgebung für den Befehl **rah** benötigen.

Kapitel 11. Erstellen von Tabellen und anderen zugehörigen Objekten

Tabellen in Umgebungen mit partitionierten Datenbanken

Durch die Erstellung einer Tabelle in mehreren Datenbankpartitionen einer Umgebung mit partitionierten Datenbanken lassen sich Leistungsvorteile erzielen. Die Arbeitslast, die mit dem Abruf von Daten verbunden ist, kann auf die Datenbankpartitionen verteilt werden.

Vorbereitende Schritte

Bevor Sie eine Tabelle erstellen, die physisch geteilt oder verteilt wird, müssen Sie folgende Punkte beachten:

- Tabellenbereiche können sich über mehr als eine Datenbankpartition erstrecken. Die Anzahl der Datenbankpartitionen, über die sie sich erstrecken, hängt von der Anzahl der Datenbankpartitionen in der Datenbankpartitionsgruppe ab.
- Tabellen können in einer Kollokation zusammengefasst werden, indem sie im selben Tabellenbereich gespeichert werden oder in einem anderen Tabellenbereich, der zusammen mit dem ersten Tabellenbereich derselben Datenbankpartitionsgruppe zugeordnet ist.

Informationen zu diesem Vorgang

Die gewünschte Zugehörigkeit einer Tabelle zu mehreren Datenbankpartitionen wird bei der Erstellung der Tabelle angegeben. Bei der Erstellung einer Tabelle in einer partitionierten Datenbankumgebung gibt es eine zusätzliche Option: den *Verteilungsschlüssel*. Ein Verteilungsschlüssel ist ein Schlüssel, der Teil der Definition einer Tabelle ist. Durch ihn wird die Datenbankpartition bestimmt, in der die jeweilige Datenzeile gespeichert wird.

Wenn Sie keinen Verteilungsschlüssel explizit angeben, werden die folgenden Standardwerte verwendet. *Stellen Sie sicher, dass der Standardverteilungsschlüssel geeignet ist.*

- Wenn ein Primärschlüssel in der Anweisung CREATE TABLE angegeben wird, wird die erste Spalte des Primärschlüssels als Verteilungsschlüssel verwendet.
- Wenn es für eine Datenbankpartitionsgruppe mit mehreren Partitionen keinen Primärschlüssel gibt, wird die erste Spalte verwendet, die keine Langfelddaten enthält.
- Wenn die vorhandenen Spalten die Anforderungen für einen Standardverteilungsschlüssel nicht erfüllen, wird die Tabelle ohne Verteilungsschlüssel erstellt (d. h., sie ist nur in Datenbankpartitionsgruppen mit einer Einzelpartition zulässig).

Sie müssen bei der Auswahl eines geeigneten Verteilungsschlüssels mit großer Sorgfalt vorgehen, da *der Schlüssel später nicht mehr geändert werden kann*. Darüber hinaus müssen alle eindeutigen Indizes (und infolgedessen eindeutige Schlüssel bzw. Primärschlüssel) als Obermenge des Verteilungsschlüssels definiert werden. Das heißt, wenn ein Verteilungsschlüssel definiert wird, müssen eindeutige Schlüssel und Primärschlüssel alle die Spalten enthalten, die der Verteilungsschlüssel enthält (sie können mehr Spalten haben).

Die Größe einer Datenbankpartition einer Tabelle ist der kleinere von zwei Werten: entweder die bestimmte Obergrenze, die dem Typ und der Seitengröße des verwendeten Tabellenbereichs zugeordnet ist, oder die Kapazität des verfügbaren Plattenspeicherplatzes. Betrachten Sie zum Beispiel einen großen DMS-Tabellenbereich (Typ LARGE) mit einer Seitengröße von 4 KB. In diesem Fall ist die Größe einer Tabelle der kleinere der folgenden Werte: entweder 8 TB multipliziert mit der Anzahl der Datenbankpartitionen oder die Kapazität des verfügbaren Plattenspeicherplatzes. Die vollständige Liste der Begrenzungen für Seitengrößen des Datenbankmanagers finden Sie über die zugehörigen Links.

Geben Sie in die Befehlszeile Folgendes ein, um eine Tabelle in einer Umgebung mit partitionierten Datenbanken zu erstellen:

```
CREATE TABLE name>
  (<spaltenname> <datentyp> <nullattribut>)
  IN <tabellenbereichsname>
  INDEX IN <indextabellenbereichsname>
  LONG IN <lob-tabellenbereichsname>
  DISTRIBUTE BY HASH (<spaltenname>)
```

Betrachten Sie folgendes Beispiel:

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
                     MIX_INT INTEGER NOT NULL,
                     MIX_INTS SMALLINT NOT NULL,
                     MIX_DEC DECIMAL NOT NULL,
                     MIX_FLT FLOAT NOT NULL,
                     MIX_DATE DATE NOT NULL,
                     MIX_TIME TIME NOT NULL,
                     MIX_TMSTMP TIMESTAMP NOT NULL)
                     IN MIXTS12
                     DISTRIBUTE BY HASH (MIX_INT)
```

In diesem Beispiel heißen der Tabellenbereich MIXTS12 und der Verteilungsschlüssel MIX_INT. Wenn der Verteilungsschlüssel nicht explizit angegeben wird, wird die Spalte MIX_CNTL verwendet. (Wenn kein Primärschlüssel angegeben und kein Verteilungsschlüssel definiert wird, wird die erste Spalte in der Liste, die keine Langfelddaten (LONG) enthält, als Verteilungsschlüssel verwendet.)

Eine Zeile einer Tabelle und sämtliche Informationen zu dieser Zeile werden immer in derselben Datenbankpartition gespeichert.

Verhalten großer Objekte in partitionierten Tabellen

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.

Ein großes Objekt (LOB) für eine partitionierte Tabelle wird standardmäßig in demselben Tabellenbereich wie das entsprechende Datenobjekt gespeichert. Dies gilt für partitionierte Tabellen, die nur einen Tabellenbereich oder mehrere Tabellenbereiche

verwenden. Wenn die Daten einer partitionierten Tabelle in mehreren Tabellenbereichen gespeichert werden, werden auch die LOB-Daten in mehreren Tabellenbereichen gespeichert.

Zum Überschreiben dieses Standardverhaltens können Sie die Klausel LONG IN der Anweisung CREATE TABLE verwenden. Sie können eine Liste der Tabellenbereiche für die Tabelle angeben, in denen die langen Daten gespeichert werden sollen. Wenn Sie das Standardverhalten überschreiben, muss der in der Klausel LONG IN angegebene Tabellenbereich ein LOB-Tabellenbereich (Typ LARGE) sein. Wenn Sie angeben, dass lange Daten (LONG) in einem separaten Tabellenbereich für eine oder mehrere Datenpartitionen gespeichert werden sollen, müssen Sie dies für alle Datenpartitionen der Tabelle tun. Das heißt, es ist nicht möglich, lange Daten für einige Datenpartitionen fern zu speichern und für andere Datenpartitionen lokal zu speichern. Unabhängig davon, ob Sie das Standardverhalten verwenden oder mit der Klausel LONG IN das Standardverhalten überschreiben, wird jeweils ein langes Objekt als Entsprechung für jede Datenpartition erstellt. Alle Tabellenbereiche, die zum Speichern von langen Datenobjekten, die den einzelnen Datenpartitionen entsprechen, verwendet werden, müssen die gleiche Seitengröße (PAGESIZE), die gleiche Speicherbereichsgröße (EXTENTSIZE), den gleichen Speichermechanismus (DMS oder AMS) und den gleichen Typ (REGULAR oder LARGE) besitzen. Ferne LOB-Tabellenbereiche müssen vom Typ LARGE sein und können keine SMS-Tabellenbereiche sein.

Mit der folgenden Anweisung CREATE TABLE werden zum Beispiel Objekte für die CLOB-Daten für jede Datenpartition in demselben Tabellenbereich wie die Daten erstellt:

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1,
 STARTING FROM 101 ENDING AT 200 IN tbsp2,
 STARTING FROM 201 ENDING AT 300 IN tbsp3,
 STARTING FROM 301 ENDING AT 400 IN tbsp4);
```

Sie können die Klausel LONG IN verwenden, um die CLOB-Daten in einem oder mehreren LOB-Tabellenbereichen zu speichern, die sich von denjenigen unterscheiden, in denen die Daten enthalten sind.

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1 LONG IN large1,
STARTING FROM 101 ENDING AT 200 IN tbsp2 LONG IN large1,
STARTING FROM 201 ENDING AT 300 IN tbsp3 LONG IN large2,
STARTING FROM 301 ENDING AT 400 IN tbsp4 LONG IN large2);
```

Anmerkung: Auf Tabellenebene und pro Datenpartition ist nur eine Klausel LONG IN zulässig.

Erstellen partitionierter Tabellen

Partitionierte Tabellen arbeiten mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als Datenpartitionen oder Datenbereiche 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.

Vorbereitende Schritte

Zum Erstellen einer Tabelle müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte beinhalten:

- Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für alle von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
  - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
  - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht
- Berechtigung DBADM

Informationen zu diesem Vorgang

Sie können eine partitionierte Tabelle mit der Anweisung CREATE TABLE erstellen.

Vorgehensweise

Zum Erstellen einer partitionierten Tabelle über die Befehlszeile geben Sie die Anweisung CREATE TABLE ein:

```
CREATE TABLE NAME (spaltenname datentyp null_attribut) IN
 tabellenbereichsliste PARTITION BY RANGE (spaltenausdruck)
STARTING FROM konstante ENDING konstante EVERY konstante
```

Zum Beispiel erstellt die folgende Anweisung eine Tabelle, in der Zeilen mit a ≥ 1 und a ≤ 20 in PART0 (der ersten Datenpartition), Zeilen mit 21 ≤ a ≤ 40 in PART1 (der zweiten Datenpartition) usw. und schließlich Zeilen mit 81 ≤ a ≤ 100 in PART4 (der letzten Datenpartition) gespeichert werden.

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE (a) (STARTING FROM (1)
 ENDING AT (100) EVERY (20))
```

Definieren von Bereichen für partitionierte Tabellen

Bei der Erstellung einer partitionierten Tabelle können Sie einen Bereich für jede Datenpartition angeben. Eine partitionierte Tabelle arbeitet mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Datenpartitionen entsprechend den Werten der Spalten des Tabellenpartitionierungsschlüssels der Tabelle verteilt werden.

Informationen zu diesem Vorgang

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. Ein Bereich wird durch Werte mit STARTING FROM und ENDING AT in der Klausel PARTITION BY angegeben.

Zur vollständigen Definition des Bereichs für eine Datenpartition müssen Sie ausreichende Grenzen spezifizieren. Die folgende Liste enthält Richtlinien, die bei der Definition von Bereichen für eine partitionierte Tabelle berücksichtigt werden sollten:

- Die Klausel STARTING gibt eine untere Grenze für den Datenpartitionsbereich an. Diese Klausel ist obligatorisch für den niedrigsten Datenpartitionsbereich

(obwohl Sie die Grenze als MINVALUE definieren können). Der niedrigste Datenpartitionsbereich ist die Datenpartition mit der niedrigsten angegebenen Grenze.

- Die Klausel ENDING (oder VALUES) gibt eine obere Grenze für den Datenpartitionsbereich an. Diese Klausel ist obligatorisch für den höchsten Datenpartitionsbereich (obwohl Sie die Grenze als MAXVALUE definieren können). Der höchste Datenpartitionsbereich ist die Datenpartition mit der höchsten angegebenen Grenze.
- Wenn Sie für eine Datenpartition keine Klausel ENDING angeben, müssen Sie für die nächst höhere Datenpartition eine Klausel STARTING angeben. Wenn Sie für eine Datenpartition keine Klausel STARTING angeben, müssen Sie analog für die vorherige Datenpartition eine Klausel ENDING angeben.
- MINVALUE gibt einen Wert an, der kleiner als jeder mögliche Wert für den Spaltentyp ist, der verwendet wird. MINVALUE und INCLUSIVE bzw. EXCLUSIVE können nicht zusammen angegeben werden.
- MAXVALUE gibt einen Wert an, der größer als jeder mögliche Wert für den Spaltentyp ist, der verwendet wird. MAXVALUE und INCLUSIVE bzw. EXCLUSIVE können nicht zusammen angegeben werden.
- INCLUSIVE gibt an, dass alle Werte gleich dem angegeben Wert in die Datenpartition, die diese Grenze enthält, mit einzuschließen zu sind.
- EXCLUSIVE gibt an, dass alle Werte gleich dem angegeben Wert in die Datenpartition, die diese Grenze enthält, NICHT mit einzuschließen zu sind.
- Die Klausel NULL der Anweisung CREATE TABLE gibt an, ob Nullwerte bei der Platzierung in einer Datenpartition als hoch oder niedrig einzusortieren sind. Standardmäßig werden Nullwerte hoch einsortiert. Nullwerte in Spalten des Tabellenpartitionierungsschlüssels werden als positiv unendlich behandelt und in einen Bereich eingefügt, der mit MAXVALUE endet. Wenn keine solche Datenpartition definiert ist, werden Nullwerte als außerhalb des gültigen Bereichs betrachtet. Verwenden Sie die Integritätsbedingung NOT NULL, wenn Sie Nullwerte aus den Spalten des Tabellenpartitionierungsschlüssels ausschließen möchten. LAST gibt an, dass Nullwerte als letzte in eine sortierte Liste von Werten einzufügen sind. FIRST gibt an, dass Nullwerte als erste in eine sortierte Liste von Werten einzufügen sind.
- Wenn Sie die Langform der Syntax verwenden, muss für jede Datenpartition mindestens eine Grenze angegeben werden.

Tipp: Vor der Definition von Datenpartitionen für eine Tabelle ist es wichtig, zu verstehen, wie Tabellen von der Tabellenpartitionierung profitieren und welche Faktoren sich auf die Spalten auswirken, die Sie als Partitionierungsspalten auswählen.

Die Bereiche, die für die einzelnen Datenpartitionen angegeben werden, können automatisch oder manuell generiert werden.

Automatische Generierung

Die automatische Generierung ist eine einfache und zeitsparende Methode zur Erstellung zahlreicher Datenpartitionen. Diese Methode eignet sich für Bereiche gleicher Größe auf der Basis von Datumswerten oder Nummern.

Die Beispiele 1 und 2 veranschaulichen, wie die Anweisung CREATE TABLE zur Definition und automatischen Generierung der für die einzelnen Datenpartitionen angegebenen Bereiche verwendet wird.

Beispiel 1:

Führen Sie eine Anweisung CREATE TABLE mit den folgenden definierten Bereichen aus:

```
CREATE TABLE lineitem (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
     PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1992') EVERY 1 MONTH);
```

Diese Anweisung erstellt 12 Datenpartitionen mit jeweils einem Schlüsselwert (l_shipdate) >= ('1/1/1992'), (l_shipdate) < ('3/1/1992'), (l_shipdate) < ('4/1/1992'), (l_shipdate) < ('5/1/1992'), ..., (l_shipdate) < ('12/1/1992'), (l_shipdate) < ('12/31/1992').

Der Anfangswert (STARTING) der ersten Datenpartition ist einschließend, weil die Gesamtanfangsgrenze ('1/1/1992') einschließend ist (standardmäßig). Analog ist die Endgrenze der letzten Datenpartition einschließend, weil die Gesamtendgrenze ('12/31/1992') einschließend ist (standardmäßig). Die übrigen STARTING-Werte sind einschließend, während die übrigen ENDING-Werte alle ausschließend sind. Jede Datenpartition enthält n Schlüsselwerte, wobei n durch die Klausel EVERY gegeben ist. Mithilfe der Formel (STARTING + EVERY) können Sie das Ende des Bereichs jeder einzelnen Datenpartition ermitteln. Die letzte Datenpartition enthält möglicherweise weniger Schlüsselwerte, wenn der EVERY-Wert nicht glatt in den STARTING- und ENDING-Bereich teilbar ist.

Beispiel 2:

Führen Sie eine Anweisung CREATE TABLE mit den folgenden definierten Bereichen aus:

```
CREATE TABLE t(a INT, b INT)
 PARTITION BY RANGE(b) (STARTING FROM (1)
  EXCLUSIVE ENDING AT (1000) EVERY (100))
```

Durch diese Anweisung werden 10 Datenpartitionen jeweils mit 100 Schlüsselwerten (1 < b <= 101, 101 < b <= 201, ..., 901 < b <= 1000) erstellt.

Der Anfangswert der ersten Datenpartition (b > 1 und b <= 101) ist ausschließend, weil der Gesamtanfangswert (1) ausschließend ist. Analog ist die Endgrenze der letzten Datenpartition (b > 901 b <= 1000) einschließend, weil die Gesamtendgrenze (1000) einschließend ist. Die übrigen STARTING-Werte sind alle ausschließend, während die übrigen ENDING-Werte alle einschließend sind. Jede Datenpartition enthält n Schlüsselwerte, wobei n durch die Klausel EVERY gegeben ist. Wenn schließlich beide Grenzen, d. h. STARTING- und ENDING-Grenze, der gesamten Klausel ausschließend sind, ist der STARTING-Wert der ersten Datenpartition ausschließend, weil die Gesamtanfangsgrenze (1) ausschließend ist. Analog ist die Endgrenze der letzten Datenpartition ausschließend, weil die Gesamtendgrenze (1000) ausschließend ist. Die übrigen STARTING-Werte sind alle ausschließend, während die ENDING-Werte alle einschließend sind. Jede Datenpartition (außer der letzten) enthält n Schlüsselwerte, wobei n durch die Klausel EVERY gegeben ist.

Manuelle Generierung

Durch die manuelle Generierung wird eine neue Datenpartition für jeden Bereich erstellt, der in der Klausel PARTITION BY angegeben wird. Diese Form der Syntax bietet eine höhere Flexibilität bei der Definition von Bereichen und dadurch mehr Optionen zur Platzierung von Daten und LOB-Daten. Die Beispiele 3 und 4 veranschaulichen, wie die Anweisung CREATE TABLE zur Definition und manuellen Generierung der für die einzelnen Datenpartitionen angegebenen Bereiche verwendet wird.

Beispiel 3:

Durch die folgende Anweisung wird eine Partitionierung in zwei Datumsspalten ausgeführt, die beide generiert werden. Beachten Sie, dass hier die Form der CREATE TABLE-Syntax für automatische Generierung verwendet und nur jeweils ein Ende der Bereiche angegeben wird. Das andere Ende wird durch die benachbarte Datenpartition und die Verwendung der Option INCLUSIVE impliziert:

```
CREATE TABLE sales(invoice_date date, inv_month int NOT NULL
GENERATED ALWAYS AS (month(invoice_date)), inv_year INT NOT
NULL GENERATED ALWAYS AS ( year(invoice_date)), item_id int NOT NULL,
cust_id int NOT NULL) PARTITION BY RANGE (inv_year, inv_month)
(PART Q1_02 STARTING (2002,1) ENDING (2002, 3) INCLUSIVE,
PART Q2_02 ENDING (2002, 6) INCLUSIVE,
PART Q3_02 ENDING (2002, 9) INCLUSIVE,
PART Q4_02 ENDING (2002,12) INCLUSIVE,
PART CURRENT ENDING (MAXVALUE, MAXVALUE));
```

Lücken zwischen den Bereichen sind zulässig. Die CREATE TABLE-Syntax unterstützt Lücken, indem sie die Angabe eines STARTING-Werts für einen Bereich zulässt, der nicht am ENDING-Wert der vorherigen Datenpartition ausgerichtet ist.

Beispiel 4:

Die folgende Anweisung erstellt eine Tabelle mit einer Lücke zwischen den Werten 101 und 200.

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE(a)
   (STARTING FROM (1) ENDING AT (100),
   STARTING FROM (201) ENDING AT (300))
```

Durch die Verwendung der Anweisung ALTER TABLE, die ein Hinzufügen oder Entfernen von Datenpartitionen ermöglicht, können ebenfalls Lücken zwischen den Bereichen entstehen.

Wenn Sie eine Zeile in eine partitionierte Tabelle einfügen, wird sie automatisch in der ihrem Schlüsselwert entsprechenden Datenpartition und dem zugehörigen Bereich abgelegt. Wenn sie sich außerhalb aller für die Tabelle definierten Bereiche befindet, schlägt die Einfügung fehl und der folgende Fehler wird an die Anwendung zurückgegeben:

```
SQL0327N  Die Zeile kann nicht in Tabelle "<tabellenname>" eingefügt werden,
weil sie sich außerhalb der Grenzen der definierten Datenpartitionsbereiche befindet.
   SQLSTATE=22525
```

Einschränkungen

- Einschränkungen auf Tabellenebene:
  - Tabellen, die mit der Form der Syntax für automatische Generierung (mit der Klausel EVERY) erstellt werden, sind auf die Verwendung eines numerischen oder eines DATE/TIME-Datentyps im Tabellenpartitionierungsschlüssel beschränkt.

- Einschränkungen auf Anweisungsebene:
  - Die Optionen MINVALUE und MAXVALUE werden in der Form der Syntax für automatische Generierung nicht unterstützt.
  - Bereiche sind aufsteigend.
  - In der Form der Syntax für automatische Generierung kann nur eine Spalte angegeben werden.
  - Das Inkrement in der Klausel EVERY muss größer als null sein.
  - Der ENDING-Wert muss größer oder gleich dem STARTING-Wert sein.

Platzierung von Daten, Index und langen Daten einer Datenpartition

Das Erstellen einer partitionierten Tabelle ermöglicht Ihnen, die verschiedenen Teile der Tabelle und die zugeordneten Tabellenobjekte in bestimmten Tabellenbereichen zu platzieren.

Beim Erstellen einer Tabelle können Sie angeben, in welchem Tabellenbereich die gesamten Tabellendaten und zugeordneten Tabellenobjekte platziert werden. Alternativ können Sie von der Tabelle den Index, die langen oder großen Daten oder die Tabellenpartitionen in bestimmten Tabellenbereichen platzieren. Alle Tabellenbereiche müssen sich in derselben Datenbankpartitionsgruppe befinden.

Die Anweisung CREATE TABLE hat die folgenden Klauseln, die diese Fähigkeit zum Platzieren der Tabellendaten und der zugeordneten Tabellenobjekte innerhalb bestimmter Tabellenbereiche veranschaulichen:

```
CREATE TABLE tabellenname IN tabellenbereichsname1
   INDEX IN tabellenbereichsname2
   LONG IN tabellenbereichsname3
   PARTITIONED BY ...
     PARTITION partitionsname | boundary specification | IN tabellenbereichsname4
       INDEX IN tabellenbereichsname5
       LONG IN tabellenbereichsname6
```

Jede Partition der partitionierten Tabelle kann in unterschiedlichen Tabellenbereichen platziert werden.

Sie können auch den Tabellenbereich für einen benutzererstellten, nicht partitionierten Index für eine partitionierte Tabelle mithilfe der Anweisung `CREATE INDEX ... IN tabellenbereichsname1` angeben, der ein anderer ist, als der Tabellenbereich für Indizes, der in der Anweisung `CREATE TABLE ... INDEX IN tabellenbereichsname2` angegeben ist. Die Klausel IN der Anweisung CREATE INDEX wird nur für partitionierte Tabellen verwendet. Wenn die Klausel INDEX IN in den Anweisungen CREATE TABLE oder CREATE INDEX nicht angegeben wird, wird der Index in demselben Tabellenbereich platziert, wie die erste sichtbare oder zugeordnete Partition der Tabelle.

Systemgenerierte, nicht partitionierte Indizes, wie z. B. XML-Spaltenpfadindizes, werden in dem Tabellenbereich platziert, der in der Klausel INDEX IN der Anweisung CREATE TABLE angegeben wurde.

Für eine partitionierte Tabelle mit XML-Daten ist der Regionsindex immer genauso partitioniert wie die Tabellendaten. Der Tabellenbereich der partitionierten Indizes wird auf Partitionsebene definiert.

Die XML-Daten befinden sich in den Tabellenbereichen, die von den langen Daten für eine Tabelle verwendet werden. Die Platzierung der XML-Daten für eine partitionierte Tabelle richten sich nach den Regeln der Platzierung langer Daten.

Der Tabellenbereich für lange Daten kann von Ihnen explizit angegeben oder vom Datenbankmanager implizit festgelegt werden. Bei einer partitionierten Tabelle kann die Klausel LONG IN auf Tabellenebene zusammen mit der Klausel LONG IN auf Partitionsebene verwendet werden. Bei Angabe beider Klauseln erhält die Klausel LONG IN auf Partitionsebene Vorrang vor der Klausel LONG IN auf Tabellenebene.

Migrieren vorhandener Tabellen und Sichten in partitionierte Tabellen

Sie können eine nicht partitionierte Tabelle oder eine UNION ALL-Sicht in eine leere partitionierte Tabelle migrieren.

Vorbereitende Schritte

Die Zuordnung einer Datenpartition ist nicht zulässig, wenn SYSCAT.COLUMNS.IMPLICITVALUE für eine bestimmte Spalte in der Quellen- und der Zielspalte einen Wert ungleich null angibt und die Werte nicht übereinstimmen. In diesem Fall müssen Sie die Quellentabelle löschen und anschließend erneut erstellen.

Eine Spalte kann einen Wert ungleich null im Feld IMPLICITVALUE von SYSCAT.COLUMNS enthalten, wenn eine der folgenden Bedingungen zutrifft:

- Das Feld IMPLICITVALUE wird während der Zuordnung aus einer Quellentabelle weitergegeben.
- Das Feld IMPLICITVALUE wird während der Aufhebung einer Zuordnung aus einer Quellentabelle übernommen.
- Das Feld IMPLICITVALUE wird während der Migration von Version 8 auf Version 9 festgelegt, wobei die Spalte als hinzugefügte Spalte ermittelt wird oder möglicherweise eine hinzugefügte Spalte ist. Eine hinzugefügte Spalte ist eine Spalte, die durch eine Anweisung ALTER TABLE...ADD COLUMN erstellt wurde.

Sie sollten die Quellen- und die Zieltabelle, die an einer Zuordnungsoperation beteiligt sind, stets mit den gleichen definierten Spalten erstellen. Verwenden Sie insbesondere nie die Anweisung ALTER TABLE, um der Zieltabelle einer Zuordnungsoperation Spalten hinzuzufügen.

Informationen zur Vermeidung einer Abweichung bei der Arbeit mit partitionierten Tabellen finden Sie in „Richtlinien zum Hinzufügen von Datenpartitionen zu partitionierten Tabellen” auf Seite 236.

Informationen zu diesem Vorgang

Bei der Migration regulärer Tabellen entladen Sie die Quellentabelle mit dem Befehl **EXPORT** oder einem leistungsfähigen Entladetool. Erstellen Sie eine neue, leere partitionierte Tabelle und verwenden Sie den Befehl **LOAD**, um diese partitionierte Tabelle mit Daten zu füllen. Um die Daten von der alten Tabelle direkt in die partitionierte Tabelle zu versetzen, ohne weitere Zwischenschritte auszuführen, müssen Sie den Befehl **LOAD FROM CURSOR** (siehe Schritt 1) verwenden.

Sie können nicht partitionierte Daten in einer UNION ALL-Sicht in eine partitionierte Tabelle (siehe Schritt 2) konvertieren. UNION ALL-Sichten dienen zur Verwaltung großer Tabellen sowie zur Realisierung einfacher Rollin- und Rollout-Operationen für Tabellendaten bei gleichzeitiger Nutzung der Leistungsvorteile des Verzweigungsausschlusses. Mit der Operation ALTER TABLE...ATTACH PARTITION können Sie die Konvertierung durchführen, ohne Daten in der Basistabelle zu versetzen. Nicht partitionierte Indizes und abhängige Sichten oder MQTs (Materialized Query Tables) müssen nach der Konvertierung erneut erstellt werden. Die empfohlene Strategie zum Konvertieren von UNION ALL-Sichten in partitionierte Tabellen besteht in der Erstellung einer partitionierten Tabelle mit einer einzigen Pseudodatenpartition und der anschließenden Zuordnung aller Tabellen der UNION ALL-Sicht. Stellen Sie sicher, dass Sie die Pseudodatenpartition bei diesem Prozess frühzeitig löschen, um Probleme aufgrund von Bereichsüberschneidungen zu vermeiden.

Vorgehensweise

1. Migrieren Sie eine reguläre Tabelle in eine partitionierte Tabelle. Verwenden Sie den Befehl **LOAD FROM CURSOR**, um die Ausführung von Zwischenschritten zu vermeiden. Das folgende Beispiel zeigt, wie Tabelle T1 in die Tabelle SALES_DP migriert wird:
   a. Erstellen Sie eine reguläre Tabelle T1 und füllen Sie sie mit Daten.
   ```
   CREATE TABLE t1 (c1 int, c2 int);INSERT INTO t1 VALUES (0,1), (4, 2), (6, 3);
   ```
   b. Erstellen Sie eine leere partitionierte Tabelle.
   ```
   CREATE TABLE sales_dp (c1 int, c2 int)
     PARTITION BY RANGE (c1)
     (STARTING FROM 0 ENDING AT 10 EVERY 2);
   ```
   c. Verwenden Sie den Befehl **LOAD FROM CURSOR**, um die Daten von einer SQL-Abfrage direkt in die neue partitionierte Tabelle zu extrahieren.
   ```
   SELECT * FROM t1;
   DECLARE c1 CURSOR FOR SELECT * FROM t1;
   LOAD FROM c1 of CURSOR INSERT INTO sales_dp;SELECT * FROM sales_dp;
   ```
2. Konvertieren Sie nicht partitionierte Daten in einer UNION ALL-Sicht in eine partitionierte Tabelle. Das folgende Beispiel zeigt, wie die UNION ALL-Sicht mit dem Namen ALL_SALES in die Tabelle SALES_DP konvertiert wird:
   a. Erstellen Sie die UNION ALL-Sicht.
   ```
   CREATE VIEW all_sales AS
    (
    SELECT * FROM sales_0198
    WHERE sales_date BETWEEN '01-01-1998' AND '01-31-1998'
    UNION ALL
    SELECT * FROM sales_0298
    WHERE sales_date BETWEEN '02-01-1998' AND '02-28-1998'
    UNION ALL
    ...
    UNION ALL
    SELECT * FROM sales_1200
    WHERE sales_date BETWEEN '12-01-2000' AND '12-31-2000'
    );
   ```
   b. Erstellen Sie eine partitionierte Tabelle mit einer einzigen Pseudopartition. Wählen Sie den Bereich aus, sodass er keine Überschneidung mit der ersten zuzuordnenden Datenpartition aufweist.
   ```
   CREATE TABLE sales_dp (
    sales_date DATE NOT NULL,
    prod_id INTEGER,
    city_id INTEGER,
    channel_id INTEGER,
   ```

```
  revenue DECIMAL(20,2))
  PARTITION BY RANGE (sales_date)
  (PART dummy STARTING FROM '01-01-1900' ENDING AT '01-01-1900');
```

c. Ordnen Sie die erste Tabelle zu.

```
ALTER TABLE sales_dp ATTACH PARTITION
  STARTING FROM '01-01-1998' ENDING AT '01-31-1998'
  FROM sales_0198;
```

d. Löschen Sie die Pseudopartition.

```
ALTER TABLE sales_dp DETACH PARTITION dummy
  INTO dummy;
DROP TABLE dummy;
```

e. Ordnen Sie die verbleibenden Partitionen zu.

```
ALTER TABLE sales_dp ATTACH PARTITION
  STARTING FROM '02-01-1998' ENDING AT '02-28-1998'
  FROM sales_0298;
...
ALTER TABLE sales_dp ATTACH PARTITION
  STARTING FROM '12-01-2000' ENDING AT '12-31-2000'
  FROM sales_1200;
```

f. Setzen Sie die Anweisung SET INTEGRITY ab, damit Abfragen auf Daten in der neu zugeordneten Partition zugreifen können.

```
SET INTEGRITY FOR sales_dp IMMEDIATE CHECKED
FOR EXCEPTION IN sales_dp USE sales_ex;
```

Tipp: Wenn die Prüfung der Datenintegrität (einschließlich Bereichsprüfung und Prüfung anderer Integritätsbedingungen) durch vom Datenserver unabhängige Anwendungslogik vor einer Zuordnungsoperation erfolgen kann, können die neu zugeordneten Daten deutlich früher verfügbar gemacht werden. Sie können den Prozess für Dateneinlagerung optimieren, indem Sie mithilfe der Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED die Bereichsprüfung und die Prüfung auf ungültige Integritätsbedingungen überspringen. In diesem Fall verlässt die Tabelle den Status 'Festlegen der Integrität anstehend' und die neuen Daten sind sofort für Anwendungen verfügbar, sofern die Zieltabelle keine nicht partitionierten Benutzerindizes enthält.

g. Erstellen Sie geeignete Indizes.

Konvertieren vorhandener Indizes in partitionierte Indizes

Vom System oder Benutzer erstellte Indizes müssen möglicherweise aus nicht partitionierten in partitionierte Indizes migriert werden. Bei der Konvertierung benutzergenerierter Indizes bleiben die Tabelle und Indizes für den größten Teil der Migration verfügbar. Bei systemgenerierten Indizes, die zur Umsetzung von über Primärschlüssel definierten Integritätsbedingungen oder von eindeutigen Integritätsbedingungen verwendet werden, werden die Integritätsbedingungen während der Konvertierung nicht aufrechterhalten.

Vorbereitende Schritte

Indizes, die unter einer früheren Version des Produkts erstellt wurden, sind möglicherweise nicht partitioniert. Dabei kann es sich um Indizes handeln, die Sie selbst erstellt haben, oder um systemgenerierte Indizes, die der Datenbankmanager erstellt hat. Beispiele systemgenerierter Indizes sind Indizes, die eindeutige bzw. über Primärschlüssel definierte Integritätsbedingungen, und die Blockindizes einer MDC-Tabelle.

Informationen zu diesem Vorgang

Indizes, die Sie erstellt haben, können von nicht partitionierten in partitionierte Indizes konvertiert werden, wobei die Daten, die den Index verwenden, während der ganzen Zeit verfügbar bleiben. Sie können einen partitionierten Index mit denselben Schlüsseln wie der entsprechende nicht partitionierte Index erstellen. Während der Partitionierungsindex erstellt wird, können Sie weiterhin die aktuellen Indizes und die Tabelle verwenden, in der der Index erstellt wird. Nach Erstellung des partitionierten Indexes können Sie den entsprechenden nicht partitionierten Index löschen und den neuen partitionierten Index bei Bedarf umbenennen.

Ergebnisse

Die folgenden Beispiele zeigen, wie vorhandene nicht partitionierte Indizes in partitionierte Indizes umgewandelt werden.

Beispiel

Im folgenden Beispiel wird gezeigt, wie Sie einen nicht partitionierten Index, den Sie erstellt haben, in einen partitionierten Index konvertieren können:

```
UPDATE COMMAND OPTIONS USING C OFF;
CREATE INDEX data_part ON sales(sale_date) PARTITIONED;
DROP INDEX dateidx;
RENAME INDEX data_part TO dateidx;
COMMIT;
```

Im folgenden Beispiel wird gezeigt, wie ein nicht partitionierter Index, der vom Datenbankmanager erstellt wurde, in einen partitionierten Index konvertiert werden kann. In diesem Fall vergeht eine gewisse Zeit zwischen dem Löschen der Integritätsbedingung und der Erstellung der neuen Integritätsbedingung.

```
ALTER TABLE employees DROP CONSTRAINT emp_uniq;
ALTER TABLE employees ADD CONSTRAINT emp_uniq UNIQUE (employee_id);
```

MDC-Tabellen, die mit DB2 Version 9.7 und früheren Releases erstellt wurden, verfügen über nicht partitionierte Blockindizes. Zur Nutzung von Verfügbarkeitsfunktionen für Daten partitionierter Tabellen, wie z. B. Daten-Rollin/-Rollout oder Reorganisation von Tabellendaten und -indizes auf Partitionsebene, müssen die Daten in der MDC-Tabelle (MDC = Multidimensional Clustering), die mit DB2 V9.7 und früheren Releases erstellt wurden, in eine partitionierte MDC-Tabelle mit partitionierte Blockindizes versetzt werden, die mit DB2 V9.7 Fixpack 1 oder einem späteren Release erstellt wurde.

Onlineprozedur zum Versetzen einer partitionierten MDC-Tabelle zur Verwendung partitionierter Blockindizes

Mithilfe einer Onlineprozedur zum Versetzen von Tabellen können Sie Daten aus einer MDC-Tabelle mit nicht partitionierten Blockindizes in eine MDC-Tabelle mit partitionierten Blockindizes versetzen.

Im folgenden Beispiel enthält die Tabelle `company1.parts` die MDC-Schlüsselspalten **`region`** und **`color`**; die entsprechenden Blockindizes sind nicht partitioniert.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
 'COMPANY1',  --Tabellenschema
 'PARTS',     --Tabellenname
 ' ',         --null; Keine Änderung der Spaltendefinition
 ' ',         --null; Keine zusätzlichen Optionen
 'MOVE');     --Tabelle in einem Schritt versetzen
```

Offlineprozedur zum Versetzen einer partitionierten MDC-Tabelle zur Verwendung partitionierter Blockindizes

Um den Aufwand für das Versetzen von Daten zu minimieren, können Sie Daten aus einer MDC-Tabelle mit nicht partitionierten Blockindizes in eine MDC-Tabelle mit partitionierten Blockindizes versetzen, wenn sich die Tabelle im Offlinemodus befindet. Der Prozess umfasst die folgenden Schritte:

1. Erstellen Sie eine neue MDC-Tabelle mit einer Partition mit derselben Definition wie die zu konvertierende Tabelle. Wenn Sie den Bereich für die Partition angeben, verwenden Sie einen Bereich außerhalb der Bereiche der zu konvertierenden partitionierten MDC-Tabelle.

   Die Blockindizes neuer MDC-Tabellen mit einer Partition sind partitioniert. Die Zuordnung der Partition, die bei Angabe des Bereichs erstellt wird, wird in einem späteren Schritt aufgehoben.
2. Heben Sie die Zuordnung jeder Partition der MDC-Tabelle auf. Jede Partition wird zu einer eigenständigen MDC-Tabelle.

   Wenn die Zuordnung einer Partition aufgehoben wird, werden die Partitionsdaten einer neuen Zieltabelle zugeordnet, ohne die Daten in der Partition zu versetzen.

   Anmerkung: Die Zuordnung der letzten Partition der MDC-Tabelle kann nicht aufgehoben werden. Sie ist eine MDC-Tabelle mit einer Partition mit nicht partitionierten Blockindizes.
3. Für jede eigenständige Tabelle, die durch Aufhebung der Zuordnung der MDC-Tabellenpartitionen erstellt wurde, und für die MDC-Tabelle mit einer Partition mit nicht partitionierten Blockindizes ordnen Sie die Tabelle der neuen partitionierten MDC-Tabelle zu, die in Schritt 1 erstellt wurde.

   Beim Zuordnen der Tabelle werden die Tabellendaten der neuen partitionierten MDC-Tabelle ohne Versetzen der Daten zugeordnet und die Blockindizes werden als partitionierte Blockindizes erstellt.
4. Nach dem Zuordnen der ersten eigenständigen MDC-Tabelle können Sie die Zuordnung der leeren Partition aufheben, die beim Erstellen der neuen MDC-Tabelle generiert wurde.
5. Setzen Sie die Anweisung SET INTEGRITY für die neue partitionierte MDC-Tabelle ab.

Nächste Schritte

Verhalten von partitionierten MQTs

Alle Typen von MQTs (Materialized Query Table, gespeicherte Abfragetabelle) werden mit partitionierten Tabellen unterstützt. Wenn Sie mit partitionierten MQTs arbeiten, können Ihnen einige Richtlinien helfen, zugeordnete Datenpartitionen und Datenpartitionen mit aufgehobener Zuordnung möglichst effektiv zu verwalten.

Die folgenden Richtlinien und Einschränkungen gelten für die Arbeit mit partitionierten MQTs bzw. partitionierten Tabellen mit abhängigen Tabellen, deren Zuordnung aufgehoben wurde:

- Wenn Sie eine Anweisung ALTER TABLE ... DETACH PARTITION eingeben, erstellt die DETACH-Operation die Zieltabelle für die Daten der Partition, für die die Zuordnung aufgehoben wurde. Wenn abhängige Tabellen vorhanden sind, die im Hinblick auf die Datenpartition mit aufgehobener Zuordnung inkrementell gewartet werden müssen (solche abhängigen Tabellen werden als abhängige Tabellen mit aufgehobener Zuordnung bezeichnet), muss die Anweisung SET

INTEGRITY für die abhängigen Tabellen mit aufgehobener Zuordnung ausgeführt werden, um die Tabellen inkrementell zu warten. Nach der Ausführung der Anweisung SET INTEGRITY für alle abhängigen Tabellen mit aufgehobener Zuordnung macht die Task zur asynchronen Aufhebung von Partitionszuordnungen die Datenpartition ab DB2 V9.7 Fixpack 1 zu einer eigenständigen Zieltabelle. Bis zum Abschluss der asynchronen DETACH-Operation für Partitionen ist die Zieltabelle nicht verfügbar. Die Zieltabelle wird mit dem Wert 'L' in der Spalte TYPE der Katalogsicht SYSCAT.TABLES markiert. Dies kennzeichnet sie als Tabelle mit aufgehobener Zuordnung (bzw. als freigegebene Tabelle). Dadurch wird verhindert, dass die Zieltabelle gelesen, modifiziert oder gelöscht wird, bevor die Anweisung SET INTEGRITY ausgeführt wird, um die freigegebenen abhängigen Tabellen inkrementell zu warten. Nach der Ausführung der Anweisung SET INTEGRITY für alle abhängigen Tabellen mit aufgehobener Zuordnung wird für die Datenpartition die logische Zuordnung zur Quellentabelle aufgehoben und die asynchrone DETACH-Operation für Partitionen hebt die Zuordnung der Datenpartition von der Quellentabelle zur Zieltabelle auf. Bis zum Abschluss der asynchronen DETACH-Operation für Partitionen ist die Zieltabelle nicht verfügbar.

- Fragen Sie die Katalogsicht SYSCAT.TABDETACHEDDEP ab, um eine Tabelle mit aufgehobener Zuordnung zu ermitteln, auf die noch nicht zugegriffen werden kann. Wenn Tabellen mit aufgehobener Zuordnung erkannt werden, auf die noch nicht zugegriffen werden kann, führen Sie die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED für alle freigegebenen abhängigen Tabellen aus, um die Tabelle mit aufgehobener Zuordnung in eine reguläre Tabelle umzuwandeln, auf die zugegriffen werden kann. Wenn Sie versuchen, auf eine Tabelle mit aufgehobener Zuordnung zuzugreifen, bevor die zugehörigen abhängigen Tabellen gewartet werden, wird der Fehlercode SQL20285N zurückgegeben.
- Die Funktion DATAPARTITIONNUM kann in der Definition einer MQT nicht verwendet werden. Ein Versuch, eine MQT mit dieser Funktion zu erstellen, führt zu einem Fehler (SQLCODE SQL20058N, SQLSTATE 428EC).
- Wenn ein nicht partitionierter Index für eine Tabelle mit Datenpartitionen mit aufgehobener Zuordnung und dem Status D in SYSCAT.DATAPARTITIONS erstellt wird, enthält der Index die Daten der Partitionen mit aufgehobener Zuordnung nur dann, wenn die Datenpartition mit aufgehobener Zuordnung über eine MQT verfügt, die im Hinblick auf die Datenpartition inkrementell aktualisiert werden muss. In diesem Fall enthält der Index die Daten für diese Datenpartition mit aufgehobener Zuordnung.
- Das Ändern einer Tabelle mit zugeordneten Datenpartitionen in eine MQT ist nicht zulässig.
- Partitionierte Zwischenspeichertabellen werden nicht unterstützt.
- Das Zuordnung von Datenpartitionen zu einer MQT wird nicht direkt unterstützt. Weitere detaillierte Informationen finden Sie unter Beispiel 1.

Beispiel 1: Konvertieren einer partitionierten MQT in eine normale Tabelle

Obwohl die Operation ATTACH nicht direkt für partitionierte MQTs unterstützt wird, können Sie denselben Effekt erzielen, indem Sie eine partitionierte MQT in eine normale Tabelle umwandeln, die gewünschten Rollin- und Rollout-Operationen für Tabellendaten ausführen und anschließend die Tabelle wieder in eine MQT umwandeln. Die folgenden Beispielanweisungen CREATE TABLE und ALTER TABLE veranschaulichen, wie sich dieser Effekt realisieren lässt:

```
CREATE TABLE lineitem (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
```

```
  l_shipdate    DATE,
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
    PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1993') EVERY 1 MONTH);
CREATE TABLE lineitem_ex (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT,
  ts            TIMESTAMP,
  msg           CLOB(32K));

CREATE TABLE quan_by_month (
  q_year_month, q_count) AS
    (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
     FROM lineitem
     GROUP BY l_year_month)
     DATA INITIALLY DEFERRED REFRESH IMMEDIATE
     PARTITION BY RANGE(q_year_month)
     (STARTING (199201) ENDING (199212) EVERY (1),
      STARTING (199301) ENDING (199312) EVERY (1));
CREATE TABLE quan_by_month_ex(
  q_year_month  INT,
  q_count       INT NOT NULL,
  ts            TIMESTAMP,
  msg           CLOB(32K));

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
CREATE INDEX qbmx ON quan_by_month(q_year_month);

ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;

SET INTEGRITY FOR li_reuse OFF;
ALTER TABLE li_reuse ALTER l_year_month SET GENERATED ALWAYS
AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate));


LOAD FROM part_mqt_rotate.del OF DEL MODIFIED BY GENERATEDIGNORE
MESSAGES load.msg REPLACE INTO li_reuse;

DECLARE load_cursor CURSOR FOR
  SELECT l_year_month, COUNT(*)
    FROM li_reuse
    GROUP BY l_year_month;
LOAD FROM load_cursor OF CURSOR MESSAGES load.msg
  REPLACE INTO qm_reuse;

ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
  ENDING '1/31/1994' FROM li_reuse;

SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN lineitem USE lineitem_ex;

ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
  ENDING 199401 FROM qm_reuse;

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED
  FOR EXCEPTION IN quan_by_month USE quan_by_month_ex;

ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
  (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
     FROM lineitem
       GROUP BY l_year_month)
```

```
        DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

Verwenden Sie die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED, um die zugeordnete Datenpartition auf Integritätsverletzungen zu überprüfen. Dieser Schritt muss erfolgen, bevor die Tabelle wieder in eine MQT zurückgeändert wird. Die Anweisung SET INTEGRITY mit der Option IMMEDIATE UNCHECKED dient zur Umgehung der erforderlichen vollständigen Aktualisierung der MQT. Der Index für die MQT ist erforderlich, um eine optimale Leistung zu erreichen. Die Verwendung von Ausnahmetabellen mit der Anweisung SET INTEGRITY wird empfohlen, sofern geeignet.

Typischerweise wird eine partitionierte MQT für eine große Fakttabelle erstellt, die ebenfalls partitioniert ist. Wenn Sie ein Rollout oder Rollin von Tabellendaten für eine große Fakttabelle durchführen, müssen Sie die partitionierte MQT manuell anpassen, wie in Beispiel 2 gezeigt.

Beispiel 2: Manuelles Anpassen einer partitionierten MQT

Ändern Sie die MQT (quan_by_month), um sie in eine normale partitionierte Tabelle umzuwandeln:

```
ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
```

Heben Sie die Zuordnung der Daten auf, für die ein Rollout aus der Fakttabelle (lineitem) und der MQT ausgeführt werden soll, und laden Sie die neuen Daten, für die ein Rollin durchgeführt werden soll, in die Zwischenspeichertabelle (li_reuse):

```
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
LOAD FROM part_mqt_rotate.del OF DEL MESSAGES load.msg REPLACE INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;
```

Bereinigen Sie die Tabelle 'qm_reuse', bevor Sie die Einfügung ausführen. Dadurch werden die Daten mit aufgehobener Zuordnung gelöscht, bevor die Subselect-Daten eingefügt werden. Dies wird durch Laden (LOAD) mit der Option REPLACE in die MQT erreicht, wobei die Datendatei (datafile.del) der Ladeoperation der Inhalt des Subselect ist.

```
db2 load from datafile.del of del replace into qm_reuse
```

Sie können die Tabelle manuell mit der Anweisung INSERT INTO ... (SELECT ...) aktualisieren. Dies ist nur für die neuen Daten erforderlich, sodass die Anweisung vor dem Zuordnen ausgeführt werden sollte:

```
INSERT INTO qm_reuse
  (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
       FROM li_reuse
         GROUP BY l_year_month);
```

Jetzt können Sie das Rollin der neuen Daten für die Fakttabelle durchführen:

```
ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
ENDING '1/31/1994' FROM TABLE li_reuse;
SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR
EXCEPTION IN li_reuse USE li_reuse_ex;
```

Im nächsten Schritt führen Sie das Rollin der Daten für die MQT durch:

```
ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
ENDING 199401 FROM TABLE qm_reuse;
SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
```

Nach der Zuordnung der Datenpartition müssen die neuen Daten überprüft werden, um sicherzustellen, dass sie im gültigen Bereich liegen.

```
ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
  (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
     FROM lineitem
       GROUP BY l_year_month)
         DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

Der Zugriff auf die Daten ist erst möglich, wenn sie durch die Anweisung SET INTEGRITY geprüft wurden. Obwohl auch die Operation REFRESH TABLE unterstützt wird, zeigt dieses Szenario die manuelle Pflege einer partitionierten MQT durch die Operationen ATTACH PARTITION und DETACH PARTITION. Die Daten werden durch den Benutzer mithilfe der Klausel IMMEDIATE UNCHECKED der Anweisung SET INTEGRITY als geprüft markiert.

Erstellen von Bereichsclustertabellen (RCT)

Richtlinien zur Verwendung von Bereichsclustertabellen

In diesem Abschnitt werden einige Richtlinien aufgeführt, die für die Arbeit mit Bereichsclustertabellen (RCT - Range-Clustered Tables) zu befolgen sind.

- Da beim Erstellen einer Bereichsclustertabelle der erforderliche Plattenspeicherplatz vorab zugeordnet wird, muss dieser Speicherplatz verfügbar sein.
- Beim Definieren des Bereichs der Schlüsselwerte ist der Mindestwert optional. Wird dieser Wert nicht angegeben, wird als Standardwert 1 verwendet. Ein negativer Mindestwert muss explizit angegeben werden. Beispiel:

  ```
  ORGANIZE BY KEY SEQUENCE (f1 STARTING FROM -100 ENDING AT -10)
  ```

- Es ist nicht möglich, einen regulären Index für dieselben Schlüsselwerte zu erstellen, die für die Definition der Bereichsclustertabelle verwendet werden.
- Optionen der Anweisung ALTER TABLE, die die physische Struktur der Tabelle beeinflussen, sind nicht zulässig.

Szenarios: Bereichsclustertabellen

Bereichsclustertabellen können einspaltige oder mehrspaltige Schlüssel aufweisen und Zeilen mit Schlüsselwerten außerhalb des definierten Wertebereichs zulassen oder nicht zulassen. In diesem Abschnitt sind Szenarios enthalten, die die Erstellung solcher Tabellen veranschaulichen.

Szenario 1: Erstellen einer Bereichsclustertabelle (Überlauf zulässig)

Im folgenden Beispiel sehen Sie eine Bereichsclustertabelle, die zum Abrufen von Daten zu einem bestimmten Kursteilnehmer verwendet werden kann. Jeder Kursteilnehmerdatensatz enthält die folgenden Informationen:

- ID der Schule (SCHOOOL_ID)
- ID des Kurses (PROGRAM_ID)
- Kursteilnehmernummer (STUDENT_NUM)
- Kursteilnehmer-ID (STUDENT_ID)
- Vorname des Kursteilnehmers (FIRST_NAME)

- Nachname des Kursteilnehmers (LAST_NAME)
- Durchschnittliche Leistung des Kursteilnehmers (GPA)

```
CREATE TABLE students (
 school_id      INT NOT NULL,
 program_id     INT NOT NULL,
 student_num    INT NOT NULL,
 student_id     INT NOT NULL,
 first_name     CHAR(30),
 last_name      CHAR(30),
 gpa            FLOAT
)
 ORGANIZE BY KEY SEQUENCE
  (student_id STARTING FROM 1 ENDING AT 1000000)
  ALLOW OVERFLOW
;
```

In diesem Beispiel wird die Spalte STUDENT_ID, die als Tabellenschlüssel dient, zum Hinzufügen, Aktualisieren oder Löschen von Kursteilnehmerdatensätzen verwendet.

Die Größe der einzelnen Datensätze basiert auf der Summe der zugehörigen Spaltenlängen. In diesem Beispiel hat jeder Datensatz eine Größe von 97 Byte (10 Byte großer Header + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 Byte für Spalten, die Nullwerte enthalten können). Bei einer Seitengröße von 4 KB (oder 4096 Byte) sind nach Abrechnung des Systemaufwands 4038 Byte pro Seite verfügbar (genug für 41 Datensätze). Insgesamt sind 24391 solcher Seiten erforderlich, um 1 Million Kursteilnehmerdatensätze unterzubringen. Unter der Annahme von vier Seiten für den Tabellenaufwand und drei Seiten für die Speicherbereichszuordnung würden 24384 Seiten mit einer Größe von 4 KB bei der Erstellung dieser Tabelle vorab zugeordnet. (Bei der Speicherbereichszuordnung wird von einem einzigen Container mit drei Seiten für die Tabelle ausgegangen.)

Szenario 2: Erstellen einer Bereichsclustertabelle (Überlauf nicht zulässig)

Im folgenden Beispiel umfasst eine Schulvereinigung 200 Schulen, die jeweils 20 Schulungsräume mit einer Kapazität von je 35 Kursteilnehmern umfassen. Diese Schulvereinigung kann maximal 140.000 Kursteilnehmer unterbringen.

```
CREATE TABLE students (
 school_id      INT NOT NULL,
 class_id       INT NOT NULL,
 student_num    INT NOT NULL,
 student_id     INT NOT NULL,
 first_name     CHAR(30),
 last_name      CHAR(30),
 gpa            FLOAT
)
 ORGANIZE BY KEY SEQUENCE
  (school_id STARTING FROM 1 ENDING AT 200,
   class_id STARTING FROM 1 ENDING AT 20,
   student_num STARTING FROM 1 ENDING AT 35)
  DISALLOW OVERFLOW
;
```

In diesem Beispiel dienen die Spalten SCHOOL_ID, CLASS_ID und STUDENT_NUM zusammen als Tabellenschlüssel, der zum Hinzufügen, Aktualisieren oder Löschen von Kursteilnehmerdatensätzen verwendet wird.

Ein Überlauf ist nicht zulässig, da Richtlinien für Schulvereinigungen die Anzahl der Kursteilnehmer in den einzelnen Schulungsräumen begrenzen; außerdem gibt

es eine feste Anzahl an Schulen und Schulungsräumen, die von dieser Schulvereinigung verwaltet werden. Einige kleinere Schulen (d. h. Schulen mit weniger Schulungsräumen als die größte Schule) verfügen über vorab zugeordneten Speicher in der Tabelle, der wahrscheinlich nie verwendet wird.

Aspekte der Erstellung von MDC- oder ITC-Tabellen

Bei der Erstellung von MDC- oder ITC-Tabellen sind zahlreiche Faktoren zu beachten. Die Entscheidungen zur Erstellung, Platzierung und Verwendung Ihrer MDC- oder ITC-Tabellen können durch Ihre aktuelle Datenbankumgebung (z. B. ob es sich um eine partitionierte Datenbank handelt oder nicht) und durch Ihre Auswahl in Bezug auf Dimensionen beeinflusst werden.

Versetzen von Daten aus vorhandenen Tabellen in MDC-Tabellen

Zur Verbesserung der Abfrageleistung und zur Verringerung der Anforderungen für Datenpflegeoperationen in einem Data-Warehouse oder einer großen Datenbankumgebung können Sie Daten aus regulären Tabellen in Tabellen mit mehrdimensionalem Clustering (MDC) versetzen. Gehen Sie wie folgt vor, um Daten aus einer vorhandenen Tabelle in eine MDC-Tabelle zu versetzen:

1. Exportieren Sie Ihre Daten.
2. Löschen Sie die ursprüngliche Tabelle (optional).
3. Erstellen Sie eine MDC-Tabelle (mithilfe der Anweisung CREATE TABLE mit der Klausel ORGANIZE BY DIMENSIONS).
4. Laden Sie Ihre Daten in die MDC-Tabelle.

Eine ALTER TABLE-Prozedur mit dem Namen SYSPROC.ALTOBJ kann zur Ausführung der Umsetzung von Daten aus einer vorhandenen Tabelle in eine MDC-Tabelle verwendet werden. Die Prozedur wird über den DB2-Designadvisor aufgerufen. Die Zeit, die zur Umsetzung der Daten zwischen den Tabellen erforderlich ist, kann enorm sein und hängt von der Größe der Tabelle und der Menge von Daten ab, die umgesetzt werden müssen.

Die Prozedur ALTOBJ führt bei der Änderung einer Tabelle die folgenden Schritte aus:

1. Löschen aller abhängigen Objekte der Tabelle.
2. Umbenennen der Tabelle.
3. Erstellen der Tabelle mit der neuen Definition.
4. Erneutes Erstellen aller abhängigen Objekte der Tabelle.
5. Umsetzen vorhandener Daten in der Tabelle in die Daten, die für die neue Tabelle erforderlich sind. Das heißt, es werden die Daten aus der alten Tabelle ausgewählt und in die neue Tabelle geladen, wobei Spaltenfunktionen zur Umsetzung eines alten Datentyps in einen neuen Datentyp eingesetzt werden können.

Versetzen von Daten aus vorhandenen Tabellen in ITC-Tabellen

Zur Verringerung der Anforderungen für Datenpflegeoperationen können Sie Daten aus regulären Tabellen in ITC-Tabellen (ITC = Insert Time Clustering; Clustering anhand der Einfügungszeit) versetzen. Verwenden Sie die folgende gespeicherte Prozedur zur Onlinetabellenversetzung, um Daten aus einer vorhandenen Tabelle in eine ITC-Tabelle zu versetzen:

Im Szenario "ExampleBank" wird gezeigt, wie Daten aus einer vorhandenen Tabelle in eine ITC-Tabelle versetzt werden. Das Szenario zeigt außerdem, welche Vorteile die Freigabe von Speicherplatz bei der Verwendung von ITC-Tabellen hat. Weitere Informationen finden Sie in den Links zu den zugehörigen Konzepten.

MDC-Advisorfunktion im DB2-Designadvisor

Der DB2-Designadvisor (**db2advis**) verfügt über eine MDC-Funktion. Diese Funktion empfiehlt Clusteringdimensionen zur Verwendung in einer MDC-Tabelle, einschließlich Vergröberungen (d. h. Verringerungen der Granularität) von Basisspalten, um die Auslastungsleistung zu verbessern. Mit der Bezeichnung *Vergröberung* ist ein mathematischer Ausdruck zur Verringerung der Kardinalität (Anzahl unterschiedlicher Werte) in einer Clusterdimension gemeint. Ein allgemeines Beispiel ist die Vergröberung nach Datum, Woche des Datums, Monat des Datums oder Quartal des Jahres.

Eine Voraussetzung für die Verwendung der MDC-Funktion des DB2-Designadvisors ist das Vorhandensein mindestens mehrerer EXTENTSIZE großer Speicherbereiche mit Daten innerhalb der Datenbank. Der DB2-Designadvisor verwendet die Daten zur Modellierung der Datendichte und Kardinalität.

Wenn die Datenbank keine Daten in den Tabellen enthält, empfiehlt der DB2-Designadvisor kein MDC, selbst wenn die Datenbank zwar leere Tabellen enthält, jedoch über nachgebildete Statistikdaten verfügt, die eine mit Daten gefüllte Datenbank simulieren.

Die Empfehlung schließt die Ermittlung potenzieller generierter Spalten mit ein, die eine Vergröberung von Dimensionen definieren. Die Empfehlung bezieht sich nicht auf mögliche Blockgrößen. Bei der Generierung von Empfehlungen für MDC-Tabellen wird der Wert für EXTENTSIZE des Tabellenbereichs zugrunde gelegt. Die Empfehlung geht von der Annahme aus, dass die empfohlene MDC-Tabelle im gleichen Tabellenbereich wie die vorhandene Tabelle erstellt wird, sodass sie die gleiche Speicherbereichsgröße hat. Die Empfehlungen für MDC-Dimensionen variieren entsprechend dem Wert für EXTENTSIZE des Tabellenbereichs, da die Speicherbereichsgröße Auswirkungen darauf hat, wie viele Datensätze in einen Block oder eine Zelle passen. Die Speicherbereichsgröße hat unmittelbaren Einfluss auf die Dichte der Zellen.

Lediglich Dimensionen, die nur aus einer Spalte und nicht aus mehreren Spalten zusammengesetzt sind, werden in Betracht gezogen, obwohl einzelne oder mehrere Dimensionen für die Tabelle empfohlen werden können. Die MDC-Funktion empfiehlt Vergröberungen für die meisten unterstützten Datentypen, um die Kardinalität von Zellen in der resultierenden MDC-Lösung zu reduzieren. Ausnahmen bilden die Datentypen CHAR, VARCHAR, GRAPHIC und VARGRAPHIC. Alle unterstützten Datentypen werden in INTEGER umgesetzt und ihre Kardinalität durch einen generierten Ausdruck verringert.

Das Ziel der MDC-Funktion des DB2-Designadvisors besteht darin, MDC-Lösungen auszuwählen, die zu einer Leistungssteigerung führen. Ein sekundäres Ziel besteht darin, das Anwachsen des Speicherbedarfs der Datenbank auf einer moderaten Ebene zu halten. Das maximale Speicherwachstum jeder Tabelle wird mithilfe einer statistischen Methode ermittelt.

Die Analyseoperation im Designadvisor berücksichtigt nicht nur die Vorteile des Zugriffs über Blockindizes, sondern auch die Auswirkungen des mehrdimensionalen Clustering (MDC) auf Einfüge-, Aktualisierungs- und Löschoperationen für Di-

mensionen der Tabelle. Diese Aktionen an der Tabelle können potenziell bewirken, dass Datensätze zwischen Zellen versetzt werden. Darüber hinaus modelliert die Analyseoperation die potenziellen Leistungsauswirkungen einer Tabellenerweiterung, die sich aus der Organisation von Daten nach bestimmten MDC-Dimensionen ergibt.

Die MDC-Funktion wird über die Markierung `-m <empfehlungstyp>` im Dienstprogramm 'db2advis' ausgeführt. Der Empfehlungstyp „C" dient zur Angabe von MDC-Tabellen. Folgende Empfehlungstypen sind verfügbar: „I" für Index, „M" für MQTs (Materialized Query Tables, gespeicherte Abfragetabellen), „C" für MDC und „P" für eine Umgebung mit partitionierten Datenbanken. Die Empfehlungstypen können miteinander kombiniert werden.

Anmerkung: Der DB2-Designadvisor untersucht keine Tabellen, die weniger als zwölf EXTENTSIZE-Speicherbereiche groß sind.

Der Designadvisor analysiert sowohl MQTs als auch reguläre Basistabellen, um Empfehlungen vorzuschlagen.

Die Ausgabe aus der MDC-Funktion umfasst folgende Informationen:

- Ausdrücke für generierte Spalten für jede Tabelle zur Vergröberung von Dimensionen, die in der MDC-Lösung auftreten.
- Eine Empfehlung für eine ORGANIZE BY DIMENSIONS-Klausel für jede Tabelle.

Die Empfehlungen werden sowohl auf der Standardausgabeeinheit (stdout) als auch in den ADVISE-Tabellen ausgegeben, die zur EXPLAIN-Einrichtung gehören.

MDC-Tabellen und Umgebungen mit partitionierten Datenbanken

Das mehrdimensionale Clustering (MDC) kann in einer Umgebung mit partitionierten Datenbanken verwendet werden. Tatsächlich kann MDC eine Umgebung mit partitionierten Datenbanken ergänzen. Eine Umgebung mit partitionierten Datenbanken dient zur Verteilung von Daten aus einer Tabelle auf mehrere physische oder logische Datenbankpartitionen zu folgenden Zwecken:

- Nutzung der Vorteile mehrerer Maschinen, um die Parallelverarbeitung von Anforderungen zu verbessern
- Physische Vergrößerung der Tabelle über die Grenzen einer einzelnen Datenbankpartition hinaus
- Verbesserung der Skalierbarkeit der Datenbank

Der Grund zur Verteilung einer Tabelle ist davon unabhängig, ob die Tabelle eine MDC-Tabelle oder eine reguläre Tabelle ist. Zum Beispiel unterscheiden sich die Regeln zur Auswahl von Spalten für den Verteilungsschlüssel nicht. Der Verteilungsschlüssel für eine MDC-Tabelle kann jede beliebige Spalte ohne Rücksicht darauf enthalten, ob die Spalten Teil einer Dimension der Tabelle sind oder nicht.

Wenn der Verteilungsschlüssel mit einer Dimension aus der Tabelle identisch ist, enthält jede Datenbankpartition einen anderen Abschnitt der Tabelle. Beispiel: Wenn die MDC-Tabelle aus dem erläuterten Beispiel nach Farbe auf zwei Datenbankpartitionen verteilt wird, wird die Spalte 'Farbe' zur Unterteilung der Daten verwendet. Infolgedessen können zum Beispiel die Sektoren 'Rot' und 'Blau' in einer Datenbankpartition und der Sektor 'Gelb' in einer anderen Datenbankpartition abgelegt werden. Wenn der Verteilungsschlüssel nicht mit den Dimensionen aus der Tabelle identisch ist, wird in jeder Datenbankpartition eine Untergruppe der

Daten aus allen Sektoren gespeichert. Bei der Auswahl der Dimensionen und der Abschätzung der Zellbelegung ist zu beachten, dass sich die durchschnittliche Menge der Daten pro Zelle durch Dividieren der gesamten Daten durch die Anzahl der Datenbankpartitionen bestimmen lässt.

MDC-Tabellen mit mehreren Dimensionen

Wenn Sie wissen, dass bestimmte Vergleichselemente in Abfragen sehr häufig verwendet werden, können Sie die Tabelle nach den beteiligten Spalten in Clustern organisieren. Hierfür können Sie die Klausel ORGANIZE BY DIMENSIONS verwenden.

Beispiel 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

Die Tabelle in Beispiel 1 wird nach Werten aus drei Spalten in Cluster organisiert, sodass ein logischer Würfel (mit drei Dimensionen) gebildet wird. Die Tabelle kann nun bei der Abfrageverarbeitung unter Verwendung einer oder mehrerer dieser Dimensionen logisch in Sektoren untergliedert werden, sodass nur die Blöcke in den entsprechenden Sektoren oder Zellen durch die angewendeten relationalen Operatoren verarbeitet werden. Die Größe eines Blocks (die Anzahl von Seiten) stimmt mit dem Wert für EXTENTSIZE der Tabelle überein.

MDC-Tabellen mit Dimensionen auf der Basis mehrerer Spalten

Jede Dimension kann aus einer oder mehreren Spalten bestehen. Zum Beispiel können Sie eine Tabelle erstellen, die nach einer Dimension mit zwei Spalten organisiert ist.

Beispiel 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

In Beispiel 2 wird die Tabelle in zwei Dimensionen organisiert: c1 und (c3, c4). Bei der Abfrageverarbeitung kann die Tabelle also logisch in einen Sektor nach der Dimension für c1 oder einen Sektor nach der zusammengesetzten Dimension (c3, c4) untergliedert werden. Die Tabelle enthält die gleiche Anzahl von Blöcken wie die Tabelle in Beispiel 1, jedoch einen Dimensionsblockindex weniger. In Beispiel 1 werden drei Dimensionsblockindizes erstellt: jeweils einer für jede der Spalten c1, c3 und c4. In Beispiel 2 werden zwei Dimensionsblockindizes erstellt: einer für Spalte c1 und der andere für die Spalten c3 und c4. Der Hauptunterschied zwischen diesen beiden Ansätzen besteht darin, dass in Beispiel 1 Abfragen, die die Spalte c4 betreffen, den Dimensionsblockindex für c4 nutzen können, um rasch und direkt auf Blöcke mit relevanten Daten zuzugreifen. In Beispiel 2 ist die Spalte c4 der zweite Teil eines Schlüssels in einem Dimensionsblockindex, sodass Abfragen, die die Spalte c4 betreffen, mehr Verarbeitungsaufwand erfordern. Allerdings ist in Beispiel 2 ein Blockindex weniger zu verwalten und zu speichern.

Der DB2-Designadvisor schlägt keine Empfehlungen für Dimensionen aus mehr als einer Spalte vor.

MDC-Tabellen mit Spaltenausdrücken als Dimensionen

Spaltenausdrücke können ebenfalls zum Clustering von Dimensionen verwendet werden. Die Möglichkeit, Daten nach Spaltenausdrücken in Clustern zu organisie-

ren, ist nützlich, wenn Dimensionen in eine gröbere Granularität hochgestuft werden sollen, zum Beispiel wenn eine Adresse auf eine geographische Position bzw. Region oder ein Datum auf eine Woche, Monat oder Jahr abgebildet wird. Zur Implementierung der Hochstufung von Dimensionen auf diese Weise können Sie generierte Spalten verwenden. Dieser Typ von Spaltendefinition ermöglicht die Erstellung von Spalten durch Ausdrücke, die Dimensionen darstellen können. In Beispiel 3 erstellt die Anweisung eine Tabelle, die nach einer Basisspalte und zwei Spaltenausdrücken in Clustern organisiert wird.

Beispiel 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

In Beispiel 3 ist die Spalte c5 ein Ausdruck, der auf den Spalten c3 und c4 basiert. Von der Spalte c6 wird die Spalte c1 in eine grobere Granularität der Zeit hochgestuft. Die Anweisung organisiert die Tabelle in Clustern nach den Werten in den Spalten c2, c5 und c6.

Bereichsabfragen für Dimensionen auf generierten Spalten

Bereichsabfragen für eine Dimension auf einer generierten Spalte erfordern monotone Spaltenfunktionen. Ausdrücke müssen monoton sein, damit Bereichsvergleichselemente für Dimensionen auf generierten Spalten abgeleitet werden können. Wenn Sie eine Dimension für eine generierte Spalte erstellen, können Abfragen auf die Basisspalte den Blockindex für die generierte Spalte nutzen, um die Leistung zu verbessern, allerdings mit einer Ausnahme. Für Bereichsabfragen auf die Basisspalte (z. B. Datum) muss der Ausdruck, der zur Generierung der Spalte in der Anweisung CREATE TABLE verwendet wird, monoton sein, um eine Bereichssuche im Dimensionsblockindex verwenden zu können. Obwohl ein Spaltenausdruck jeden gültigen Ausdruck (einschließlich benutzerdefinierter Funktionen - UDFs) enthalten kann, können bei nicht monotonen Ausdrücken nur Gleichheitsvergleichselemente und Vergleichselemente mit IN-Listen den Blockindex verwenden, um die Abfrage zu erfüllen, wenn sich diese Vergleichselemente auf die Basisspalte beziehen.

Angenommen, eine MDC-Tabelle mit Dimensionen wird auf der Basis der generierten Spalte für Monat mit `monat = INTEGER (datum)/100` erstellt. Für Abfragen auf die Dimension (monat) können Blockindexsuchen ausgeführt werden. Für Abfragen auf die Basisspalte (datum), können ebenfalls Blockindexsuchen ausgeführt werden, um die zu durchsuchenden Blöcke einzugrenzen. Anschließend können die Vergleichselemente für das Datum nur auf die Datensätze in diesen Blöcken angewendet werden.

Der Compiler generiert zusätzliche Vergleichselemente zur Verwendung bei der Blockindexsuche. Zum Beispiel wird die folgende Abfrage formuliert:

```
SELECT * FROM MDCTABLE WHERE DATE > "1999-03-03" AND DATE < "2000/01/15"
```

Der Compiler generiert die folgenden zusätzlichen Vergleichselemente: „month >= 199903" und „month <= 200001". Diese Vergleichselemente können zur Suche im Dimensionsblockindex verwendet werden. Beim Durchsuchen der resultierenden Blöcke werden die ursprünglichen Vergleichselemente auf die Datensätze in den Blöcken angewendet.

Ein nicht monotoner Ausdruck lässt die Anwendung von Gleichheitsvergleichselementen auf diese Dimension zu. Ein gutes Beispiel für eine nicht monotone Funktion ist die Funktion MONTH(), wie sie in der Definition von Spalte c6 in Beispiel 3 zu sehen ist. Wenn die Spalte c1 ein Datum, eine Zeitmarke oder eine gültige Zeichenfolgendarstellung eines Datums oder einer Zeitmarke ist, liefert die Funktion einen Ganzzahlwert aus dem Bereich von 1 bis 12. Obwohl die Ausgabe der Funktion deterministisch ist, ähnelt sie einer Schrittfunktion (d. h. einem zyklischen Muster):

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

Obwohl das Datum in diesem Beispiel kontinuierlich fortschreitet, tut dies der Wert der Funktion MONTH(datum) nicht. Dies bedeutet zum Beispiel, dass nicht garantiert ist, dass bei zwei Werten datum1 und datum2, wobei datum1 größer ist als datum2, der Wert von MONTH(datum1) größer als oder gleich dem Wert von MONTH(datum2) ist. Eben dies ist die Monotoniebedingung. Diese Nichteinhaltung der Monotoniebedingung ist zwar zulässig, aber sie beschränkt die Dimension in der Weise, dass ein Bereichsvergleichselement, das die Basisspalte betrifft, kein Bereichsvergleichselement für die Dimension generieren kann. Jedoch ist ein Bereichsvergleichselement für den Ausdruck problemlos möglich, wie zum Beispiel `where month(c1) between 4 and 6`. Die entsprechende Verarbeitung kann den Index für die Dimension mit einem Startschlüsselwert 4 und einem Endschlüsselwert 6 ganz normal verwenden.

Um diese Funktion monoton zu machen, schließen Sie das Jahr als übergeordnete Ordnungskomponente des Monats mit ein. Es gibt eine Erweiterung zur integrierten Funktion INTEGER(), um die Definition eines monotonen Ausdrucks für Datum zu unterstützten. INTEGER(datum) liefert eine Ganzzahldarstellung des Datums, die dann dividiert werden kann, um eine Ganzzahldarstellung des Jahres und des Monats zu ermitteln. Zum Beispiel liefert `INTEGER(date('2000/05/24'))` den Wert `20000524`. Somit ist `INTEGER(date('2000/05/24'))/100 = 200005`. Die Funktion INTEGER(datum)/100 ist monoton.

Analog verfügen auch die integrierten Funktionen DECIMAL() und BIGINT() über Erweiterungen, die es Ihnen ermöglichen, monotone Funktionen abzuleiten. Die Funktion DECIMAL(zeitmarke) liefert eine dezimale Darstellung einer Zeitmarke und ihr Wert kann in monotonen Ausdrücken zur Ableitung steigender Werte für Monat, Tag, Stunde, Minute usw. verwendet werden. Die Funktion BIGINT(datum) liefert eine BIGINTEGER-Darstellung des Datums ähnlich wie INTEGER(datum).

Der Datenbankmanager bestimmt die Monotonie eines Ausdrucks, falls möglich, bei der Erstellung der generierten Spalte für die Tabelle oder bei der Erstellung einer Dimension aus einem Ausdruck in der DIMENSIONS-Klausel. Bestimmte Funktionen können als Monotonie bewahrend erkannt werden, wie zum Beispiel DATENUM( ), DAYS( ), YEAR( ). Außerdem sind verschiedene mathematische Ausdrücke wie Division, Multiplikation oder Addition einer Spalte und einer Konstante Monotonie bewahrend. In Fällen, in denen DB2 ermittelt, dass ein Ausdruck die Monotonie nicht bewahrt, oder wenn sich dies nicht feststellen lässt, unterstützt die Dimension nur die Verwendung von Gleichheitsvergleichselementen in der entsprechenden Basisspalte.

Kapitel 12. Ändern einer Datenbank

Ändern einer Instanz

Ändern der Datenbankkonfiguration über mehrere Datenbankpartitionen

Wenn Sie eine Datenbank haben, die auf mehrere Datenbankpartitionen verteilt ist, sollte die Konfigurationsdatei für die Datenbank in allen Datenbankpartitionen die gleiche sein.

Informationen zu diesem Vorgang

Diese Konsistenz ist erforderlich, da der SQL-Compiler verteilte SQL-Anweisungen anhand der Informationen in der Datei für die Datenbankpartitionskonfiguration kompiliert und einen Zugriffsplan erstellt, der die Anforderungen der jeweiligen SQL-Anweisung erfüllt. Wenn sich verschiedene Konfigurationsdateien in den Datenbankpartitionen befinden, kann dies je nachdem, in welcher Datenbankpartition die Anweisung vorbereitet wird, zu verschiedenen Zugriffsplänen führen. Verwenden Sie den Befehl **db2_all**, um die Konfigurationsdateien in allen Datenbankpartitionen zu verwalten.

Ändern von Tabellen und anderen zugehörigen Objekten

Ändern partitionierter Tabellen

Für eine partitionierte Tabelle werden alle relevanten Klauseln der Anweisung ALTER TABLE unterstützt. Darüber hinaus ermöglicht Ihnen die Anweisung ALTER TABLE, neue Datenpartitionen hinzuzufügen, Rollins neuer Datenpartitionen (*attach*) sowie Rollouts für vorhandene Datenpartitionen (*detach*) durchzuführen.

Vorbereitende Schritte

Zum Ändern einer partitionierten Tabelle zum Zweck der Aufhebung der Zuordnung einer Datenpartition muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Der Benutzer, der die DETACH PARTITION-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Quellentabelle zu ändern (ALTER), Daten aus der Quellentabelle auszuwählen (SELECT) und die Quellentabelle zu löschen (DELETE).
- Der Benutzer muss außerdem über die Berechtigung verfügen, die zum Erstellen (CREATE) der Zieltabelle erforderlich ist. Daher müssen zum Ändern einer Tabelle für die Aufhebung der Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle beinhalten:
  - Berechtigung DBADM
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):

- Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
- Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht

Zum Ändern einer partitionierten Tabelle zu dem Zweck, eine Datenpartition zuzuordnen, müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Quellentabelle beinhalten:

- Berechtigung DATAACCESS oder Zugriffsrecht SELECT für die Quellentabelle und Berechtigung DBADM und Zugriffsrecht DROPIN für das Schema der Quellentabelle
- Zugriffsrecht CONTROL für die Quellentabelle

Zum Ändern einer partitionierten Tabelle zu dem Zweck, eine Datenpartition hinzuzufügen, müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung Zugriffsrechte zur Verwendung des Tabellenbereichs, in dem die neue Partition hinzugefügt wird, sowie mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Quellentabelle beinhalten:

- Zugriffsrecht ALTER
- Zugriffsrecht CONTROL
- Berechtigung DBADM
- Zugriffsrecht ALTERIN für das Tabellenschema

Informationen zu diesem Vorgang

- Jede Anweisung ALTER TABLE, die mit der Klausel PARTITION ausgeführt wird, muss sich einer separaten SQL-Anweisung befinden.
- In einer SQL-Anweisung mit einer Operation ALTER TABLE...PARTITION sind keine anderen ALTER-Operationen zulässig. Es ist zum Beispiel nicht möglich, in derselben SQL-Anweisung einer Tabelle eine Datenpartition zuzuordnen und eine Spalte hinzuzufügen.
- Es ist möglich, im Anschluss an die Ausführung mehrerer ALTER-Anweisungen nur eine Anweisung SET INTEGRITY auszuführen.

Vorgehensweise

Zum Ändern einer partitionierten Tabelle über die Befehlszeile geben Sie die Anweisung ALTER TABLE ein.

Richtlinien und Einschränkungen für das Ändern partitionierter Tabellen

In diesem Abschnitt werden die gängigsten ALTER TABLE-Aktionen behandelt und spezielle Aspekte im Hinblick auf zugeordnete Datenpartitionen und Datenpartitionen mit aufgehobener Zuordnung erläutert.

Die Spalte STATUS der Katalogsicht SYSCAT.DATAPARTITIONS enthält die Statusinformationen für die Partitionen einer Tabelle.

- Wenn der Wert für STATUS eine leere Zeichenfolge ist, ist die Partition sichtbar und befindet sich im normalen Status.
- Wenn der Status 'A' lautet, ist die Partition neu zugeordnet. Die Anweisung SET INTEGRITY muss abgesetzt werden, um die zugeordnete Partition in den normalen Status zu versetzen.

- Wenn der Status 'D', 'L' oder 'I' lautet, wird die Zuordnung für die Partition gerade aufgehoben, die DETACH-Operation ist jedoch noch nicht abgeschlossen.
  - Für eine Partition im Status 'D' muss die Anweisung SET INTEGRITY für alle abhängigen Tabellen mit aufgehobener Zuordnung abgesetzt werden, um die logische Zuordnung für die Partition aufzuheben.
  - Wenn sich die Partition im Status 'L' befindet, handelt es sich bei der Partition um eine Partition mit aufgehobener logischer Zuordnung. Die Task zur asynchronen Aufhebung von Partitionszuordnungen führt die DETACH-Operation für die Partition aus (ab DB2 Version 9.7 Fixpack 1).
  - Wenn sich die Partition im Status 'I' befindet, ist die Task zur asynchronen Aufhebung von Partitionszuordnungen abgeschlossen und die asynchrone Indexbereinigung aktualisiert nicht partitionierte Indizes, die für diese Partition definiert sind.

Hinzufügen oder Ändern einer Integritätsbedingung

Das Hinzufügen einer Prüfung auf Integritätsbedingung oder einer Integritätsbedingung über Fremdschlüssel wird mit Datenpartitionen unterstützt, die zugeordnet sind oder deren Zuordnung aufgehoben wurde. Wenn eine partitionierte Tabelle über zugeordnete Partitionen im Status 'D' oder 'L' verfügt, wird beim Hinzufügen einer über Primärschlüssel definierten oder eindeutigen Integritätsbedingung ein Fehler zurückgegeben, wenn das System einen neuen partitionierten Index generieren muss, um die Integritätsbedingung umzusetzen. Wenn sich eine Partition im Status 'L' befindet, gibt die Operation SQL20285N (SQLSTATE 55057) zurück. Wenn sich eine Partition im Status 'D' befindet, gibt die Operation SQL20054 (SQLSTATE 55019) zurück.

Hinzufügen einer Spalte

Wenn eine Spalte einer Tabelle mit zugeordneten Datenpartitionen hinzugefügt wird, wird die Spalte auch den zugeordneten Datenpartitionen hinzugefügt. Wenn eine Spalte einer Tabelle, die Datenpartitionen mit aufgehobener Zuordnung im Status 'I' hat, hinzugefügt wird, wird die Spalte den Datenpartitionen mit aufgehobener Zuordnung nicht hinzugefügt, weil diese Datenpartitionen der Tabelle nicht mehr physisch zugeordnet sind.

Für eine Partition mit aufgehobener Zuordnung im Status 'L' oder 'D' schlägt die Operation fehl und ein Fehler wird zurückgegeben. Wenn sich eine Partition im Status 'L' befindet, gibt die Operation SQL20285N (SQLSTATE 55057) zurück. Wenn sich eine Partition im Status 'D' befindet, gibt die Operation SQL20296N (SQLSTATE 55057) zurück.

Ändern einer Spalte

Wenn eine Spalte in einer Tabelle mit zugeordneten Datenpartitionen geändert wird, wird die Spalte auch in den zugeordneten Datenpartitionen geändert. Wenn eine Spalte in einer Tabelle, die Datenpartitionen mit aufgehobener Zuordnung hat, geändert wird, wird die Spalte in den Datenpartitionen mit aufgehobener Zuordnung nicht geändert, weil diese Datenpartitionen der Tabelle nicht mehr physisch zugeordnet sind.

Wenn eine Spalte gelöscht oder umbenannt wird und sich die Partition mit aufgehobener Zuordnung im Status 'L' oder 'D' befindet, schlägt die Operation fehl und es wird ein Fehler zurückgegeben. Wenn sich eine Partition im Status 'L' befindet, gibt die Operation SQL20285N (SQLSTATE 55057) zurück. Wenn sich eine Partition im Status 'D' befindet, gibt die Operation SQL0270N (SQLSTATE 42997) zurück.

Hinzufügen einer generierten Spalte

Wenn eine generierte Spalte einer partitionierten Tabelle mit zugeordneten

Datenpartitionen oder Datenpartitionen mit aufgehobener Zuordnung hinzugefügt wird, müssen dabei die Regeln zum Hinzufügen aller anderen Typen von Spalten berücksichtigt werden.

Hinzufügen oder Ändern eines nicht partitionierten Indexes
Wenn ein Index für eine Tabelle mit hinzugefügten Datenpartitionen erstellt, erneut erstellt oder reorganisiert wird, schließt der Index die Daten in den zugeordneten Datenpartitionen nicht mit ein, weil alle Indizes für sämtliche zugeordneten Datenpartitionen durch die Anweisung SET INTEGRITY gewartet werden. Wenn ein Index für eine Tabelle mit Datenpartitionen, deren Zuordnung aufgehoben ist, erstellt, erneut erstellt oder reorganisiert wird, schließt der Index die Daten der Datenpartitionen mit aufgehobener Zuordnung nur dann ein, wenn eine Datenpartition mit aufgehobener Zuordnung über freigegebene abhängige Tabellen oder Zwischenspeichertabellen verfügt, die im Hinblick auf die Datenpartition inkrementell aktualisiert werden müssen; die Partition befindet sich im Status 'D'. In diesem Fall enthält der Index die Daten für diese Datenpartition mit aufgehobener Zuordnung.

Hinzufügen oder Ändern eines partitionierten Indexes

Wenn bei Vorhandensein zugeordneter Datenpartitionen ein partitionierter Index erstellt wird, wird für jede zugeordnete Datenpartition eine Indexpartition erstellt. Die Indexeinträge für Indexpartitionen für zugeordnete Datenpartitionen werden erst sichtbar, nachdem Sie die Anweisung SET INTEGRITY ausgeführt haben, um die zugeordneten Datenpartitionen online zur Verfügung zu stellen. Hinweis: Da beim Erstellen eines Indexes die zugeordneten Datenpartitionen einbezogen werden, werden beim Erstellen eindeutiger partitionierter Indizes möglicherweise Zeilen in der zugeordneten Datenpartition gefunden, die doppelte Schlüsselwerte sind und keine erfolgreiche Indexerstellung ermöglichen. Es wird deshalb empfohlen, bei Vorhandensein zugeordneter Partitionen keine partitionierten Indizes zu erstellen, um dieses Problem zu vermeiden.

Wenn die Tabelle über abhängige Tabellen mit aufgehobener Zuordnung verfügt, wird die Erstellung partitionierter Indizes nicht für partitionierte Tabellen mit abhängigen Tabelle mit aufgehobener Zuordnung nicht unterstützt. Jeder Versuch, in dieser Situation einen partitionierten Index zu erstellen, verursacht SQLSTATE 55019. Wenn ein partitionierter Index für eine Tabelle mit Partitionen im Status 'L' erstellt wird, gibt die Operation SQL20285N (SQLSTATE 55057) zurück.

WITH EMPTY TABLE
Sie können eine Tabelle mit zugeordneten Datenpartitionen nicht leeren.

ADD MATERIALIZED QUERY AS
Das Ändern einer Tabelle mit zugeordneten Datenpartitionen in eine MQT ist nicht zulässig.

Ändern zusätzlicher Tabellenattribute, die in einer Datenpartition gespeichert sind
Die folgenden Tabellenattribute werden zusätzlich in einer Datenpartition gespeichert. Änderungen an diesen Attributen werden auf die zugeordneten Datenpartitionen, jedoch nicht auf die Datenpartitionen mit aufgehobener Zuordnung übertragen.

- DATA CAPTURE
- VALUE COMPRESSION
- APPEND
- COMPACT/LOGGED FOR LOB COLUMNS

Erstellen von und Zugreifen auf Datenpartitionen innerhalb einer Transaktion
Wenn eine Tabelle über einen nicht partitionierten Index verfügt, ist der Zugriff auf eine neue Datenpartition in dieser Tabelle innerhalb derselben Transaktion (z. B. die ADD- oder ATTACH-Operation, die die Partition erstellt hat), nicht möglich, wenn die Transaktion das Sperren der Tabelle im Exklusivmodus nicht veranlasst (SQL0668N, Ursachencode 11).

Besondere Hinweise für XML-Indizes beim Ändern einer Tabelle zum Hinzufügen, Zuordnen oder Aufheben der Zuordnung einer Partition

Ähnlich wie bei einem nicht partitionierten relationalen Index ist ein nicht partitionierter Index zu einer XML-Spalte ein unabhängiges Objekt, das von allen Datenpartitionen einer partitionierten Tabelle gemeinsam genutzt wird. Das Hinzufügen, Zuordnen oder Aufheben der Zuordnung einer Partition beim Ändern einer Tabelle wirkt sich auf XML-Regionsindizes und -Spaltenpfadindizes aus. Indizes zu XML-Spaltenpfaden sind stets nicht partitioniert und Indizes zu XML-Daten werden standardmäßig partitioniert erstellt.

XML-Regionsindex

Mit ADD PARTITION wird eine neue Regionsindexpartition für die neue leere Datenpartition erstellt, die hinzugefügt wird. Ein neuer Eintrag für die Regionsindexpartition wird zur Tabelle SYSINDEXPARTITIONS hinzugefügt. Der Tabellenbereich für das partitionierte Indexobjekt für die neue Partition wird durch INDEX IN <tabellenbereich> in der Klausel ADD PARTITION bestimmt. Wird die Option INDEX IN <tabellenbereich> nicht in der Klausel ADD PARTITION angegeben, wird für das partitionierte Indexobjekt standardmäßig derselbe Tabellenbereich verwendet wie für die entsprechende Datenpartition.

Der systemgenerierte XML-Regionsindex für eine partitionierte Tabelle ist immer partitioniert. Ein partitionierter Index verwendet ein Indexorganisationsschema, bei dem Indexdaten über mehrere Speicherobjekte, die als Indexpartitionen bezeichnet werden, entsprechend dem Tabellenpartitionierungsschema der Tabelle verteilt werden. Jede Indexpartition bezieht sich ausschließlich auf Tabellenzeilen in der entsprechenden Datenpartition.

Da der Regionsindex für eine partitionierte Tabelle mit XML-Spalte immer partitioniert ist, kann der Regionsindex für die Quellentabelle bei ATTACH nach Abschluss der ATTACH-Operation als neue Regionsindexpartition für die neue Tabellenpartition beibehalten werden. Daten und Indexobjekte werden nicht versetzt; deshalb müssen die Katalogtabelleneinträge aktualisiert werden. Der Katalogtabelleneintrag für den Regionsindex für die Quellentabelle wird bei ATTACH entfernt und eine Regionsindexpartition wird in Tabelle SYSINDEXPARTITIONS hinzugefügt. Die Pool-ID und Objekt-ID ändern sich gegenüber der Quellentabelle nicht. Die Index-ID (IID) wird geändert, damit sie mit der Index-ID des Regionsindexes für die Zieltabelle übereinstimmt.

Nach Abschluss der DETACH-Operation wird der Regionsindex für die Tabelle mit aufgehobener Zuordnung beibehalten. Der Indexpartitionseintrag der Partition, deren Zuordnung aufgehoben wird, wird aus Tabelle SYSINDEXPARTITIONS entfernt. Ein neuer Regionsindexeintrag wird in der Katalogtabelle SYSINDEXES für die Tabelle mit aufgehobener Zuordnung hinzugefügt, die dieselbe Pool-ID und Objekt-ID wie die Regionsindexpartition vor der DETACH-Operation erhält.

Index zu XML-Daten

Ab DB2 Version 9.7 Fixpack 1 können Sie einen Index zu XML-Daten in einer partitionierten Tabelle als partitioniert oder nicht partitioniert erstellen. Standardmäßig wird ein partitionierter Index erstellt.

Partitionierte und nicht partitionierte Indizes zu XML-Daten werden bei ATTACH- und DETACH-Operationen genauso wie andere relationale Indizes behandelt.

Indizes für die Quellentabelle werden während der ATTACH-Operation gelöscht. Dies betrifft sowohl logische als auch physische XML-Indizes. Ihre Einträge in den Systemkatalogen werden während der ATTACH-Operation entfernt.

SET INTEGRITY muss nach der ATTACH-Operation ausgeführt werden, um die nicht partitionierten Indizes zu XML-Daten für die Zieltabelle zu warten.

Bei DETACH werden nicht partitionierte Indizes zu XML-Spalten für die Quellentabelle nicht von der Zieltabelle übernommen.

XML-Spaltenpfadindizes

Indizes zu XML-Spaltenpfaden sind stets nicht partitionierte Indizes. Die XML-Spaltenpfadindizes für die Quellen- und Zieltabellen werden während der Rollin- und Rollout-Operationen gewartet.

Bei ATTACH wartet der DB2-Datenbankmanager die nicht partitionierten XML-Spaltenpfadindizes für die Zieltabelle (dies ist anders als bei anderen nicht partitionierten Indizes, die während der Ausführung von SET INTEGRITY nach Abschluss der ATTACH-Operation gewartet werden). Danach werden die XML-Spaltenpfadindizes für die Quellentabelle gelöscht und ihre Katalogeinträge werden entfernt, weil die Spaltenpfadindizes für die Zieltabelle nicht partitioniert sind.

Beim Rollout ist zu beachten, dass die XML-Spaltenpfadindizes nicht partitioniert sind. Und nicht partitionierte Indizes werden nicht in die eigenständige Zieltabelle übertragen. Doch müssen XML-Spaltenpfadindizes (einer für jede Spalte) für eine Tabelle mit XML-Spalten bestehen, bevor externe Benutzer auf die Tabelle zugreifen können. Deshalb müssen XML-Spaltenpfadindizes für die Zieltabelle erstellt werden, bevor sie verwendet werden kann. Der Zeitpunkt der Erstellung der Spaltenpfadindizes hängt davon ab, ob es während der DETACH-Operation abhängige Tabellen mit aufgehobener Zuordnung gibt. Wenn es keine abhängigen Tabellen mit aufgehobener Zuordnung gibt, werden die Pfadindizes während der DETACH-Operation erstellt. Andernfalls werden sie von SET INTEGRITY oder der MQT-Aktualisierung erstellt, um die abhängigen Objekte mit aufgehobener Zuordnung zu warten.

Nach Abschluss der DETACH-Operation befinden sich die für die Zieltabelle erstellten XML-Spaltenpfadindizes in demselben Indexobjekt wie alle anderen Indizes für diese Tabelle.

Zuordnen von Datenpartitionen

Die Partitionierung von Tabellen ermöglicht eine effiziente Ein- und Auslagerung von Tabellendaten (engl. Rollin und Rollout). Die Anweisung ALTER TABLE mit der Klausel ATTACH PARTITION vereinfacht den Rollin von Daten.

Vorbereitende Schritte

Wenn die Prüfung der Datenintegrität (einschließlich Bereichsprüfung und Prüfung anderer Integritätsbedingungen) durch vom Datenserver unabhängige Anwendungslogik vor einer Zuordnungsoperation erfolgen kann, können die neu zugeordneten Daten deutlich früher verfügbar gemacht werden. Sie können den Prozess für Dateneinlagerung optimieren, indem Sie mithilfe der Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED die Bereichsprüfung und die Prüfung auf ungültige Integritätsbedingungen überspringen. In diesem Fall verlässt die Tabelle den Status 'Festlegen der Integrität anstehend' und die neuen Daten sind sofort für Anwendungen verfügbar, sofern es sich bei allen Benutzerindizes der Zieltabelle um partitionierte Indizes handelt.

Wenn sich in der Tabelle nach einer Zuordnungsoperation nicht partitionierte Indizes befinden, die gepflegt werden müssen (mit Ausnahme von XML-Spaltenpfadindizes), verhält sich die Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED wie eine Anweisung vom Typ SET INTEGRITY...IMMEDIATE CHECKED. Die gesamte Integritätsverarbeitung, die Verwaltung nicht partitionierter Indizes und die Tabellenstatusübergänge werden so ausgeführt, als sei eine Anweisung SET INTEGRITY...IMMEDIATE CHECKED ausgegeben worden. Dadurch wird sichergestellt, dass ein Rollin-Script, das mit der Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED arbeitet, seine Arbeit bei der Erstellung eines nicht partitionierten Index für die Zieltabelle nicht irgendwann nach seiner Inbetriebnahme einstellt.

Zum Ändern einer Tabelle zu dem Zweck, eine Datenpartition zuzuordnen, müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen oder eines der folgenden Zugriffsrechte für die Quellentabelle beinhalten:

- Zugriffsrecht SELECT für die Tabelle und Zugriffsrecht DROPIN für das Schema der Tabelle
- Zugriffsrecht CONTROL für die Tabelle
- Berechtigung DATAACCESS

Informationen zu diesem Vorgang

Beim Zuordnen von Datenpartitionen wird eine vorhandene Tabelle (Quellentabelle) genommen und als neue Datenpartition der Zieltabelle zugeordnet. Wenn eine Datenpartition mithilfe der Anweisung ALTER TABLE und der Klausel ATTACH PARTITION an eine partitionierte Tabelle angehängt wird, bleibt die partitionierte Zieltabelle online und dynamische Abfragen, die für die Tabelle mit der Isolationsstufe 'Lesestabilität', 'Cursorstabilität' oder 'Nicht festgeschriebener Lesevorgang' ausgeführt werden, werden fortgesetzt.

Einschränkungen und Verwendungshinweise

Die folgenden Bedingungen müssen erfüllt sein, bevor Sie eine Datenpartition zuordnen können:

- Die Zieltabelle, der Sie die neue Datenpartition zuordnen wollen, muss eine vorhandene partitionierte Tabelle sein.
- Die Quellentabelle muss eine vorhandene, nicht partitionierte Tabelle oder eine partitionierte Tabelle mit einer einzigen Datenpartition sein, der keine Datenpartitionen zugeordnet bzw. von der keine Datenpartitionen getrennt wurden. Zur Zuordnung mehrerer Datenpartitionen ist es erforderlich, mehrere ATTACH-Anweisungen abzusetzen.

- Die Quellentabelle darf keine typisierte Tabelle sein.
- Die Quellentabelle darf keine Bereichsclustertabelle sein.
- Die Quellen- und Zieltabellendefinitionen müssen übereinstimmen.
- Die Anzahl, der Typ und die Anordnung von Quellen- und Zieltabellenspalten müssen übereinstimmen.
- Die Quellen- und Zieltabellenspalten müssen in der Hinsicht übereinstimmen, ob sie Standardwerte enthalten.
- Die Quellen- und Zieltabellenspalten müssen in der Hinsicht übereinstimmen, ob sie Nullwerte zulassen.
- Die Quellen- und Zieltabellenangaben zur Komprimierung müssen übereinstimmen, dies schließt auch die Klauseln VALUE COMPRESSION und COMPRESS SYSTEM DEFAULT ein.
- Die Quellen- und Zieltabellenspezifikationen für die Optionen DATA CAPTURE, ACTIVATE NOT LOGGED INITIALLY und APPEND müssen übereinstimmen.
- Die Zuordnung einer Datenpartition ist zulässig, selbst wenn eine Zielspalte eine generierte Spalte und die zugehörige Quellenspalte keine generierte Spalte ist. Die folgende Anweisung generiert die Werte für die generierte Spalte der zugeordneten Zeilen:

  ```
  SET INTEGRITY FOR tabellenname
    ALLOW WRITE ACCESS  IMMEDIATE CHECKED FORCE GENERATED
  ```

  Die Quellentabellenspalte, die einer generierten Spalte entspricht, muss in Typ und Optionalität der Dateneingabe übereinstimmen; es ist jedoch kein Standardwert erforderlich. Das empfohlene Verfahren besteht darin, sicherzustellen, dass die Quellentabelle der ATTACH-Operation den korrekten generierten Wert in der generierten Spalte enthält. Wenn Sie die empfohlene Vorgehensweise befolgen, müssen Sie die Option FORCE GENERATED nicht verwenden und die nachfolgenden Anweisungen können verwendet werden.

  ```
  SET INTEGRITY FOR tabellenname
    GENERATED COLUMN    IMMEDIATE UNCHECKED
  ```

  Die Anweisung gibt an, dass die Prüfung der generierten Spalte übergangen werden soll.

  ```
  SET INTEGRITY FOR tabellenname
    ALLOW WRITE ACCESS  IMMEDIATE CHECKED  FOR EXCEPTION IN tabellenname USE tabellenname
  ```

  Diese Anweisung führt eine Integritätsprüfung der zugeordneten Datenpartition aus, überprüft jedoch nicht die generierte Spalte.
- Die Zuordnung einer Datenpartition ist zulässig, selbst wenn die Zielspalte eine Identitätsspalte und die Quellenspalte keine Identitätsspalte ist. Die Anweisung SET INTEGRITY IMMEDIATE CHECKED generiert keine Identitätswerte für die zugeordneten Zeilen. Die Anweisung SET INTEGRITY FOR T GENERATE IDENTITY ALLOW WRITE ACCESS IMMEDIATE CHECKED fügt die Identitätswerte für die zugeordneten Zeilen ein. Die Spalte, die einer Identitätsspalte entspricht, muss in Typ und Optionalität der Dateneingabe übereinstimmen. Die Standardwerte dieser Spalte unterliegen keiner bestimmten Anforderung. Das empfohlene Verfahren besteht darin, dass die korrekten Identitätswerte in die Zwischenspeichertabelle einzufügen. In diesem Fall brauchen Sie im Anschluss an die ATTACH-Zuordnung die Option GENERATE IDENTITY nicht zu verwenden, weil die Identitätswerte in der Quellentabelle bereits garantiert sind.
- Bei Tabellen, deren Daten über mehrere Datenbankpartitionen verteilt sind, muss die Quellentabelle ebenfalls in derselben Datenbankpartitionsgruppe mit demselben Verteilungsschlüssel und derselben Verteilungszuordnung verteilt sein.

- Die Quellentabelle muss löschbar sein (d. h., RESTRICT DROP darf für sie nicht angegeben sein).
- Wenn ein Datenpartitionsname angegeben wird, darf dieser in der Zieltabelle nicht bereits vorhanden sein.
- Wenn die Zieltabelle eine MDC-Tabelle (MDC, Multi-Dimensional Clustering) ist, muss die Quellentabelle ebenfalls eine MDC-Tabelle sein.
- Bei Verwendung einer nicht partitionierten Tabelle muss der Datentabellenbereich für die Quellentabelle mit den Datentabellenbereichen für die Zieltabelle bezüglich Typ (d. h. DMS oder SMS), Seitengröße, Speicherbereichsgröße und Datenbankpartitionsgruppe übereinstimmen. Wenn eine Abweichung in Bezug auf die Vorablesezugriffsgröße vorliegt, wird eine Warnung ausgegeben. Der Indextabellenbereich für die Quellentabelle muss mit den Indextabellenbereichen, die von den partitionierten Indizes für die Zieltabelle verwendet werden, bezüglich Typ, Datenbankpartitionsgruppe sowie Seitengröße und Speicherbereichsgröße übereinstimmen. Der Tabellenbereich für große Objektdaten der Quellentabelle muss mit den Tabellenbereichen für große Objektdaten der Zieltabelle bezüglich Typ, Datenbankpartitionsgruppe und Seitengröße übereinstimmen. Bei Verwendung einer partitionierten Tabelle muss der Datentabellenbereich für die Quellentabelle mit den Datentabellenbereichen für die Zieltabelle bezüglich Typ, Seitengröße, Speicherbereichsgröße und Datenbankpartitionsgruppe übereinstimmen.
- Wenn Sie die Anweisung ALTER TABLE ATTACH für eine partitionierte Tabelle mit beliebigen strukturierten, XML- oder LOB-Spalten absetzen, muss der Wert für INLINE LENGTH beliebiger strukturierter, XML- oder LOB-Spalten der Quellentabelle mit dem Wert für INLINE LENGTH der entsprechenden strukturierten, XML- oder LOB-Spalten der Zieltabelle übereinstimmen.
- Wenn die Klausel REQUIRE MATCHING INDEXES zusammen mit der Klausel ATTACH PARTITION verwendet wird und falls es partitionierte Indizes für die Zieltabelle gibt, die keine Übereinstimmung in der Quellentabelle haben, wird SQL20307N zurückgegeben.
- Der Versuch, eine Quellentabelle zuzuordnen, die keinen übereinstimmenden Index für jeden partitionierten eindeutigen Index der Zieltabelle hat, schlägt mit Fehler SQL20307N (Ursachencode 17) fehl.
- Angenommen, für eine Tabelle wird gerade als Folge eines MDC-Rollouts eine verzögerte Indexbereinigung ausgeführt. Da die Nutzung der verzögerten Indexbereinigung bei einem MDC-Rollout für partitionierte Indizes nicht unterstützt wird, wird die ATTACH-Operation nicht zugelassen, wenn RID-Indizes für die Quellentabelle vorhanden sind, die während der ATTACH-Operation beibehalten (nicht erneut erstellt) werden und für die eine asynchrone Indexbereinigung der Blöcke ansteht, für die das Rollout durchgeführt wurde.
- Das Zuordnen einer Quellentabelle mit einem XML-Datenformat, das nicht mit dem der Zieltabelle übereinstimmt, wird nicht unterstützt.
- Wenn eine Tabelle XML-Spalten enthält, die das XML-Datensatzformat von Version 9.5 oder früher verwenden, ist es nicht möglich, die Tabelle einer partitionierten Tabelle zuzuordnen, die XML-Spalten enthält, die das Datensatzformat von Version 9.7 oder später verwenden.

  Vor dem Zuordnen der Tabelle müssen Sie das XML-Datensatzformat der Tabelle ändern, sodass es mit dem Datensatzformat der partitionierten Zieltabelle übereinstimmt. Mit den beiden folgenden Verfahren kann das XML-Datensatzformat einer Tabelle aktualisiert werden:

  – Ausführen einer Onlineoperation zum Versetzen von Tabellen mit der Prozedur ADMIN_MOVE_TABLE
  – Ausführen der folgenden Schritte:

1. Erstellung einer Kopie der Tabellendaten mit dem Befehl EXPORT.
2. Verwendung der Anweisung TRUNCATE zum Löschen aller Zeilen aus der Tabelle und Freigabe des Speichers, der der Tabelle zugeordnet ist.
3. Verwendung des Befehls LOAD, um die Daten in die Tabelle zu laden.

Ordnen Sie die Tabelle der partitionierten Zieltabelle zu, nachdem das XML-Datensatzformat der Tabelle aktualisiert wurde.

- Wenn eine Tabelle über einen nicht partitionierten Index verfügt, ist der Zugriff auf eine neue Datenpartition in dieser Tabelle innerhalb derselben Transaktion (z. B. die ADD- oder ATTACH-Operation, die die Partition erstellt hat), nicht möglich, wenn die Transaktion das Sperren der Tabelle im Exklusivmodus nicht veranlasst (SQL0668N, Ursachencode 11).

Bevor die ATTACH-Operation ausgeführt wird, erstellen Sie Indizes für die Quellentabelle, die mit den partitionierten Indizes in der Zieltabelle übereinstimmen. Wenn die partitionierten Indizes dann übereinstimmen, kann die Rollin-Operation effizienter ausgeführt werden und es wird weniger Speicher für die aktiven Protokolldateien benötigt. Wenn die Indizes für die Quellentabelle nicht ordnungsgemäß vorbereitet sind, muss der Datenbankmanager sie für Sie warten. Um sicherzustellen, dass Ihr Rollin keinen zusätzlichen Aufwand zur Wartung der partitionierten Indizes verursacht, können Sie REQUIRE MATCHING INDEXES für die Operation zum Zuordnen der Partition angeben. Die Angabe von REQUIRE MATCHING INDEXES stellt sicher, dass die ATTACH-Operation fehlschlägt, wenn eine Quellentabelle nicht über Indizes verfügt, die mit den partitionierten Indizes der Zieltabelle übereinstimmen. Sie können dann die erforderlichen Korrekturmaßnahmen durchführen und die ATTACH-Operation erneut ausführen.

Löschen Sie ferner alle zusätzlichen Indizes für die Quellentabelle, bevor Sie die ATTACH-Operation ausführen. Unter zusätzlichen Indizes sind bei der Quellentabelle solche zu verstehen, für die es entweder keine Übereinstimmung bei der Zieltabelle gibt oder die nicht partitionierten Indizes der Zieltabelle entsprechen. Durch Löschen der zusätzlichen Indizes wird die Ausführung der ATTACH-Operation beschleunigt.

Beispiel: Angenommen, es gibt eine partitionierte Tabelle mit dem Namen ORDERS, die 12 Datenpartitionen enthält (eine für jeden Monat des Jahres). Am Ende jedes Monats wird eine separate Tabelle mit dem Namen NEWORDERS der partitionierten Tabelle ORDERS zugeordnet.

1. Erstellen Sie partitionierte Indizes für die Tabelle ORDERS.
   ```
   CREATE INDEX idx_delivery_date ON orders(delivery) PARTITIONED
   CREATE INDEX idx_order_price ON orders(price) PARTITIONED
   ```
2. Bereiten Sie die ATTACH-Operation vor, indem Sie die entsprechenden Indizes für die Tabelle NEWORDERS erstellen.
   ```
   CREATE INDEX idx_delivery_date_for_attach ON neworders(delivery)
   CREATE INDEX idx_order_price_for_attach ON neworders(price)
   ```
3. Die ATTACH-Operation setzt sich aus zwei Schritten zusammen:
   a. ATTACH. Die Indizes für die Tabelle NEWORDERS, die mit den partitionierten Indizes für Tabelle ORDERS übereinstimmen, werden beibehalten.
      ```
      ALTER TABLE orders ATTACH PARTITION part_jan2009 STARTING FROM ('01/01/2009')
         ENDING AT ('01/31/2009') FROM TABLE neworders
      ```

      Die Tabelle ORDERS wird automatisch in den Status 'Festlegen der Integrität anstehend' versetzt. Sowohl der Index idx_delivery_date_for_attach als auch der Index idx_order_price_for_attach werden nach Abschluss der AT-

TACH-Operation in die Tabelle ORDERS aufgenommen. Während dieser Operation erfolgt kein Versetzen von Daten.

b. SET INTEGRITY. Für die neu zugeordnete Partition wird eine Bereichsprüfung durchgeführt. Alle vorhandenen Integritätsbedingungen werden umgesetzt. Nach Abschluss werden die neu zugeordneten Daten in der Datenbank sichtbar.

```
SET INTEGRITY FOR orders IMMEDIATE CHECKED
```

Wenn nicht partitionierte Indizes für die Zieltabelle bestehen, muss die Anweisung SET INTEGRITY die Indexwartung zusammen mit anderen Aufgaben wie Bereichsüberprüfung und Prüfung von Integritätsbedingungen für die Daten aus der neu zugeordneten Partition durchführen. Die Wartung nicht partitionierter Indizes erfordert viel Speicherplatz für die aktiven Protokolldateien, der den Datenvolumen in der neu zugeordneten Partition proportional entspricht. Außerdem wird die Schlüsselgröße jedes nicht partitionierten Index sowie die Anzahl der nicht partitionierten Indizes benötigt.

Jeder partitionierte Index für die neue Datenpartition erhält einen Eintrag in der Katalogtabelle SYSINDEXPARTITIONS, wobei die Tabellenbereichs- und die Objektkennung der Quellentabelle verwendet werden. Die Kennungsinformationen werden entweder der Tabelle SYSINDEXES (wenn die Tabelle nicht partitioniert ist) oder der Tabelle SYSINDEXPARTITIONS (wenn die Tabelle partitioniert ist) entnommen. Die Index-ID wird dem partitionierten Index der übereinstimmenden Zieltabelle entnommen.

Wenn die Quellentabelle partitioniert ist, werden die partitionierten Indizes der Quellentabelle, die mit den partitionierten Indizes der Zieltabelle übereinstimmen, im Rahmen der ATTACH-Operation beibehalten. Indexpartitionseinträge in der Tabelle SYSINDEXPARTITIONS werden aktualisiert, um anzuzeigen, dass es sich bei ihnen um Indexpartitionen in der neuen Zieltabelle mit neuen Index-IDs handelt.

Beim Zuordnen von Datenpartitionen werden bestimmte Statistikdaten für Indizes sowie Daten für die neue Partition von der Quellentabelle in die Zieltabelle übertragen. Konkret werden alle Felder in den Tabellen SYSDATAPARTITIONS und SYSINDEXPARTITIONS für die neue Partition der Zieltabelle anhand von Daten aus der Quellentabelle gefüllt. Wenn die Quellentabelle nicht partitioniert ist, werden diese Statistikdaten den Tabellen SYSTABLES und SYSINDEXES entnommen. Wenn die Quellentabelle eine partitionierte Tabelle mit einer Partition ist, werden diese Statistikdaten den Tabellen SYSDATAPARTITIONS und SYSINDEXPARTITIONS der einzelnen Quellenpartition entnommen.

Anmerkung: Führen Sie eine RUNSTATS-Operation nach Abschluss der ATTACH-Operation durch, weil sich die übertragenen Statistikdaten nicht auf die zusammengefassten Statistikdaten in den Tabellen SYSINDEXES und SYSTABLES auswirken.

Pflege von nicht partitionierten Indizes bei SET INTEGRITY...ALL IMMEDIATE UNCHECKED. Wenn SET INTEGRITY...ALL IMMEDIATE UNCHECKED für eine partitionierte Tabelle abgesetzt wird, um die Bereichsprüfung für eine neu zugeordnete Partition zu überspringen, sofern in der Tabelle nicht partitionierte Indizes vorhanden sind (mit Ausnahme des XML-Spaltenpfadindex), wird SET INTEGRITY...ALL IMMEDIATE UNCHECKED wie folgt ausgeführt:

- Wenn die Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED eine einzige Zieltabelle referenziert, dann entspricht das Systemverhalten dem Verhalten, das auftritt, wenn stattdessen eine Anweisung SET INTEGRITY...ALLOW WRITE ACCESS...IMMEDIATE CHECKED abgesetzt wird. Mit der Anweisung

SET INTEGRITY...ALL IMMEDIATE UNCHECKED werden alle nicht partitionierten Indizes (mit der Ausnahme von XML-Spaltenpfadindizes) gepflegt, jede weitere Integritätsverarbeitung durchgeführt, die Flagwerte für die Überprüfung der Integritätsbedingungen in der Spalte CONST_CHECKED in der Katalogsicht SYSCAT.TABLES aktualisiert sowie Fehler zurückgegeben; außerdem wird die Ausführung unmittelbar gestoppt, wenn Verstöße gegen Integritätsbedingungen festgestellt werden.

- Wenn die Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED mehr als eine Zieltabelle referenziert, wird ein Fehler zurückgegeben (SQL20209N mit dem Ursachencode 13).

Erneutes Erstellen ungültiger partitionierter Indizes bei SET INTEGRITY. Mithilfe der Anweisung SET INTEGRITY kann ermittelt werden, ob das partitionierte Indexobjekt für eine neu zugeordnete Partition ungültig ist; außerdem kann mit dieser Anweisung ein partitionierter Index erneut erstellt werden, falls dies notwendig ist.

Richtlinien zum Hinzufügen von Datenpartitionen zu partitionierten Tabellen

In diesem Abschnitt finden Sie Richtlinien für die Korrektur verschiedener Typen von Merkmalsabweichungen, die auftreten können, wenn Sie versuchen, eine Datenpartition einer partitionierten Tabelle mithilfe der Anweisung ALTER TABLE ...ATTACH PARTITION zuzuordnen. Sie können eine Übereinstimmung zwischen Tabellen erreichen, indem Sie die Quellentabelle an den Merkmalen der Zieltabelle ausrichten oder indem Sie die Zieltabelle an den Merkmalen der Quellentabelle ausrichten.

Die Quellentabelle ist die vorhandene Tabelle, der Sie die Zieltabelle zuordnen wollen. Die Zieltabelle ist die Tabelle, der Sie die neue Datenpartition zuordnen wollen.

Ein vorgeschlagenes Verfahren für eine erfolgreiche Zuordnung ist die Anwendung genau jener Anweisung CREATE TABLE, die Sie für die Zieltabelle verwendet haben, auf die Quellentabelle, jedoch ohne die Klausel PARTITION BY. In Fällen, in denen es schwierig ist, die Merkmale entweder der Quellentabelle oder der Zieltabelle zu modifizieren und Kompatibilität zu erreichen, können Sie eine neue Quellentabelle erstellen, die mit der Zieltabelle kompatibel ist. Details zum Erstellen einer neuen Quelle finden Sie in „Erstellen von Tabellen nach vorhandenen Tabellen".

Informationen zur Vermeidung abweichender Merkmale finden Sie im Abschnitt mit den Einschränkungen und Verwendungsrichtlinien in „Zuordnen von Datenpartitionen". Der Abschnitt enthält einen Überblick über die Bedingungen, die erfüllt sein müssen, bevor Sie eine Datenpartition erfolgreich zuordnen können. Wenn die aufgeführten Bedingungen nicht erfüllt sind, wird der Fehler SQL20408N oder SQL20307N zurückgegeben.

In den folgenden Abschnitten werden die verschiedenen Typen von Abweichungen beschrieben, die auftreten können, und Schritte zur Herstellung übereinstimmender Merkmale zwischen Tabellen vorgeschlagen:

Die Klausel (VALUE) COMPRESSION (Spalte COMPRESSION in SYSCAT.TABLES) stimmt nicht überein. (SQL20307N Ursachencode 2)

Verwenden Sie eine der folgenden Anweisungen, um Übereinstimmung in der Wertkomprimierung herzustellen:

```
ALTER TABLE... ACTIVATE VALUE COMPRESSION
oder
ALTER TABLE... DEACTIVATE VALUE COMPRESSION
```

Verwenden Sie eine der folgenden Anweisungen, um Übereinstimmung in der Zeilenkomprimierung herzustellen:

```
ALTER TABLE... COMPRESS YES
oder
ALTER TABLE... COMPRESS NO
```

Der Modus APPEND der Tabellen stimmt nicht überein. (SQL20307N Ursachencode 3)

Verwenden Sie eine der folgenden Anweisungen, um Übereinstimmung im APPEND-Modus herzustellen:

```
ALTER TABLE ... APPEND ON
oder
ALTER TABLE ... APPEND OFF
```

Die Codepages der Quellen- und Zieltabelle stimmen nicht überein. (SQL20307N Ursachencode 4)

Erstellen Sie eine neue Quellentabelle.

Die Quellentabelle ist eine partitionierte Tabelle mit mehreren Datenpartitionen oder mit Datenpartitionen, die zugeordnet wurden bzw. deren Zuordnung aufgehoben wurde. (SQL20307N Ursachencode 5)

Heben Sie mit der folgenden Anweisung so viele Zuordnungen von Datenpartitionen zur Quellentabelle auf, dass nur eine sichtbare Datenpartition vorhanden ist:

```
ALTER TABLE ... DETACH PARTITION
```

Partitionen, deren Zuordnung aufgehoben wurde, bleiben in diesem Status, bis die folgenden Schritte ausgeführt wurden:

1. Ausführen aller erforderlichen Anweisungen SET INTEGRITY für die schrittweise Aktualisierung von abhängigen Objekten mit aufgehobener Zuordnung.
2. Ab Version 9.7.1: Warten auf die asynchrone Beendigung des Aufhebens der Zuordnung. Um diesen Prozess zu beschleunigen, stellen Sie sicher, dass der gesamte Zugriff auf die Tabelle, der vor dem Aufheben der Zuordnung begonnen hat, abgeschlossen oder beendet wird.
3. Bei Quellentabellen mit nicht partitionierten Indizes: Warten, bis die asynchrone Indexbereinigung abgeschlossen ist. Eine Option zur Beschleunigung dieses Prozesses kann darin bestehen, die nicht partitionierten Indizes in der Quellentabelle zu löschen.

Wenn Sie eine Zuordnungsoperation sofort ausführen möchten, haben Sie auch die Möglichkeit, eine neue Quellentabelle zu erstellen.

Die Quellentabelle ist eine Systemtabelle, eine Sicht, eine typisierte Tabelle, eine mit ORGANIZED BY KEY SEQUENCE definierte Tabelle, eine erstellte temporäre Tabelle oder eine deklarierte temporäre Tabelle. (SQL20307N Ursachencode 6)

Erstellen Sie eine neue Quellentabelle.

Die Zieltabelle und die Quellentabelle sind identisch. (SQL20307N Ursachencode 7)

Sie können keine Tabelle sich selbst zuordnen. Bestimmen Sie die richtige Tabelle, die als Quellen- oder Zieltabelle zu verwenden ist.

Die Klausel NOT LOGGED INITIALLY wurde für die Quellentabelle oder für die Zieltabelle angegeben, jedoch nicht für beide. (SQL20307N Ursachencode 8)

Machen Sie die Tabelle, die mit der Klausel NOT LOGGED INITIALLY definiert wurde, zu einer protokollierten Tabelle, indem Sie die Anweisung COMMIT ausführen. Oder machen Sie die Tabelle, die ohne die Klausel NOT LOGGED INITIALLY definiert wurde, zu einer Tabelle mit NOT LOGGED INITIALLY, indem Sie die folgende Anweisung eingeben:

```
ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY
```

Die Klausel DATA CAPTURE CHANGES wurde entweder für die Quellentabelle oder für die Zieltabelle, jedoch nicht für beide angegeben. (SQL20307N Ursachencode 9)

Führen Sie die folgende Anweisung aus, um der Tabelle, die ohne Klausel DATA CAPTURE CHANGES definiert wurde, diese Klausel hinzuzufügen:

```
ALTER TABLE ... DATA CAPTURE CHANGES
```

Führen Sie die folgende Anweisung aus, um für die Tabelle mit der Klausel DATA CAPTURE CHANGES die Datenerfassung zu inaktivieren:

```
ALTER TABLE ... DATA CAPTURE NONE
```

Die Klauseln DISTRIBUTION der Tabellen stimmen nicht überein. Der Verteilungsschlüssel muss für die Quellen- und Zieltabelle gleich sein. (SQL20307N Ursachencode 10)

Es wird empfohlen, eine neue Quellentabelle zu erstellen. Sie können den Verteilungsschlüssel einer Tabelle, die sich über mehrere Datenbankpartitionen erstreckt, nicht ändern. Führen Sie die folgenden Anweisungen aus, um einen Verteilungsschlüssel für Tabellen in einer Einzelpartitionsdatenbank zu ändern:

```
ALTER TABLE ... DROP DISTRIBUTION;
       ALTER TABLE ... ADD DISTRIBUTION(schlüsselspezifikation)
```

Ein Fehler wird zurückgegeben, wenn während einer ATTACH-Operation festgestellt wird, dass Indizes fehlen (SQL20307N Ursachencode 18).

Die ATTACH-Operation erstellt implizit fehlende Indizes für die Quellentabelle entsprechend den partitionierten Indizes der Zieltabelle. Die implizite Erstellung der fehlenden Indizes nimmt eine gewisse Zeit in Anspruch. Sie haben die Möglichkeit, eine Fehlerbedingung zu erstellen, wenn die ATTACH-Operation fehlende Indizes feststellt. Die Option lautet ERROR ON MISSING INDEXES und ist eine der möglichen Optionen der ATTACH-Operation. Wenn der betreffende Fall eintritt, wird der Fehler SQL20307N, SQLSTATE 428GE, Ursachencode 18 zurückgegeben. Informationen zu den nicht übereinstimmenden Indizes werden im Verwaltungsprotokoll aufgezeichnet.

Die ATTACH-Operation löscht Indizes in der Quellentabelle, die nicht mit den partitionierten Indizes in der Zieltabelle übereinstimmen. Das Identifizieren und Lö-

schen dieser nicht übereinstimmenden Indizes nimmt eine gewisse Zeit in Anspruch. Sie sollten diese Indizes löschen, bevor Sie die ATTACH-Operation ausführen.

Ein Fehler wird zurückgegeben, wenn die nicht übereinstimmenden Indizes für die Zieltabelle eindeutige Indizes sind oder wenn die XML-Indizes mit der Klausel REJECT INVALID VALUES definiert sind (während einer ATTACH-Operation, SQL20307N Ursachencode 17).

Wenn es partitionierte Indizes für die Zieltabelle ohne übereinstimmende Indizes für die Quellentabelle gibt und ERROR ON MISSING INDEXES nicht verwendet wird, sind folgende Ergebnisse zu erwarten:

1. Wenn die nicht übereinstimmenden Indizes für die Zieltabelle eindeutige Indizes sind oder wenn die XML-Indizes mit der Klausel REJECT INVALID VALUES definiert sind, schlägt die ATTACH-Operation fehl und folgende Fehlernachricht wird zurückgegeben: SQL20307N, SQLSTATE 428GE, Ursachencode 17.
2. Wenn die nicht übereinstimmenden Indizes für die Zieltabelle die Bedingungen im vorherigen Punkt nicht erfüllen, wird das Indexobjekt für die Quellentabelle während der ATTACH-Operation als ungültig markiert. Die ATTACH-Operation wird erfolgreich abgeschlossen, aber das Indexobjekt für die neue Datenpartition wird als ungültig markiert. Die SET INTEGRITY-Operation wird verwendet, um die Indexobjekte für die neu zugeordnete Partition erneut zu erstellen. Dies ist normalerweise die nächste Operation, die Sie nach dem Zuordnen der Datenpartition ausführen würden. Das erneute Erstellen der Indizes nimmt eine gewisse Zeit in Anspruch.

Das Verwaltungsprotokoll enthält nähere Informationen über Abweichungen zwischen den Indizes für die Quellen- und Zieltabellen.

Nur für eine der Tabellen wurde eine Klausel ORGANIZE BY DIMENSIONS angegeben oder die Organisationsdimensionen sind verschieden. (SQL20307N Ursachencode 11)

Erstellen Sie eine neue Quellentabelle.

Der Datentyp der Spalten (TYPENAME) stimmt nicht überein. (SQL20408N Ursachencode 1)

Führen Sie die folgende Anweisung aus, um abweichende Datentypen zu korrigieren:

```
ALTER TABLE ... ALTER COLUMN ... SET DATA TYPE...
```

Die Optionalität der Dateneingabe der Spalten (NULL-Werte) stimmt nicht überein. (SQL20408N Ursachencode 2)

Führen Sie eine der folgenden Anweisungen aus, um die Optionalität der Dateneingabe der nicht übereinstimmenden Spalte in einer der Tabellen zu ändern:

```
ALTER TABLE... ALTER COLUMN... DROP NOT NULL
oder
ALTER TABLE... ALTER COLUMN... SET NOT NULL
```

Der implizite Standardwert (SYSCAT.COLUMNS IMPLICITVALUE) der Spalten ist inkompatibel. (SQL20408N Ursachencode 3)

Erstellen Sie eine neue Quellentabelle. Implizite Standardwerte müssen exakt übereinstimmen, wenn sowohl die Zieltabellenspalte als auch die Quellentabellenspalte implizite Standardwerte besitzen (d. h. wenn IMPLICITVALUE nicht gleich NULL ist).

Wenn IMPLICITVALUE für eine Spalte in der Zieltabelle nicht gleich NULL ist und IMPLICITVALUE für die entsprechende Spalte der Quellentabelle nicht gleich NULL ist, wurde jede der Spalten erst nach der Ausführung der ursprünglichen Anweisung CREATE TABLE für die Tabelle hinzugefügt. In diesem Fall müssen die in IMPLICITVALUE gespeicherten Werte für diese Spalte übereinstimmen.

Es ist möglich, dass durch die Migration einer Tabelle von einer Version vor Version 9.1 oder durch das Zuordnen einer Datenpartition aus einer Tabelle vor Version 9.1 der Wert von IMPLICITVALUE nicht NULL ist, weil das System nicht ermitteln konnte, ob die Spalte nach Ausführung der ursprünglichen Anweisung CREATE TABLE hinzugefügt wurde oder nicht. Wenn die Datenbank nicht sicher ermitteln kann, ob die Spalte hinzugefügt wurde oder nicht, wird die Spalte wie eine hinzugefügte Spalte behandelt. Eine hinzugefügte Spalte ist eine Spalte, die durch eine Anweisung ALTER TABLE ...ADD COLUMN erstellt wurde. In diesem Fall ist die Anweisung nicht zulässig, weil der Wert der Spalte ungültig werden könnte, wenn eine Fortsetzung der Zuordnung zugelassen würde. Sie müssen die Daten aus der Quellentabelle in eine neue Tabelle (deren IMPLICITVALUE-Wert für diese Spalte NULL ist) kopieren und die neue Tabelle als Quellentabelle für die Zuordnungsoperation verwenden.

Die Codepage (COMPOSITE_CODEPAGE) der Spalten stimmt nicht überein. (SQL20408N Ursachencode 4)

Erstellen Sie eine neue Quellentabelle.

Die Standardklausel für Systemkomprimierung (COMPRESS) stimmt nicht überein. (SQL20408N Ursachencode 5)

Führen Sie eine der folgenden Anweisungen aus, um die Komprimierung von Systemstandardwerten in der Spalte zu ändern und die Abweichung zu beheben:

```
ALTER TABLE ... ALTER COLUMN ... COMPRESS SYSTEM DEFAULT
oder
ALTER TABLE ... ALTER COLUMN ... COMPRESS OFF
```

Bedingungen für die Übereinstimmung eines Quellentabellenindex mit dem partitionierten Index einer Zieltabelle (während ATTACH PARTITION)

Wenn Sie die Indizes in der Quellentabelle als Indexpartitionen in der Zieltabelle verwenden möchten, müssen alle Indexschlüsselspalten oder Ausdrücke für die Indizes in der Quellentabelle mit den Indexschlüsselspalten oder Ausdrücken für die partitionierten Indizes in der Zieltabelle übereinstimmen.

Wenn die Quellentabelle über ausdrucksbasierte Indizes verfügt, werden die durch das System generierten Statistiksichten und Pakete, die dem Index zugeordnet sind, im Rahmen der ATTACH-Verarbeitung implizit gelöscht. Wenn die Zieltabelle der ATTACH-Verarbeitung über partitionierte ausdrucksbasierte Indizes verfügt, wird der Ausdruck bei der Ermittlung von übereinstimmenden Indizes in der Quellentabelle verwendet. Wenn alle anderen Eigenschaften des Indexes gleich sind, wird der Index für die Quellentabelle als mit dem partitionierten Index für

die Zieltabelle übereinstimmend angesehen. D. h., der Index für die Quellentabelle kann als Index für die Zieltabelle verwendet werden.

Die folgende Tabelle ist nur dann zweckmäßig und anwendbar, wenn der Zielindex partitioniert ist. Die Zielindexeigenschaft wird vom Quellenindex in allen Fällen angenommen, in denen die Indizes als übereinstimmend angesehen werden.

Tabelle 14. Ermitteln, ob der Quellenindex übereinstimmt, wenn sich die Zielindexeigenschaft von der Quellenindexeigenschaft unterscheidet.

Regel-Nr.	Zielindex-eigenschaft	Quellenindex-eigenschaft	Stimmt der Quellenindex überein?
1.	Nicht eindeutig	Eindeutig	Ja, wenn es sich beim Index um keinen XML-Index handelt.
2.	Eindeutig	Nicht eindeutig	Nein
3.	Spalte X ist absteigend	Spalte X ist aufsteigend	Nein
4.	Spalte X ist aufsteigend	Spalte X ist absteigend	Nein
5.	Spalte X ist wahlfrei	Spalte X ist aufsteigend	Nein
6.	Spalte X ist aufsteigend	Spalte X ist wahlfrei	Nein
7.	Spalte X ist wahlfrei	Spalte X ist absteigend	Nein
8.	Spalte X ist absteigend	Spalte X ist wahlfrei	Nein
9.	Partitioniert	Nicht partitioniert	Nein. Hinweis: Hier wird vorausgesetzt, dass die Quellentabelle partitioniert ist.
10.	pctfree n1	pctfree n2	Ja
11.	level2pctfree n1	level2pctfree n2	Ja
12.	minpctused n1	minpctused n2	Ja
13.	Rückwärtsdurchsuchen nicht zulassen	Rückwärtsdurchsuchen zulassen	Ja, die physische Indexstruktur ist gleich, unabhängig davon, ob Rückwärtsdurchsuchen zugelassen ist oder nicht.
14.	Rückwärtsdurchsuchen zulassen	Rückwärtsdurchsuchen nicht zulassen	Ja, gleicher Grund wie bei (9).
15.	pagesplit [L|H|S]	pagesplit [L|H|S]	Ja
16.	Geprüfte Statistikdaten	Detaillierte Statistikdaten	Ja
17.	Detaillierte Statistikdaten	Geprüfte Statistikdaten	Ja
18.	Nicht zusammengefasst	CLUSTER	Ja
19.	CLUSTER	Nicht zusammengefasst	Ja. Der Index wird ein Clusterindex, aber die Daten werden erst beim Reorganisieren entsprechend dem Index zusammengefasst. Sie können eine Reorganisation auf Partitionsebene nach dem Zuordnen verwenden, um die Daten entsprechend dieser Indexpartition zusammenzufassen.
20.	Ungültige Werte ignorieren	Ungültige Werte zurückweisen	Ja
21.	Ungültige Werte zurückweisen	Ungültige Werte ignorieren	Nein. Die Zielindexeigenschaft des Zurückweisens ungültiger Werte muss beachtet werden und die Quellentabelle enthält möglicherweise Zeilen, die gegen diese Indexintegritätsbedingung verstoßen.
22.	Indexkomprimierung aktiviert	Indexkomprimierung nicht aktiviert	Ja. Hinweis: Die Komprimierung der zugrunde liegenden Indexdaten erfolgt erst, wenn der Index erneut erstellt wird.
23.	Indexkomprimierung nicht aktiviert	Indexkomprimierung aktiviert	Ja. Hinweis: Die Dekomprimierung der Indexdaten erfolgt erst, wenn der Index erneut erstellt wird.

Anmerkung: Bei Regel Nummer 9 schlägt eine Anweisung ALTER TABLE ... ATTACH PARTITON fehl und gibt die Fehlernachricht SQL20307N, SQLSTATE 428GE

zurück, wenn Sie versuchen, eine MDC-Tabelle, die mit DB2 9.7 oder früheren Releases (mit nicht partitionierten Blockindizes) erstellt wurde, einer neuen partitionierten MDC-Tabelle zuzuordnen, die mit DB2 Version 9.7 Fixpack 1 oder neueren Releases (mit partitionierten Blockindizes) erstellt wurde, und wenn dabei die Klausel ERROR ON MISSING INDEXES verwendet wird. Durch Entfernen der Klausel ERROR ON MISSING INDEXES kann die Zuordnung abgeschlossen werden, weil der Datenbankmanager die Indizes während der ATTACH-Operation wartet. Wenn Sie Fehlernachricht SQL20307N, SQLSTATE 428GE erhalten haben, sollten Sie in Betracht ziehen, die Klausel ERROR ON MISSING INDEXES zu entfernen.

Eine mögliche Alternative ist die Verwendung der Onlineprozedur zum Versetzen von Tabellen für die Konvertierung einer partitionierten MDC-Tabelle mit nicht partitionierten Blockindizes in eine Tabelle mit partitionierten Blockindizes.

Aufheben der Zuordnung von Datenpartitionen

Die Partitionierung von Tabellen ermöglicht eine effiziente Ein- und Auslagerung von Tabellendaten (engl. Rollin und Rollout). Diese Effizienz wird mithilfe der Klauseln ATTACH PARTITION und DETACH PARTITION in der Anweisung ALTER TABLE erreicht.

Vorbereitende Schritte

Zur Aufhebung der Zuordnung einer Datenpartition zu einer partitionierten Tabelle müssen Sie über die folgenden Berechtigungen und Zugriffsrechte verfügen:

- Der Benutzer, der die DETACH PARTITION-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Quellentabelle zu ändern (ALTER), Daten in der Quellentabelle auszuwählen (SELECT) und aus der Quellentabelle zu löschen (DELETE).
- Der Benutzer muss außerdem über die Berechtigung verfügen, die zum Erstellen der Zieltabelle erforderlich ist. Daher müssen zum Ändern einer Tabelle für die Aufhebung der Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle beinhalten:
  - Berechtigung DBADM
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
    - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
    - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht

Anmerkung: Wenn die Zuordnung einer Datenpartition aufgehoben wird, wird die Anweisung CREATE TABLE effektiv von der Berechtigungs-ID der Anweisung ausgeführt. Daher muss diese Berechtigungs-ID über die erforderlichen Zugriffsrechte zur Ausführung der Operation verfügen. Die Berechtigungs-ID der Anweisung ALTER TABLE wird zum definierenden Benutzer der neuen Tabelle mit der Berechtigung CONTROL, so als ob der Benutzer die Anweisung CREATE TABLE abgesetzt hätte. Von der Tabelle, die geändert wird, werden keine Zugriffsrechte auf die neue Tabelle übertragen. Nur die Berechtigungs-ID der Anweisung ALTER TABLE sowie Benutzer mit der Berechtigung DBADM bzw. DATAACCESS haben unmittelbar nach der Ausführung der Anweisung ALTER TABLE...DETACH PARTITION Zugriff auf die Daten.

Informationen zu diesem Vorgang

Das Rollout partitionierter Tabellendaten ermöglicht eine einfache Abtrennung von Datenbereichen von einer partitionierten Tabelle. Wenn die Zuordnung einer Datenpartition aufgehoben (DETACH PARTITION) und die Datenpartition in eine separate Tabelle umgewandelt wird, kann die Tabelle auf verschiedene Arten gehandhabt werden. Sie können die separate Tabelle löschen (wodurch die Daten in der Datenpartition vernichtet werden). Sie können die Tabelle archivieren oder auf andere Weise als separate Tabelle nutzen. Sie können sie einer anderen partitionierten Tabelle zuordnen (ATTACH PARTITION), zum Beispiel einer Protokolltabelle. Oder Sie können sie bearbeiten, bereinigen, umwandeln und wieder der ursprünglichen oder einer anderen partitionierten Tabelle zuordnen.

Wenn bei DB2 Version 9.7 Fixpack 1 und späteren Releases die Zuordnung einer Datenpartition zu einer partitionierten Tabelle unter Verwendung der Anweisung ALTER TABLE mit der Klausel DETACH PARTITION aufgehoben wird, bleibt die partitionierte Quellentabelle online. Abfragen, die für die Tabelle ausgeführt werden, werden fortgesetzt. Die Datenpartition, deren Zuordnung aufgehoben wird, wird in folgenden beiden Schritten in eine eigenständige Tabelle konvertiert:

1. Die Operation ALTER TABLE...DETACH PARTITION hebt die logische Zuordnung von Datenpartition zu partitionierter Tabelle auf.
2. Eine Task zur asynchronen Aufhebung von Partitionszuordnungen konvertiert die Partition, deren logische Zuordnung aufgehoben wurde, in eine eigenständige Tabelle.

Wenn abhängige Tabellen vorhanden sind, die im Hinblick auf die Datenpartition mit aufgehobener Zuordnung inkrementell gewartet werden müssen (solche abhängigen Tabellen werden als abhängige Tabelle mit aufgehobener Zuordnung bezeichnet), wird die Task zur asynchronen Aufhebung von Partitionszuordnungen erst gestartet, wenn die Anweisung SET INTEGRITY für die abhängigen Tabellen mit aufgehobener Zuordnung ausgeführt wurde.

Wenn keine abhängigen Tabellen mit aufgehobener Zuordnung vorhanden sind, wird die Task zur asynchronen Aufhebung von Partitionszuordnungen gestartet, nachdem die Transaktion, die die Anweisung ALTER TABLE...DETACH PARTITION ausgibt, festgeschrieben wurde.

Einschränkungen

Wenn es sich bei der Quellentabelle um eine mit DB2 Version 9.7 oder früheren Releases erstellte MDC-Tabelle handelt, werden Blockindizes nicht partitioniert. Der Zugriff auf die Tabelle, für die die Zuordnung soeben aufgehoben wurde, ist in derselben UOW, in der auch die Operation ALTER TABLE...DETACH PARTITION ausgeführt wurde, nicht zulässig. MDC-Tabellen unterstützen partitionierte Blockindizes nicht. In diesem Fall werden Blockindizes beim ersten Zugriff auf die Tabelle erstellt, nachdem die Operation ALTER TABLE...DETACH PARTITION festgeschrieben wurde. Wenn die Quellentabelle vor Ausführung von DETACH weitere partitionierte Indizes aufwies, wird das Indexobjekt für die Zieltabelle als ungültig markiert, damit die Blockindizes erstellt werden können. Als Folge hiervon verlängern sich die Zugriffszeiten während der Erstellung der Blockindizes und der erneuten Erstellung partitionierter Indizes.

Wenn es sich bei der Quellentabelle um eine MDC-Tabelle handelt, die mit DB2 V9.7 Fixpack 1 oder neueren Releases erstellt wurde, werden die Blockindizes par-

titioniert und die partitionierten Indizes werden zu Indizes für die Zieltabelle der DETACH-Operation, ohne dass sie erneut erstellt werden müssen.

Die folgenden Bedingungen müssen erfüllt sein, damit eine DETACH-Operation ausgeführt werden kann:

- Die Tabelle, zu der die Zuordnung einer Datenpartition aufgehoben werden soll (Quellentabelle), muss vorhanden und eine partitionierte Tabelle sein.
- Die Datenpartition, deren Zuordnung aufgehoben werden soll, muss in der Quellentabelle vorhanden sein.
- Die Quellentabelle muss mehr als eine Datenpartition haben. Eine partitionierte Tabelle muss mindestens eine Datenpartition haben. Dieser Kontext bezieht sich nur auf sichtbare und zugeordnete Datenpartitionen. Eine zugeordnete Datenpartition ist eine Datenpartition, die zwar zugeordnet ist, jedoch noch nicht durch die Anweisung SET INTEGRITY geprüft wurde.
- Der Name der Tabelle, die durch die DETACH PARTITION-Operation erstellt werden soll (Zieltabelle), darf noch nicht vorhanden sein.
- Eine DETACH PARTITION-Operation ist für eine Tabelle, die eine übergeordnete Tabelle einer erzwungenen referenziellen Integritätsbeziehung (RI) ist, nicht zulässig. Wenn Sie über Tabellen verfügen, für die eine erzwungene referenzielle Integritätsbeziehung (RI) angegeben ist, und wenn Sie für die übergeordnete Tabelle die Zuordnung einer Datenpartition aufheben wollen, dann steht dazu eine Ausweichlösung zur Verfügung. Im folgenden Beispiel werden alle Anweisungen innerhalb derselben UOW (Unit of Work; Arbeitseinheit) ausgeführt, um gleichzeitig ablaufende Aktualisierungen auszuschließen:

  ```
  // Change the RI constraint to informational:
  ALTER TABLE child ALTER FOREIGN KEY fk NOT ENFORCED;

  ALTER TABLE parent DETACH PARTITION p0 INTO TABLE pdet;

  SET INTEGRITY FOR child OFF;

  // Change the RI constraint back to enforced:
  ALTER TABLE child ALTER FOREIGN KEY fk ENFORCED;

  SET INTEGRITY FOR child ALL IMMEDIATE UNCHECKED;
  // Assuming that the CHILD table does not have any dependencies on partition P0,
  //   and that no updates on the CHILD table are permitted until this UOW is complete,
  //   no RI violation is possible during this UOW.

  COMMIT WORK;
  ```

- Wenn abhängige Tabellen vorhanden sind, die im Hinblick auf die Datenpartition mit aufgehobener Zuordnung inkrementell gewartet werden müssen (solche abhängigen Tabellen werden als abhängige Tabelle mit aufgehobener Zuordnung bezeichnet), muss die Anweisung SET INTEGRITY für die abhängigen Tabellen mit aufgehobener Zuordnung ausgeführt werden, um die Tabellen inkrementell zu warten. Nach der Ausführung der Anweisung SET INTEGRITY für alle abhängigen Tabellen mit aufgehobener Zuordnung macht die Task zur asynchronen Aufhebung von Partitionszuordnungen die Datenpartition ab DB2 V9.7 Fixpack 1 zu einer eigenständigen Zieltabelle. Bis zum Abschluss der asynchronen DETACH-Operation für Partitionen ist die Zieltabelle nicht verfügbar.

Vorgehensweise

1. Zum Ändern einer partitionierten Tabelle und zum Aufheben der Zuordnung einer Datenpartition zu dieser Tabelle geben Sie die Anweisung ALTER TABLE mit der Klausel DETACH PARTITION ein.

2. Optional: Wenn Sie wünschen, dass die eigenständige Tabelle, deren Zuordnung soeben aufgehoben wurde, dieselben Integritätsbedingungen erhält, führen Sie den Befehl ALTER TABLE...ADD CONSTRAINT für die Zieltabelle aus, wenn die DETACH-Operation abgeschlossen ist.

   Wenn der Index für die Quellentabelle partitioniert wurde, dann sind alle Indizes, die zur Erfüllung der Integritätsbedingung erforderlich sind, für die Zieltabelle bereits vorhanden.

Ergebnisse

Die Partition mit aufgehobener Zuordnung wird umbenannt und erhält einen systemgenerierten Namen (mit dem Format SQL*jjmmtthhmmssxxx*), sodass bei einer späteren Zuordnung sofort wieder der Name der Partition mit aufgehobener Zuordnung verwendet werden kann.

Jede Indexpartition, die in der Quellentabelle für die Datenpartition mit aufgehobener Zuordnung definiert ist, wird zu einem Index in der Zieltabelle. Das Indexobjekt wird während der DETACH PARTITION-Operation nicht physisch versetzt. Die Metadaten für die Indexpartitionen der Tabellenpartition mit aufgehobener Zuordnung werden aber aus der Katalogtabelle SYSINDEXPARTITIONS entfernt. Für die neue Tabelle werden als Ergebnis der DETACH PARTITION-Operation neue Indexeinträge zu SYSINDEXES hinzugefügt. Die ursprüngliche Index-ID (IID) wird beibehalten und bleibt wie in der Quellentabelle eindeutig.

Die Indexnamen für die beibehaltenen Indizes der Zieltabelle werden vom System generiert und verwenden das Format SQL*jjmmtthhmmssxxx*. Das Schema für diese Indizes ist dasselbe wie das Schema der Zieltabelle, abgesehen von Pfad- und Regionsindizes sowie MDC- oder ITC-Blockindizes im Schema SYSIBM. Andere systemgenerierte Indizes, wie z. B. diejenigen, die eindeutige bzw. über Primärschlüssel definierte Integritätsbedingungen umsetzen, haben ein Schema der Zieltabelle, weil nur die Indizes an die Tabelle mit aufgehobener Zuordnung übertragen werden, nicht aber die Integritätsbedingungen. Mit der Anweisung RENAME können Sie die Indizes umbenennen, die sich nicht im Schema SYSIBM befinden.

Die beim Erstellen der Quellentabelle angegebene Option INDEX IN auf Tabellenebene wird von der Zieltabelle nicht übernommen. Vielmehr bleibt die Option INDEX IN auf Partitionsebene (falls angegeben) oder der Standardindextabellenbereich für die DETACH-Partition auch der Indextabellenbereich für die Zieltabelle.

Beim Aufheben der Zuordnung von Datenpartitionen werden Statistikdaten von der Partition in die Zieltabelle übertragen. Im Besonderen werden Statistikdaten für partitionierte Indizes von SYSINDEXPARTITIONS für die Tabelle mit soeben aufgehobener Zuordnung in SYSINDEXES übertragen. Die Statistikdaten von SYSDATAPARTITIONS werden für die Tabelle mit soeben aufgehobener Zuordnung in SYSTABLES kopiert.

Nächste Schritte

Führen Sie nach Abschluss der DETACH PARTITION-Operation den Befehl **RUNSTATS** sowohl für die Tabelle mit soeben aufgehobener Zuordnung als auch für die Quellentabelle aus, da ein großer Teil der Statistikdaten nicht automatisch nach Abschluss der DETACH PARTITION-Operation übertragen werden.

Attribute von Datenpartitionen mit aufgehobener Zuordnung

Wenn Sie die Zuordnung einer Datenpartition zu einer partitionierten Tabelle mit der Klausel DETACH PARTITION der Anweisung ALTER TABLE aufheben, wird diese Datenpartition zu einer eigenständigen, nicht partitionierten Zieltabelle.

Viele Attribute der Zieltabelle werden aus der Quellentabelle übernommen. Attribute, die nicht aus der Quellentabelle übernommen werden, werden so definiert, als ob der Benutzer, der die DETACH-Operation initiiert, die Zieltabelle erstellen würde. Wenn für die Quellentabelle ein partionierter Index vorhanden ist, wird dieser Index in die Zieltabelle übertragen. Wenn die Quellentabelle über einen partionierten Index mit ausdrucksbasierten Schlüsselteilen verfügt, werden die durch das System generierten Statistiksichten und Pakete erstellt und dem Index in der Zieltabelle zugeordnet.

Die Zieltabelle übernimmt nach der DETACH-Operation alle für die Quellentabelle definierten partitionierten Indizes. Die Indizes umfassen systemgenerierte und benutzerdefinierte Indizes. Das Indexobjekt wird während der DETACH-Operation nicht physisch versetzt. Die Indexpartitionsmetadaten der Datenpartition, deren Zuordnung aufgehoben wird, werden aus dem Katalog SYSINDEXPARTITIONS entfernt. Neue Einträge werden für die neue Tabelle zu SYSINDEXES hinzugefügt. Die Index-ID (IID) für jeden beliebigen partitionierten Index aus der Quellentabelle wird zur IID für den Index in der Zieltabelle (die IID bleibt in Bezug auf die Tabelle eindeutig und wird während der DETACH-Operation nicht verändert).

Die Indexnamen der in der neuen Tabelle weiter bestehenden Indizes sind systemgeneriert und haben das Format SQLyymmddhhmmssxxx. Indizes für Pfade und Regionen sowie MDC-oder ITC-Indizes werden in das Schema SYSIBM aufgenommen. Alle anderen Indizes werden in das Schema der neuen Tabelle aufgenommen. Systemgenerierte Indizes, wie diejenigen, die eindeutige bzw. über Primärschlüssel definierte Integritätsbedingungen umsetzen, werden in das Schema der neuen Tabelle aufgenommen, da die Indizes in die neue Tabelle übertragen werden. Integritätsbedingungen für die Quellentabelle werden nach der DETACH-Operation nicht von der Zieltabelle übernommen.

Mit der Anweisung RENAME können Sie die nicht im Schema SYSIBM enthaltenen Indizes zu einem anderen Zeitpunkt umbenennen.

Sie können die Anweisung ALTER TABLE ... ADD CONSTRAINT nach Abschluss der DETACH-Operation für die neue Tabelle ausführen, damit für die neue Tabelle dieselben Integritätsbedingungen umgesetzt werden wie für die Quellentabelle.

Die Tabellenbereichsposition, die für die Quellentabelle durch die Klausel INDEX IN auf Tabellenebene angegeben wurde, wird nicht von der neuen Zieltabelle übernommen. Vielmehr bleibt die Tabellenbereichsposition, die durch die Klausel INDEX IN auf Partitionsebene angegeben wurde, oder der Standardindextabellenbereich für die neue Tabelle weiterhin die Position des Indextabellenbereichs für die neue Tabelle.

Übernommene Attribute der Zieltabelle

Die folgenden Attribute werden durch die Zieltabelle übernommen:

- Die folgenden Spaltendefinitionen:
  - Spaltenname
  - Datentyp (mit Länge und Genauigkeit bei Datentypen, die eine Länge und Genauigkeit besitzen, z. B. CHAR und DECIMAL)

  - Optionalität der Dateneingabe (NULLABLE)
  - Spaltenstandardwerte
  - INLINE LENGTH
  - Codepage (Spalte CODEPAGE der Katalogsicht SYSCAT.COLUMNS)
  - Protokollierung für LOBs (Spalte LOGGED der Katalogsicht SYSCAT.COLUMNS)
  - Kompaktheit für LOBs (Spalte COMPACT der Katalogsicht SYSCAT.COLUMNS)
  - Komprimierung (Spalte COMPRESS der Katalogsicht SYSCAT.COLUMNS)
  - Typ von verdeckter Spalte (Spalte HIDDEN der Katalogsicht SYSCAT.COLUMNS)
  - Spaltenreihenfolge
- Wenn die Quellentabelle eine MDC-Tabelle (Multidimensional Clustering Table; mehrdimensionale Clusteringtabelle) oder eine ITC-Tabelle (Insert Time Clustering; Clustering anhand der Einfügungszeit) ist, ist die Zieltabelle ebenfalls eine MDC- oder ITC-Tabelle, die mit den gleichen Dimensionsspalten definiert ist.
- Blockindexdefinitionen. Die Indizes werden beim ersten Zugriff auf die neue unabhängige Tabelle mit aufgehobener Zuordnung erneut erstellt, nachdem die DETACH-Operation festgeschrieben wurde.
- Die Tabellenbereichs-ID und die Tabellenobjekt-ID werden aus der Datenpartition, nicht aus der Quellentabelle übernommen. Dies liegt daran, dass während einer DETACH-Operation keine Tabellendaten versetzt werden. In Bezug auf den Katalog wird die Spalte TBSPACEID der Katalogsicht SYSCAT.DATAPARTITIONS aus der Quellendatenpartition zur Spalte TBSPACEID der Katalogsicht SYSCAT.TABLES. Übersetzt in einen Tabellenbereichsnamen ist dies die Spalte TBSPACE der Katalogsicht SYSCAT.TABLES in der Zieltabelle. Die Spalte PARTITIONOBJECTID der Katalogsicht SYSCAT.DATAPARTITIONS aus der Quellendatenpartition wird zur Spalte TABLEID der Katalogsicht SYSCAT.TABLES in der Zieltabelle.
- Die Spalte LONG_TBSPACEID der Katalogsicht SYSCAT.DATAPARTITIONS aus der Quellendatenpartition wird in einen Tabellenbereichsnamen übersetzt und wird zur Spalte LONG_TBSPACE der Katalogsicht SYSCAT.TABLES der Zieltabelle.
- Der Spaltenwert INDEX_TBSPACEID in SYSDATAPARTITIONS für die Quellendatenpartition (Indextabellenbereich auf Partitionsebene) wird in einen Tabellenbereichsnamen umgesetzt und wird zum INDEX_TBSPACE-Wert in SYSTABLES für die Zieltabelle. Der in der Anweisung CREATE TABLE durch INDEX IN <tabellenbereich> auf Tabellenebene angegebene Indextabellenbereich wird nicht von der Zieltabelle übernommen.
- Tabellenbereichsposition
- ID der Verteilungszuordnung für eine Mehrpartitionsdatenbank (Spalte PMAP_ID der Katalogsicht SYSCAT.TABLES)
- Freizuhaltender Speicherbereich (Spalte PCTFREE der Katalogsicht SYSCAT.TABLES)
- Anfügemodus (Spalte APPEND_MODE der Katalogsicht SYSCAT.TABLES)
- Bevorzugte Sperrgranularität (Spalte LOCKSIZE der Katalogsicht SYSCAT.TABLES)
- Datenerfassung (Spalte DATA_CAPTURE der Katalogsicht SYSCAT.TABLES)
- VOLATILE (Spalte VOLATILE der Katalogsicht SYSCAT.TABLES)
- DROPRULE (Spalte DROPRULE der Katalogsicht SYSCAT.TABLES)

- Komprimierung (Spalte COMPRESSION der Katalogsicht SYSCAT.TABLES)
- MAXFREESPACESEARCH (Spalte MAXFREESPACESEARCH der Katalogsicht SYSCAT.TABLES)

Anmerkung: Partitionierte hierarchische oder temporäre Tabellen, Bereichsclustertabellen und partitionierte Sichten werden nicht unterstützt.

Nicht aus der Quellentabelle übernommene Attribute

Die folgenden Attribute werden nicht aus der Quellentabelle übernommen:

- Der Typ der Zieltabelle wird nicht übernommen. Die Zieltabelle ist in jedem Fall eine reguläre Tabelle.
- Zugriffsrechte und Berechtigungen.
- Schema.
- Generierte Spalten, Identitätsspalten, Prüfungen auf Integritätsbedingung, referenzielle Integritätsbedingungen. In dem Fall, dass eine Quellenspalte eine generierte Spalte oder eine Identitätsspalte ist, hat die entsprechende Zielspalte keinen expliziten Standardwert, d. h., der Standardwert ist NULL.
- Indextabellenbereich auf Tabellenebene (Spalte INDEX_TBSPACE der Katalogsicht SYSCAT.TABLES). Indizes für die Tabelle, die aus der DETACH-Operation resultieren, befinden sich in demselben Tabellenbereich wie die Tabelle.
- Trigger
- Über Primärschlüssel definierte und eindeutige Integritätsbedingungen
- Statistikdaten für nicht partitionierte Indizes werden nicht übernommen.
- Alle anderen Attribute, die nicht explizit in der Liste der Attribute, die aus der Quellentabelle übernommen werden, aufgeführt sind.

Phasen der Aufhebung der Partitionszuordnung

Bei DB2 Version 9.7 Fixpack 1 und neueren Releases erfolgt die Aufhebung der Zuordnung einer Datenpartition zu einer Datenpartitionstabelle in zwei Phasen. In der ersten Phase wird die logische Zuordnung der Partition zur Tabelle aufgehoben. In der zweiten Phase wird die Datenpartition in eine eigenständige Tabelle konvertiert.

Der Prozess, bei dem die Zuordnung einer Datenpartition aufgehoben wird, wird durch Ausgabe einer Anweisung ALTER TABLE...DETACH PARTITION initialisiert:

1. Die Operation ALTER TABLE...DETACH PARTITION hebt die logische Zuordnung von Datenpartition zu partitionierter Tabelle auf.
2. Eine Task zur asynchronen Aufhebung von Partitionszuordnungen konvertiert die Partition, deren logische Zuordnung aufgehoben wurde, in die eigenständige Tabelle.

Wenn abhängige Tabellen vorhanden sind, die im Hinblick auf die Datenpartition mit aufgehobener Zuordnung inkrementell gewartet werden müssen (solche abhängigen Tabellen werden als abhängige Tabelle mit aufgehobener Zuordnung bezeichnet), wird die Task zur asynchronen Aufhebung von Partitionszuordnungen erst gestartet, wenn die Anweisung SET INTEGRITY für die abhängigen Tabellen mit aufgehobener Zuordnung ausgeführt wurde.

Wenn keine abhängigen Tabellen mit aufgehobener Zuordnung vorhanden sind, wird die Task zur asynchronen Aufhebung von Partitionszuordnungen gestartet, nachdem die Transaktion, die die Anweisung ALTER TABLE...DETACH PARTITION ausgibt, festgeschrieben wurde.

DETACH-Operation

Die Operation ALTER TABLE...DETACH PARTITION wird auf folgende Art und Weise ausgeführt:

- Die DETACH-Operation wartet dynamische Abfragen mit der Isolationsstufe UR (Uncommitted Read - nicht festgeschriebener Lesevorgang) vor ihrer Weiterverarbeitung nicht ab und sie unterbricht keine aktiven dynamischen UR-Abfragen. Die Operation verhält sich auch dann so, wenn die UR-Abfrage auf die Partition zugreift, deren Zuordnung aufgehoben wird.
- Wenn dynamische Nicht-UR-Abfragen (Lese- oder Schreibabfragen) die Partition, deren Zuordnung aufgehoben werden soll, nicht gesperrt haben, kann die DETACH-Operation durchgeführt werden, während dynamische Nicht-UR-Abfragen für die Tabelle ausgeführt werden.
- Wenn dynamische Nicht-UR-Abfragen die Partition, deren Zuordnung aufgehoben werden soll, gesperrt haben, wartet die DETACH-Operation, bis die Sperre freigegeben wurde.
- Eine absolute Inaktivierung muss für alle statischen Pakete erfolgen, die von der Tabelle abhängig sind, bevor die DETACH-Operation weiter ausgeführt werden kann.
- Die folgenden Einschränkungen, die für DDL-Anweisungen (DDL = Data Definition Language) gelten, betreffen auch eine DETACH-Operation, da für die Ausführung von DETACH auch Kataloge aktualisiert werden müssen:
  - Neue Abfragen können nicht für die Tabelle kompiliert werden.
  - Eine Bindeoperation bzw. erneute Bindeoperation kann nicht für Abfragen erfolgen, die gerade für eine Tabelle ausgeführt werden.

  Um den Einfluss dieser Einschränkungen zu minimieren, führen Sie eine COMMIT-Operation direkt nach der DETACH-Operation aus.

Während der DETACH-Operation wird der Name der Datenpartition in einen systemgenerierten Namen mit dem Format `SQL`*`yymmddhhmmssxxx`* geändert und in SYSCAT.DATAPARTITIONS wird der Partitionsstatus auf 'L' gesetzt, wenn es keine abhängigen Tabellen mit aufgehobener Zuordnung gibt, oder auf 'D', wenn abhängige Tabellen mit aufgehobener Zuordnung vorhanden sind.

Während der DETACH-Operation wird für die Zieltabelle ein Eintrag in SYSCAT.TABLES erstellt. Wenn abhängige Tabellen mit aufgehobener Zuordnung vorhanden sind, wird der Tabellentyp (TYPE) auf 'L' gesetzt. Nachdem SET INTEGRITY für alle abhängigen Tabellen mit aufgehobener Zuordnung ausgeführt wurde, wird TYPE auf 'T' gesetzt, die Zieltabelle ist aber weiterhin nicht verfügbar. Die Task zur asynchronen Aufhebung von Partitionszuordnungen beendet die DETACH-Operation und ermöglicht den Zugriff auf die Zieltabelle.

Bei der vorläufigen Inaktivierung von dynamischem SQL während der DETACH-Operation können dynamische SQL-Abfragen, die vor der Anweisung ALTER TABLE...DETACH PARTITION gestartet wurden, gleichzeitig mit der DETACH-Operation weiter ausgeführt werden. Die Anweisung ALTER TABLE...DETACH PARTITION fordert eine IX-Sperre für die partitionierte Tabelle und eine X-Sperre für die Datenpartition an, deren Zuordnung aufgehoben wird.

Task zur asynchronen Aufhebung von Partitionszuordnungen

Nachdem die DETACH-Operation festgeschrieben wurde und alle abhängigen Tabellen mit aufgehobener Zuordnung aktualisiert wurden, konvertiert eine Task zur asynchronen Aufhebung von Partitionszuordnungen die Partition, deren logische Zuordnung aufgehoben wurde, in die eigenständige Tabelle.

Die Task zur asynchronen Aufhebung von Partitionszuordnungen wartet, bis alle Zugriffe auf die partitionierte Tabelle beendet sind, die vor Phase 1 der DETACH-Operation gestartet wurden. Wenn die partitionierte Tabelle über nicht partitionierte Indizes verfügt, erstellt die Task zur asynchronen Aufhebung von Partitionszuordnungen die Task zur asynchronen Indexbereinigung, um eine verzögerte Indexbereinigung auszuführen. Nachdem der Zugriff beendet ist, schließt die Task zur asynchronen Aufhebung von Partitionszuordnungen Phase 2 der DETACH-Operation ab, indem sie die Partition, deren logische Zuordnung aufgehoben wurde, in eine eigenständige Tabelle konvertiert.

Der Befehl **LIST UTILITIES** kann zur Überwachung der Verarbeitung der Task zur asynchronen Aufhebung von Partitionszuordnungen verwendet werden. Der Befehl **LIST UTILITIES** gibt an, ob sich die Task zur asynchronen Aufhebung von Partitionszuordnungen in einem der beiden folgenden Statuszustände befindet:

- Die Task wartet, bis zuvor bestehende Zugriffe auf die partitionierte Tabelle beendet werden.
- Die Task schließt die DETACH-Operation ab und macht die Zieltabelle verfügbar.

Asynchrone Aufhebung der Partitionszuordnung bei Datenpartitionstabellen

Bei DB2 Version 9.7 Fixpack 1 und neueren Releases schließt die Task zur asynchronen Aufhebung von Partitionszuordnungen die Aufhebung der Zuordnung einer Datenpartition zu einer partitionierten Tabelle ab, die durch eine ALTER TABLE...DETACH-Operation initialisiert wurde. Die Task ist ein asynchroner Hintergrundprozess (Asynchronous Background Process, ABP), der initialisiert wird, nachdem die Zuordnung der Partition logisch aufgehoben wurde.

Die Task zur asynchronen Aufhebung von Partitionszuordnungen beschleunigt den Prozess, bei dem die Zuordnung einer Datenpartition zu einer partitionierten Tabelle aufgehoben wird. Wenn die partitionierte Tabelle abhängige MQTs (Materialized Query Tables) hat, wird die Task erst nach der Ausführung einer Anweisung SET INTEGRITY für die MQTs ausgeführt.

Durch das asynchrone Aufheben der Datenpartitionszuordnung wird die Verarbeitung von Abfragen, die auf die partitionierte Tabelle zugreifen und die vor der Ausgabe der Anweisung ALTER TABLE...DETACH PARTITION gestartet wurden, fortgesetzt, während die Zuordnung der Partition unmittelbar aufgehoben wird.

Wenn abhängige Tabellen vorhanden sind, die im Hinblick auf die Datenpartition mit aufgehobener Zuordnung inkrementell gewartet werden müssen (solche abhängigen Tabellen werden als abhängige Tabelle mit aufgehobener Zuordnung bezeichnet), wird die Task zur asynchronen Aufhebung von Partitionszuordnungen erst gestartet, wenn die Anweisung SET INTEGRITY für die abhängigen Tabellen mit aufgehobener Zuordnung ausgeführt wurde.

Wenn keine abhängigen Tabellen mit aufgehobener Zuordnung vorhanden sind, wird die Task zur asynchronen Aufhebung von Partitionszuordnungen gestartet, nachdem die Transaktion, die die Anweisung ALTER TABLE...DETACH PARTITION ausgibt, festgeschrieben wurde.

Die Task zur asynchronen Aufhebung von Partitionszuordnungen führt die folgenden Operationen aus:

- Sie führt eine absolute Inaktivierung für Anweisungen aus, für die die ALTER TABLE...DETACH-Operation zuvor eine vorläufige Inaktivierung ausgeführt hat.
- Sie aktualisiert die Katalogeinträge für partitionierte Quellentabelle und eigenständige Zieltabelle und macht die Zieltabelle verfügbar.
- Sie erstellt bei MDC-Tabellen mit nicht partitionierten Blockindizes und ohne andere partitionierte Indizes ein Indexobjekt für die Zieltabelle. Die Blockindizes werden beim ersten Zugriff auf die Zieltabelle erstellt, nachdem die Task zur asynchronen Aufhebung von Partitionszuordnungen festgeschrieben wurde.
- Sie erstellt den Systempfadindex für die Zieltabelle (bei Tabellen mit XML-Spalten).
- Sie aktualisiert die Mindestrecoveryzeit (MRT - Minimum Recovery Time) des Tabellenbereichs, der die Partition mit aufgehobener Zuordnung enthält.
- Sie erstellt Tasks zur asynchronen Indexbereinigung (Asynchronous Index Cleanup, AIC) für nicht partitionierte Indizes. Die AIC-Task führt die Indexbereinigung nach der asynchronen Aufhebung der Partitionszuordnung aus.
- Sie gibt die Datenpartitions-ID frei, wenn keine partitionierten Indizes für die Tabelle vorhanden sind.

Auswirkung der Task zur asynchronen Aufhebung von Partitionszuordnungen auf die Leistung

Eine Task zur asynchronen Aufhebung von Partitionszuordnungen wirkt sich nur minimal auf die Leistung aus. Die Task wartet, bis alle Zugriffe auf die Partition mit aufgehobener Zuordnung beendet sind, indem sie eine absolute Inaktivierung für Anweisungen ausführt, für die die ALTER TABLE...DETACH-Operation zuvor eine vorläufige Inaktivierung ausgeführt hat. Anschließend fordert die Task die erforderlichen Sperren für die Tabelle und die Partition an und setzt den Prozess fort, um die Partition mit aufgehobener Zuordnung zu einer eigenständigen Tabelle zu machen.

Überwachung der Task zur asynchronen Aufhebung von Partitionszuordnungen

Der Verteilungsdämonprozess und die Agenten der Task zur asynchronen Aufhebung von Partitionszuordnungen sind interne Systemanwendungen, die in der Ausgabe des Befehls **LIST APPLICATIONS** mit den Anwendungsnamen **db2taskd** bzw. **db2apd** angegeben werden. Zur Vermeidung versehentlicher Unterbrechungen lässt sich der Abbruch von Systemanwendungen nicht erzwingen. Der Verteilungsdämon bleibt online, solange die Datenbank aktiv ist. Die Tasks bleiben aktiv, bis die Aufhebung der Zuordnung abgeschlossen ist. Falls die Datenbank während der Aufhebung der Zuordnung inaktiviert wird, nimmt die Task zur asynchronen Aufhebung von Partitionszuordnungen ihre Arbeit wieder auf, wenn die Datenbank reaktiviert wird.

Der Befehl **LIST UTILITIES** gibt an, ob sich die Task zur asynchronen Aufhebung von Partitionszuordnungen in einem der beiden folgenden Statuszustände befindet:

- Die Task wartet, bis zuvor bestehende Zugriffe auf die partitionierte Tabelle beendet werden.
- Die Task schließt die DETACH-Operation ab und macht die Zieltabelle verfügbar.

Die folgende Beispielausgabe für den Befehl **LIST UTILITIES SHOW DETAIL** zeigt die Aktivität der Task zur asynchronen Aufhebung von Partitionszuordnungen in der WSDB-Datenbank:

```
ID                             = 1
Typ                            = ASYNCHRONOUS PARTITION DETACH
Datenbankname                  = WSDB
Partitionsnummer               = 0
Beschreibung                   = Aufhebung der Zuordnung für Partition '4'
                                 von Tabelle 'USER1.ORDERS' abschließen
Startzeit                      = 07/15/2009 14:52:14.476131
Status                         = Wird ausgeführt
Aufruftyp                      = Automatisch
Fortschrittsüberwachung:
    Beschreibung                   = Die Task wartet, bis zuvor bestehende Zugriffe
                                     auf die partitionierte Tabelle beendet werden.
    Startzeit                      = 07/15/2009 14:52:51.268119
```

In der Ausgabe des Befehls LIST UTILITIES gibt die Hauptbeschreibung für die Task zur asynchronen Aufhebung von Partitionszuordnungen die Datenpartition an, deren Zuordnung aufgehoben wird, sowie die Zieltabelle, die durch die DETACH-Operation erstellt wird. Die Fortschrittsüberwachungsbeschreibung bietet Informationen zum aktuellen Status der Task zur asynchronen Aufhebung von Partitionszuordnungen.

Anmerkung: Die Task zur asynchronen Aufhebung von Partitionszuordnungen ist ein asynchroner Prozess. Um zu ermitteln, wann die Zieltabelle einer DETACH-Operation verfügbar ist, kann eine gespeicherte Prozedur erstellt werden, die die Spalte STATUS der Katalogsicht SYSCAT.DATAPARTITIONS abfragt und ein Ergebnis zurückgibt, wenn die DETACH-Operation abgeschlossen ist.

Asynchrone Aufhebung von Partitionszuordnungen in einer Umgebung mit partitionierten Datenbanken

Unabhängig von der Anzahl der Datenbankpartitionen in einer Umgebung mit partitionierten Datenbanken wird immer *eine* Task zur asynchronen Aufhebung von Partitionszuordnungen für jede DETACH-Operation erstellt. Die Task wird in der Katalogdatenbankpartition erstellt und verteilt die Verarbeitungsprozesse nach Bedarf an die verbleibenden Datenbankpartitionen.

Fehlerbehandlung für die Task zur asynchronen Aufhebung von Partitionszuordnungen

Die Task zur asynchronen Aufhebung von Partitionszuordnungen arbeitet transaktionsbasiert. Alle von der Task vorgenommenen Änderungen werden intern rückgängig gemacht, wenn die Task fehlschlägt. Alle Fehler, die bei der asynchronen Aufhebung der Partitionszuordnung auftreten, werden in einer **db2diag**-Protokolldatei aufgezeichnet. Schlägt eine Task fehl, versucht das System später erneut, sie auszuführen.

Hinzufügen von Datenpartitionen zu partitionierten Tabellen

Mithilfe der Anweisung ALTER TABLE können Sie eine partitionierte Tabelle nach ihrer Erstellung ändern. Insbesondere können Sie einer vorhandenen partitionierten Tabelle mithilfe der Klausel ADD PARTITION eine neue Datenpartition hinzufügen.

Informationen zu diesem Vorgang

Wenn Daten über einen längeren Zeitraum hinweg zu einer Partition hinzugefügt werden, wenn Daten eher in kleinen Mengen als in großen Volumen (Rollin) aus externen Quellen eintreffen oder wenn Sie Daten direkt in eine partitionierte Tabelle einfügen oder laden, dann eignet sich zur Durchführung dieses Vorgangs das Hinzufügen einer Datenpartition zu einer partitionierten Tabelle besser als das Zuordnen einer Datenpartition. Beispiele für solche Fälle sind tägliche Operationen zum Laden von Daten in eine Datenpartition für Januardaten oder auch fortlaufende Einfügungen einzelner Zeilen.

Um die neue Datenpartition in eine bestimmte Tabellenbereichsposition aufzunehmen, wird die Klausel IN als Option in der Anweisung ALTER TABLE ADD PARTITION verwendet.

Um den partitionierten Index einer neuen Datenpartition in eine bestimmte Tabellenbereichsposition - separat von der Tabellenbereichsposition der Datenpartition - aufzunehmen, wird die Klausel INDEX IN auf Partitionsebene als Option in der Anweisung ALTER TABLE ADD PARTITION verwendet. Wenn die Option INDEX IN nicht angegeben wird, befinden sich die partitionierten Indizes für die neue Datenpartition standardmäßig in demselben Tabellenbereich wie die Datenpartition. Wenn partitionierte Indizes für die partitionierte Tabelle bestehen, erstellt die Klausel ADD PARTITION die entsprechenden leeren Indexpartitionen für die neue Partition. Für jeden partitionierten Index wird ein neuer Eintrag in die Katalogsicht SYSCAT.INDEXPARTITIONS eingefügt.

Um die LONG-, LOB- oder XML-Daten einer neuen Datenpartition zu einer bestimmten Tabellenbereichsposition hinzuzufügen, die von der Tabellenbereichsposition der Datenpartition getrennt ist, wird die Klausel LONG IN auf Partitionsebene als Option zur Anweisung ALTER TABLE ADD PARTITION hinzugefügt.

Wenn eine Datenpartition mithilfe der Anweisung ALTER TABLE und der Klausel ADD PARTITION zu einer partitionierten Tabelle hinzugefügt wird, bleibt die partitionierte Zieltabelle online und dynamische Abfragen, die für die Tabelle mit der Isolationsstufe 'Lesestabilität', 'Cursorstabilität' oder 'Nicht festgeschriebener Lesevorgang' ausgeführt werden, werden fortgesetzt.

Einschränkungen und Verwendungshinweise

- Einer nicht partitionierten Tabelle kann keine Datenpartition hinzugefügt werden. Detaillierte Informationen zur Migration einer vorhandenen Tabelle in eine partitionierte Tabelle finden Sie in „Migrieren vorhandener Tabellen und Sichten in partitionierte Tabellen” auf Seite 209.
- Der Bereich von Werten für jede neue Datenpartition wird durch die Klauseln STARTING und ENDING bestimmt.
- Es muss mindestens eine der Klauseln STARTING und ENDING (oder beide) angegeben werden.
- Der neue Bereich darf sich nicht mit dem Bereich einer vorhandenen Datenpartition überschneiden.

- Wenn eine neue Datenpartition vor der ersten vorhandenen Datenpartition hinzugefügt wird, muss die Klausel STARTING angegeben werden. Mithilfe von MINVALUE können Sie diesen Bereich mit offenem Ende definieren.
- Analog muss die Klausel ENDING angegeben werden, wenn eine neue Datenpartition nach der letzten vorhandenen Datenpartition hinzugefügt werden soll. Mithilfe von MAXVALUE können Sie diesen Bereich mit offenem Ende definieren.
- Wenn die Klausel STARTING nicht angegeben wird, generiert die Datenbank eine Startgrenze, die direkt hinter der Endgrenze der vorherigen Datenpartition liegt. Wenn die Klausel ENDING nicht angegeben wird, erstellt die Datenbank analog eine Endgrenze, die direkt vor der Startgrenze der nächsten Datenpartition liegt.
- Die Syntax für die Startklausel und die Endklausel ist die gleiche, die auch in der Anweisung CREATE TABLE zu verwenden ist.
- Wenn keine Klausel IN, INDEX IN oder LONG IN für ADD PARTITION angegeben wird, wird der Tabellenbereich, in dem die Datenpartition zu speichern ist, mit der gleichen Methode ausgewählt wie bei der Anweisung CREATE TABLE.
- Pakete werden bei der Operation ALTER TABLE ...ADD PARTITION ungültig gemacht.
- Die neu hinzugefügte Datenpartition wird verfügbar, nachdem die Anweisung ALTER TABLE festgeschrieben wurde.
- Wenn eine Tabelle über einen nicht partitionierten Index verfügt, ist der Zugriff auf eine neue Datenpartition in dieser Tabelle innerhalb derselben Transaktion (z. B. die ADD- oder ATTACH-Operation, die die Partition erstellt hat), nicht möglich, wenn die Transaktion das Sperren der Tabelle im Exklusivmodus nicht veranlasst (SQL0668N, Ursachencode 11).

Das Weglassen der STARTING- oder ENDING-Grenzen für eine ADD-Operation wird auch dazu genutzt, eine Lücke in Bereichswerten zu füllen. Das folgende Beispiel zeigt, wie eine Lücke mit einer ADD-Operation gefüllt wird, in der nur die STARTING-Grenze angegeben wird:

```
CREATE TABLE hole (c1 int) PARTITION BY RANGE (c1)
(STARTING FROM 1 ENDING AT 10, STARTING FROM 20 ENDING AT 30);
DB20000I  Der SQL-Befehl wurde erfolgreich ausgeführt.

ALTER TABLE hole ADD PARTITION STARTING 15;
DB20000I  Der SQL-Befehl wurde erfolgreich ausgeführt.

SELECT SUBSTR(tabname, 1,12) tabname,
SUBSTR(datapartitionname, 1, 12) datapartitionname,
seqno, SUBSTR(lowvalue, 1, 4) lowvalue, SUBSTR(highvalue, 1, 4) highvalue
FROM SYSCAT.DATAPARTITIONS WHERE TABNAME='HOLE' ORDER BY seqno;

TABNAME DATAPARTITIONNAME SEQNO LOWVALUE HIGHVALUE
------------ ----------------- ----------- -------- ---------
HOLE PART0 0 1 10
HOLE PART2 1 15 20
HOLE PART1 2 20 30

3 Satz/Sätze ausgewählt.
```

Beispiel 1: Sie fügen einer vorhandenen partitionierten Tabelle eine Datenpartition hinzu, die einen Bereich mit Werten von 901 bis 1000 (jeweils einschließlich) enthält. Die Tabelle 'sales' enthält neun Bereiche, und zwar 0 - 100, 101 - 200 usw. bis zum Wert 900. In diesem Beispiel wird ein zusätzlicher Bereich am Ende der Tabelle hinzugefügt, wie durch das Weglassen der Klausel STARTING angegeben wird:

```
 ALTER TABLE sales ADD PARTITION dp10
ENDING AT 1000 INCLUSIVE
```

Um den partitionierten Index einer neuen Datenpartition in eine bestimmte Tabellenbereichsposition - separat von der Tabellenbereichsposition der Datenpartition - aufzunehmen, wird die Klausel INDEX IN auf Partitionsebene als Option in der Anweisung ALTER TABLE ADD PARTITION verwendet. Wenn die Option INDEX IN nicht angegeben wird, befinden sich die partitionierten Indizes für die neue Datenpartition standardmäßig in demselben Tabellenbereich wie die Datenpartition. Wenn partitionierte Indizes für die partitionierte Tabelle bestehen, erstellt ADD PARTITION die entsprechenden leeren Indexpartitionen für die neue Partition. Für jeden partitionierten Index wird ein neuer Eintrag in die Katalogsicht SYSCAT.INDEXPARTITIONS eingefügt.

Beispiel 2: Sie fügen einer vorhandenen partitionierten Tabelle eine Datenpartition hinzu und trennen dabei lange Daten und Indizes vom übrigen Teil der Datenpartition.

```
ALTER TABLE newbusiness ADD PARTITION IN tsnewdata
INDEX IN tsnewindex LONG IN tsnewlong
```

Löschen von Datenpartitionen

Wenn Sie eine Datenpartition löschen möchten, müssen Sie die Zuordnung für die Partition aufheben und die durch die DETACH-Operation erstellte Tabelle löschen. Verwenden Sie die Anweisung ALTER TABLE mit der Klausel DETACH PARTITION, um die Zuordnung für die Partition aufzuheben und eine eigenständige Tabelle zu erstellen, und löschen Sie die Tabelle mit der Anweisung DROP TABLE.

Vorbereitende Schritte

Zur Aufhebung der Zuordnung einer Datenpartition zu einer partitionierten Tabelle muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Der Benutzer, der die DETACH-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Quellentabelle zu ändern (ALTER), Daten aus der Quellentabelle auszuwählen (SELECT) und Daten aus der Quellentabelle zu löschen (DELETE).
- Der Benutzer muss außerdem über die Berechtigung zum Erstellen (CREATE) der Zieltabelle verfügen. Daher müssen zum Ändern einer Tabelle für die Aufhebung der Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle beinhalten:
  - Berechtigung DBADM
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
    - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
    - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht

Zum Löschen einer Tabelle muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Sie müssen entweder der definierende Benutzer sein, der in der Spalte DEFINER der Katalogsicht SYSCAT.TABLES aufgezeichnet ist, oder über mindestens eines der folgenden Zugriffsrechte verfügen:
  - Berechtigung DBADM

– Zugriffsrecht DROPIN für das Schema der Tabelle
– Zugriffsrecht CONTROL für die Tabelle

Anmerkung: Der Fall der Aufhebung der Zuordnung einer Datenpartition impliziert, dass die Anweisung CREATE TABLE effektiv von der Berechtigungs-ID der Anweisung absetzt wird. Daher muss diese Berechtigungs-ID über die erforderlichen Zugriffsrechte zur Ausführung der Operation verfügen. Der Tabellenbereich ist derjenige, in dem sich die Datenpartition befindet, deren Zuordnung aufgehoben wird. Die Berechtigungs-ID der Anweisung ALTER TABLE wird zum definierenden Benutzer der neuen Tabelle mit der Berechtigung CONTROL, so als ob der Benutzer die Anweisung CREATE TABLE abgesetzt hätte. Von der Tabelle, die geändert wird, werden keine Zugriffsrechte auf die neue Tabelle übertragen. Nur die Berechtigungs-ID der Anweisung ALTER TABLE sowie die Berechtigungen DBADM bzw. SYSADM haben unmittelbar nach der Ausführung der Operation ALTER TABLE ... DETACH PARTITION Zugriff auf die Daten.

Vorgehensweise

Zum Aufheben der Zuordnung für eine Datenpartition einer partitionierten Tabelle geben Sie die Anweisung ALTER TABLE mit der Klausel DETACH PARTITION ein.

Beispiel

Im folgenden Beispiel wird die Zuordnung der Datenpartition 'dec01' zur Tabelle STOCK aufgehoben, und die Daten werden in die Tabelle JUNK versetzt. Wenn Sie sichergestellt haben, dass die Task zur asynchronen Aufhebung von Partitionszuordnungen die Zieltabelle JUNK verfügbar gemacht hat, können Sie die Tabelle JUNK löschen und so effektiv die entsprechende Datenpartition löschen.

```
ALTER TABLE stock DETACH PART dec01 INTO junk;
 -- Wenn die Zieltabelle verfügbar ist, setzen Sie die Anweisung DROP TABLE ab.
 DROP TABLE junk;
```

Nächste Schritte

Damit die Operation ALTER TABLE ... DETACH möglichst schnell durchgeführt werden kann, schließt die Task zur asynchronen Aufhebung von Partitionszuordnungen ab DB2 Version 9.7 Fixpack 1 die DETACH-Operation asynchron ab. Wenn abhängige Tabellen mit aufgehobener Zuordnung vorhanden sind, wird die Task zur asynchronen Aufhebung von Partitionszuordnungen nicht gestartet und die Datenpartition mit aufgehobener Zuordnung wird keine eigenständige Tabelle. In diesem Fall muss die Anweisung SET INTEGRITY für alle abhängigen Tabellen mit aufgehobener Zuordnung abgesetzt werden. Nach Abschluss von SET INTEGRITY wird die Task zur asynchronen Aufhebung von Partitionszuordnungen gestartet und ermöglicht den Zugriff auf die Zieltabelle. Wenn der Zugriff auf die Zieltabelle möglich ist, kann sie gelöscht werden.

Szenario: Rotieren von Daten in einer partitionierten Tabelle

Der Begriff des Rotierens von Daten in DB2-Datenbanken bezeichnet eine Methode zur Wiederverwendung von Speicherplatz in einer Datenpartition, bei der veraltete Daten aus einer Tabelle entfernt (DETACH PARTITION-Operation) und neue Daten anschließend hinzugefügt werden (ATTACH PARTITION-Operation).

Vorbereitende Schritte

Alternativ dazu können Sie die Partition mit der aufgehobenen Zuordnung archivieren und die neuen Daten in eine andere Quellentabelle laden, bevor eine ATTACH-Operation durchgeführt wird. Im folgenden Szenario wird eine DETACH-Operation vor den anderen Schritten ausgeführt; es kann ebenso einfach der letzte Schritt sein, je nach Ihren jeweiligen Anforderungen.

Zum Ändern einer Tabelle zum Zweck der Aufhebung der Zuordnung einer Datenpartition muss die Berechtigungs-ID der Anweisung die folgenden Berechtigungen bzw. Zugriffsrechte besitzen:

- Mindestens eine der folgenden Berechtigungen für die Zieltabelle der Partition, deren Zuordnung aufgehoben wurde:
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
    - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der neuen Tabelle nicht existiert
    - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der neuen Tabelle auf ein vorhandenes Schema bezieht
  - Berechtigung DBADM
- Mindestens eines der folgenden Zugriffsrechte bzw. eine der folgenden Berechtigungen für die Quellentabelle:
  - Zugriffsrechte SELECT, ALTER und DELETE für die Tabelle
  - Zugriffsrecht CONTROL für die Tabelle
  - Berechtigung DATAACCESS

Zum Ändern einer Tabelle zu dem Zweck, eine Datenpartition zuzuordnen, muss die Berechtigungs-ID der Anweisung die folgenden Zugriffsrechte und Berechtigungen umfassen:

- Mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Quellentabelle:
  - Zugriffsrecht SELECT für die Tabelle und Zugriffsrecht DROPIN für das Schema der Tabelle
  - Zugriffsrecht CONTROL für die Tabelle
  - Berechtigung DATAACCESS
- Mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle:
  - Zugriffsrechte ALTER und INSERT für die Tabelle
  - Zugriffsrecht CONTROL für die Tabelle
  - Berechtigung DATAACCESS

Vorgehensweise

Zum Rotieren von Daten in einer partitionierten Tabelle setzen Sie die Anweisung ALTER TABLE ab. Das folgende Beispiel veranschaulicht, wie die Tabelle STOCK aktualisiert wird, indem die Daten von Dezember 2008 entfernt und durch die aktuellsten Daten von Dezember 2010 ersetzt werden.

1. Entfernen Sie die alten Daten aus der Tabelle STOCK.

```
ALTER TABLE stock DETACH PARTITION dec08 INTO newtable;
```

2. Laden Sie die neuen Daten. Durch den Befehl **LOAD** und die Option REPLACE werden die vorhandenen Daten überschrieben.

   ```
   LOAD FROM datendatei OF DEL REPLACE INTO newtable
   ```

   Anmerkung: Wenn abhängige Objekte mit aufgehobener Zuordnung vorhanden sind, setzen Sie die Anweisung SET INTEGRITY für die abhängigen Objekte mit aufgehobener Zuordnung ab, bevor Sie die Tabelle mit aufgehobener Zuordnung laden. Wenn der Fehler SQL20285N zurückgegeben wird, warten Sie, bis die Task zur asynchronen Aufhebung von Partitionszuordnungen abgeschlossen ist, bevor Sie die Anweisung SET INTEGRITY erneut absetzen.
3. Führen Sie bei Bedarf bestimmte Datenbereinigungsaktivitäten durch, die die folgenden Aktionen umfassen können:
   - Auffüllen fehlender Werte
   - Löschen inkonsistenter und unvollständiger Daten
   - Entfernen redundanter Daten aus verschiedenen Quellen
   - Umwandlung von Daten
     - *Normalisierung*. Daten aus verschiedenen Quellen, die denselben Wert auf verschiedene Arten darstellen, müssen im Rahmen eines Rollins der Daten in das Data-Warehouse miteinander abgeglichen werden.
     - *Aggregation*. Rohdaten, die zum Speichern im Data-Warehouse zu detailliert sind, müssen vor dem Rollin aggregiert werden.
4. Ordnen Sie die Daten als neuen Bereich zu.

   ```
   ALTER TABLE stock
     ATTACH PARTITION dec10
       STARTING '12/01/2008' ENDING '12/31/2010'
     FROM newtable;
   ```
5. Verwenden Sie die Anweisung SET INTEGRITY, um Indizes und andere abhängige Objekte zu aktualisieren. Der Lese- und Schreibzugriff wird während der Ausführung der Anweisung SET INTEGRITY zugelassen.

   ```
   SET INTEGRITY FOR stock ALLOW WRITE ACCESS
     IMMEDIATE CHECKED  FOR EXCEPTION IN stock USE stock_ex;
   ```

Szenarios: Rollin und Rollout von Daten partitionierter Tabellen

Eine häufige Verwaltungsoperation in Data-Warehouses ist der regelmäßige Rollin neuer Daten und Rollout veralteter Daten. Die folgenden Szenarios zeigen diese Tasks.

Szenario 1: Rollout veralteter Daten durch Aufheben der Zuordnung für eine Datenpartition

Das folgende Beispiel zeigt, wie die Zuordnung für eine nicht mehr benötigte Partition (DEC01) zu einer partitionierten Tabelle mit dem Namen STOCK aufgehoben wird. Die Datenpartition mit aufgehobener Zuordnung wird verwendet, um eine Tabelle mit dem Namen STOCK_DROP zu erstellen, ohne dass Daten versetzt werden.

```
ALTER TABLE stock DETACH PART dec01 INTO stock_drop;
COMMIT WORK;
```

Zur Beschleunigung der DETACH-Operation wird für die Quellentabelle im Hintergrund automatisch eine Indexbereinigung durch einen asynchronen Indexbereinigungsprozess ausgeführt. Wenn keine abhängigen Tabellen mit aufgehobener Zu-

ordnung für die Quellentabelle definiert sind, besteht keine Notwendigkeit, eine Anweisung SET INTEGRITY auszuführen, um die DETACH-Operation abzuschließen.

Die neue Tabelle kann gelöscht oder einer anderen Tabelle zugeordnet werden. Es ist ebenfalls möglich, die Tabelle abzuschneiden und mit neuen Daten zu laden und sie dann erneut der Quellentabelle zuzuordnen. Sie können diese Operationen sofort (d. h. noch vor dem Abschluss der asynchronen Indexbereinigung) ausführen, es sei denn, dass die Quellentabelle die Zuordnung abhängiger Tabellen aufgehoben hat.

Um festzustellen, ob auf eine Tabelle mit aufgehobener Zuordnung Zugriff besteht, fragen Sie die Katalogsicht SYSCAT.TABDETACHEDDEP ab. Wenn auf eine Tabelle mit aufgehobener Zuordnung nicht zugegriffen werden kann, geben Sie die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED für alle abhängigen Tabelle mit aufgehobener Zuordnung ein. Wenn Sie versuchen, auf eine Tabelle mit aufgehobener Zuordnung zuzugreifen, bevor die Verwaltungsoperation für alle abhängigen Tabellen mit aufgehobener Zuordnung abgeschlossen ist, wird ein Fehler (SQL20285N) zurückgegeben.

Szenario 2: Erstellen eines neuen, leeren Bereichs

Das folgende Beispiel zeigt, wie eine leere Datenpartition (DEC02) zu einer partitionierten Tabelle mit dem Namen STOCK hinzugefügt wird. Die Klauseln STARTING FROM und ENDING AT geben den Wertebereich für die neue Datenpartition an.

```
ALTER TABLE stock ADD PARTITION dec02
 STARTING FROM '12/01/2002' ENDING AT '12/31/2002';
```

Diese Anweisung vom Typ ALTER TABLE...ADD PARTITION wartet auf die Beendigung vorhandener statischer Abfragen oder Abfragen für wiederholbares Lesen, die für die Tabelle STOCK ausgeführt werden, und inaktiviert zugehörige Pakete. Solche vorhandenen Abfragen werden in der Regel abgeschlossen, bevor die Hinzufügeoperation mit der Arbeit in der Tabelle STOCK beginnt. Vorhandene dynamische Abfragen, bei denen es sich nicht um Abfragen für wiederholbares Lesen handelt, für die Tabelle STOCK werden fortgesetzt und können gleichzeitig mit der Hinzufügeoperation ausgeführt werden. Nachdem die Anweisung ALTER TABLE...ADD PARTITION die Verarbeitung in der Tabelle STOCK aufgenommen hat, müssen sämtliche neuen Abfragen, die auf diese Tabelle zugreifen, warten, bis die Hinzufügeoperation abgeschlossen ist.

Laden Sie Daten in die Tabelle:

```
LOAD FROM datendatei OF DEL
 INSERT INTO stock
 ALLOW READ ACCESS;
```

Setzen Sie die Anweisung SET INTEGRITY ab, um Integritätsbedingungen zu überprüfen und abhängige MQTs (Materialized Query Tables, gespeicherte Abfragetabellen) zu aktualisieren: Alle Zeilen, die definierte Integritätsbedingungen verletzen, werden in die Ausnahmetabelle STOCK_EX versetzt.

```
SET INTEGRITY FOR stock
 ALLOW READ ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;

COMMIT WORK;
```

Szenario 3: Rollin neuer Daten durch Zuordnen einer geladenen Datenpartition

Das folgende Beispiel zeigt, wie eine Zuordnungsoperation dazu verwendet werden kann, das Laden neuer Daten in eine vorhandene partitionierte Tabelle (die Zieltabelle STOCK) zu vereinfachen. Die Daten werden in eine neue, leere Tabelle (DEC03) geladen, in der sie bei Bedarf überprüft und bereinigt werden können, ohne dass dies Auswirkungen auf die Zieltabelle hat. Zur Datenbereinigung gehören folgende Aktivitäten:

- Auffüllen fehlender Werte
- Löschen inkonsistenter und unvollständiger Daten
- Entfernen redundanter Daten aus verschiedenen Quellen
- Umsetzen der Daten durch Normalisierung oder Aggregation:
  - *Normalisierung*. Daten aus verschiedenen Quellen, die dieselben Werte auf verschiedene Weise darstellen, müssen im Rahmen des Rollins miteinander abgeglichen werden.
  - *Aggregation*. Rohdaten, die zum Speichern in einem Data-Warehouse zu detailliert sind, müssen beim Rollin aggregiert werden.

Nachdem die Daten auf diese Weise vorbereitet wurden, kann die neu geladene Datenpartition der Zieltabelle zugeordnet werden.

```
CREATE TABLE dec03(...);
LOAD FROM datendatei OF DEL REPLACE INTO dec03;
(Datenbereinigung, falls erforderlich)
ALTER TABLE stock ATTACH PARTITION dec03
 STARTING FROM '12/01/2003' ENDING AT '12/31/2003'
 FROM dec03;
```

Während einer Zuordnungsoperation muss mindestens eine der Klauseln STARTING FROM und ENDING AT angegeben werden; die untere Grenze (Klausel STARTING FROM) muss kleiner-gleich der oberen Grenze (Klausel ENDING AT) sein. Die neu zugeordnete Datenpartition darf sich mit keinem vorhandenen Datenpartitionsbereich in der Zieltabelle überschneiden. Wenn die obere Grenze des höchsten vorhandenen Bereichs als MAXVALUE definiert wurde, schlagen alle Versuche, einen neuen höchsten Bereich zuzuordnen, fehl, da sich der neue Bereich mit dem vorhandenen höchsten Bereich überschneidet. Eine ähnliche Einschränkung gilt für die unteren Bereiche, die bei MINVALUE enden. Darüber hinaus können Sie keine neue Datenpartition in der Mitte hinzufügen oder zuordnen, es sei denn, der neue Bereich liegt in einer Lücke zwischen den vorhandenen Bereichen. Grenzen, die vom Benutzer nicht angegeben werden, werden bei der Erstellung der Tabelle bestimmt.

Die Anweisung ALTER TABLE...ATTACH PARTITION wartet auf die Beendigung vorhandener statischer Abfragen oder Abfragen für wiederholbares Lesen, die für die Tabelle STOCK ausgeführt werden, und inaktiviert zugehörige Pakete. Solche vorhandenen Abfragen werden in der Regel abgeschlossen, bevor die Zuordnungsoperation mit der Arbeit in der Tabelle STOCK beginnt. Vorhandene dynamische Abfragen, bei denen es sich nicht um Abfragen für wiederholbares Lesen handelt, für die Tabelle STOCK werden fortgesetzt und können gleichzeitig mit der Zuordnungsoperation ausgeführt werden. Nachdem die Anweisung ALTER TABLE...ATTACH PARTITION die Verarbeitung in der Tabelle STOCK aufgenommen hat, müssen sämtliche neuen Abfragen, die auf diese Tabelle zugreifen, warten, bis die Zuordnungsoperation abgeschlossen ist.

Die Daten in der zugeordneten Datenpartition sind noch nicht sichtbar, weil sie noch nicht durch die Anweisung SET INTEGRITY überprüft wurden. Die Anweisung SET INTEGRITY ist erforderlich, um zu überprüfen, ob sich die neu zugeordneten Daten innerhalb des definierten Bereichs befinden. Darüber hinaus werden damit alle erforderlichen Verwaltungsaktivitäten für Indizes und andere abhängige Objekte, wie z. B. MQTs, durchgeführt. Neue Daten sind erst nach dem Commit der Anweisung SET INTEGRITY sichtbar; wird die Anweisung SET INTEGRITY jedoch online ausgeführt, besteht auf vorhandene Daten in der Tabelle STOCK vollständiger Zugriff für Lese- und Schreiboperationen.

Tipp: Wenn die Prüfung der Datenintegrität (einschließlich Bereichsprüfung und Prüfung anderer Integritätsbedingungen) durch vom Datenserver unabhängige Anwendungslogik vor einer Zuordnungsoperation erfolgen kann, können die neu zugeordneten Daten deutlich früher verfügbar gemacht werden. Sie können den Prozess für Dateneinlagerung optimieren, indem Sie mithilfe der Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED die Bereichsprüfung und die Prüfung auf ungültige Integritätsbedingungen überspringen. In diesem Fall verlässt die Tabelle den Status 'Festlegen der Integrität anstehend' und die neuen Daten sind sofort für Anwendungen verfügbar, sofern die Zieltabelle keine nicht partitionierten Benutzerindizes enthält.

Anmerkung: Es können keine DDL-Anweisungen oder Dienstprogrammoperationen für die Tabelle ausgeführt werden, während die Ausführung der Anweisung SET INTEGRITY läuft. Zu diesen Operationen gehören unter anderem die nachfolgend aufgeführten Anweisungen und Befehle:

- Befehl LOAD
- Befehl REDISTRIBUTE DATABASE PARTITION GROUP
- Befehl REORG INDEXES/TABLE
- Anweisung ALTER TABLE
  - ADD COLUMN
  - ADD PARTITION
  - ATTACH PARTITION
  - DETACH PARTITION
- Anweisung CREATE INDEX

Die Anweisung SET INTEGRITY prüft die Daten in der neu zugeordneten Datenpartition.

```
SET INTEGRITY FOR stock
 ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

Durch das Commit der Transaktion wird die Tabelle verfügbar:

```
COMMIT WORK;
```

Alle Zeilen, die außerhalb des gültigen Bereichs liegen oder sonstige Integritätsbedingungen verletzen, werden in die Ausnahmetabelle STOCK_EX versetzt. Sie können Abfragen für diese Tabelle ausführen, die Zeilen korrigieren und diese in die Tabelle STOCK einfügen.

Laden von Daten

Parallelität und Laden

Das Dienstprogramm LOAD kann die Vorteile einer Hardwarekonfiguration nutzen, in der mehrere Prozessoren oder mehrere Speichereinheiten verwendet werden, wie z. B. in einer symmetrischen Mehrprozessorumgebung (SMP - Symmetric Multiprocessor).

Es gibt mehrere Möglichkeiten, wie eine parallele Verarbeitung umfangreicher Datenmengen mit dem Dienstprogramm LOAD durchgeführt werden kann. Eine Möglichkeit besteht in der Verwendung mehrerer Speichereinheiten, die eine E/A-Parallelität während der Ladeoperation ermöglicht (siehe Abb. 37). Eine weitere Möglichkeit ist der Einsatz mehrerer Prozessoren in einer SMP-Umgebung, der eine partitionsinterne Parallelität ermöglicht (siehe Abb. 38). Beide Möglichkeiten können kombiniert verwendet werden, um ein noch schnelleres Laden der Daten zu erreichen.

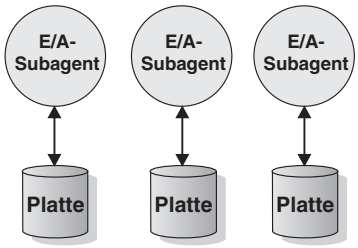


Abbildung 37. E/A-Parallelität beim Laden von Daten

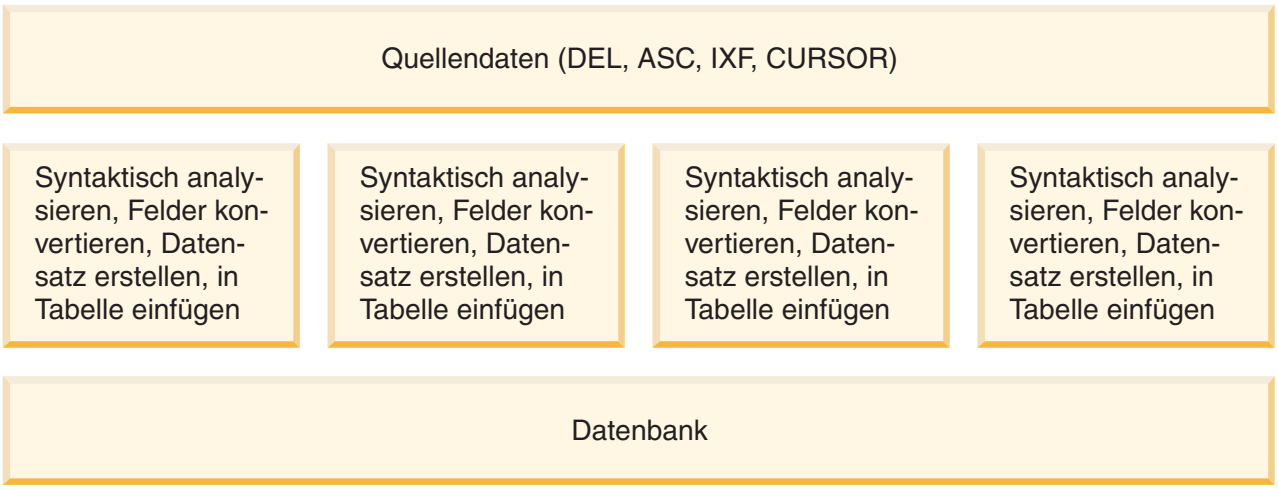


Abbildung 38. Partitionsinterne Parallelität beim Laden von Daten

Hinweise zu MDC und ITC

Für das Laden von Daten in MDC-Tabellen (Multi-Dimensional Clustering - mehrdimensionales Clustering) und ITC-Tabellen (Insert-Time Clustering - Clustering anhand der Einfügungszeit) gelten die folgenden Einschränkungen:

- Die Option `SAVECOUNT` des Befehls **`LOAD`** wird nicht unterstützt.
- Der Änderungswert `totalfreespace` für den Dateityp wird nicht unterstützt, da diese Tabellen ihren freien Speicherbereich selbst verwalten.

- Der Änderungswert `anyorder` für den Dateityp ist für MDC- und ITC-Tabellen erforderlich. Werden Daten ohne den Änderungswert `anyorder` in eine MDC- oder ITC-Tabelle geladen, wird dieser Änderungswert vom Dienstprogramm explizit aktiviert.

Bei Verwendung des Befehls **LOAD** mit einer MDC- oder ITC-Tabelle werden Verstöße gegen eindeutige Integritätsbedingungen wie folgt behandelt:

- Wenn die Tabelle vor der Ladeoperation einen eindeutigen Schlüssel enthielt und doppelte Datensätze in die Tabelle geladen werden, verbleibt der ursprüngliche Datensatz in der Tabelle, und die neuen Datensätze werden während der DELETE-Phase gelöscht.
- Enthielt die Tabelle vor der Ladeoperation keinen eindeutigen Schlüssel und werden sowohl ein eindeutiger Schlüssel als auch doppelte Datensätze in die Tabelle geladen, wird nur einer der Datensätze mit dem eindeutigen Schlüssel geladen, und die anderen Datensätze werden während der DELETE-Phase gelöscht.

  Anmerkung: Es gibt keine explizite Methode, mit der bestimmt werden kann, welche Datensätze geladen und welche gelöscht werden.

Leistungsaspekte

Um eine bessere Leistung des Dienstprogramms LOAD beim Laden von MDC-Tabellen mit mehreren Dimensionen zu erzielen, sollte der Wert des Datenbankkonfigurationsparameters *util_heap_sz* erhöht werden. Die Leistung des Algorithmus 'mdc-load' ist deutlich besser, wenn dem Dienstprogramm mehr Speicher zur Verfügung steht. Dies verringert die Platten-E/A während des Datenclusterings, das in der LOAD-Phase stattfindet. Ab Version 9.5 kann der Wert für die Option DATA BUFFER des Befehls **LOAD** den Wert von *util_heap_sz* temporär überschreiten, falls im System mehr Speicher zur Verfügung steht. .

MDC- oder ITC-Ladeoperationen enthalten immer eine BUILD-Phase, da alle MDC- und ITC-Tabellen mit Blockindizes ausgestattet sind.

Während der LOAD-Phase werden zusätzliche Protokollierungen zur Verwaltung der Blockzuordnung ausgeführt. Pro zugeordnetem Speicherbereich werden ca. zwei zusätzliche Protokollsätze erstellt. Um eine zufriedenstellende Leistung zu gewährleisten, sollte der Datenbankkonfigurationsparameter *logbufsz* auf einen Wert gesetzt werden, der diesen Umstand berücksichtigt.

Zum Laden von Daten in MDC- und ITC-Tabellen wird eine temporäre Systemtabelle mit einem Index eingesetzt. Die Größe der Tabelle ist proportional zur Anzahl der unterschiedlichen geladenen Zellen. Die Größe jeder Zeile in der Tabelle ist proportional zur Größe des Schlüssels für die MDC-Dimension. ITC-Tabellen (ITC = Insert Time Clustering - Clustering anhand der Einfügungszeit) haben nur eine Zelle und verwenden einen 2-Byte-Dimensionsschlüssel. Um die Platten-E/A zu reduzieren, die durch die Bearbeitung dieser Tabelle während einer Ladeoperation entsteht, muss sichergestellt sein, dass der Pufferpool für den Tabellenbereich für temporäre Tabellen groß genug ist.

Ladeaspekte für partitionierte Tabellen

Alle vorhandenen Ladefunktionen werden unterstützt, wenn die Zieltabelle partitioniert ist. Hierbei gelten jedoch die folgenden allgemeinen Einschränkungen:

- Konsistenzpunkte werden nicht unterstützt, wenn die Anzahl der Partitionierungsagenten größer als 1 ist.

- Das Laden von Daten in eine Untergruppe von Datenpartitionen, während die verbleibenden Datenpartitionen vollständig online bleiben, wird nicht unterstützt.
- Die von einer Ladeoperation verwendete Ausnahmetabelle kann nicht partitioniert sein.
- Es kann keine Ausnahmetabelle angegeben werden, wenn die Zieltabelle eine XML-Spalte enthält.
- Ein eindeutiger Index kann nicht erneut erstellt werden, wenn das Dienstprogramm LOAD im Einfügemodus (INSERT) oder Neustartmodus (RESTART) ausgeführt wird und freigegebene Objekte von der Zieltabelle der Ladeoperation abhängen.
- Ähnlich wie beim Laden von MDC-Tabellen bleibt die genaue Reihenfolge der Eingabedatensätze beim Laden von partitionierten Tabellen nicht erhalten. Sortierungen werden lediglich innerhalb der Zelle oder Datenpartition beibehalten.
- Ladeoperationen, die mehrere Formatierungsprogramme auf jeder Datenbankpartition verwenden, erhalten lediglich eine ungefähre Reihenfolge der Eingabedatensätze. Beim Ausführen eines einzelnen Formatierungsprogramms auf jeder Datenbankpartition werden die Eingabedatensätze nach Zellen- oder Tabellenpartitionierungsschlüssel gruppiert. Um auf jeder Datenbankpartition ein einziges Formatierungsprogramm auszuführen, muss für `CPU_PARALLELISM` explizit der Wert 1 angefordert werden.

Allgemeines Verhalten des Dienstprogramms LOAD

Das Dienstprogramm LOAD fügt Datensätze in die richtige Datenpartition ein. Es ist nicht erforderlich, ein externes Dienstprogramm (wie beispielsweise einen Verteilerprozess) zu verwenden, um die Eingabedaten vor dem Laden zu partitionieren.

Das Dienstprogramm LOAD greift nicht auf freigegebene oder zugeordnete Datenpartitionen zu. Daten werden lediglich in sichtbare Datenpartitionen eingefügt. Sichtbare Datenpartitionen sind weder zugeordnet noch freigegeben. Darüber hinaus werden freigegebene oder zugeordnete Datenpartitionen von LOAD REPLACE-Operationen nicht abgeschnitten. Da das Dienstprogramm LOAD Sperren für die Systemkatalogtabellen anfordert, wartet das Dienstprogramm LOAD auf alle ALTER TABLE-Transaktionen, für die noch kein Commit durchgeführt wurde. Solche Transaktionen fordern eine exklusive Sperre für die relevanten Zeilen in den Katalogtabellen an, und die exklusive Sperre muss beendet werden, bevor die Ladeoperation fortgesetzt werden kann. Dies bedeutet, dass nicht festgeschriebene Transaktionen ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION nicht zulässig sind, während die Ladeoperation ausgeführt wird. Alle Eingabequellensätze, die für eine zugeordnete (ATTACHED) oder freigegebene (DETACHED) Datenpartition bestimmt sind, werden zurückgewiesen und können aus der Ausnahmetabelle abgerufen werden, sofern eine angegeben wurde. Eine Informationsnachricht wird in die Nachrichtendatei geschrieben, um anzugeben, dass einige Datenpartitionen der Zieltabelle im Status zugeordnet (ATTACHED) oder freigegeben (DETACHED) waren. Sperren für die relevanten Zeilen der Katalogtabelle, die der Zieltabelle entsprechen, verhindern, dass Benutzer die Partitionierung der Zieltabelle durch Absetzen einer Operation ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION während der Ausführung des Dienstprogramms LOAD ändern.

Verarbeitung ungültiger Zeilen

Wenn das Dienstprogramm LOAD auf einen Datensatz trifft, der zu keiner der sichtbaren Datenpartitionen gehört, wird der betreffende Datensatz zurückgewiesen, und das Dienstprogramm LOAD setzt die Verarbeitung fort. Die Anzahl der Datensätze, die aufgrund einer Verletzung der Bereichsvorgabe zurückgewiesen wurden, wird nicht explizit angegeben, ist aber in der Gesamtanzahl der zurückgewiesenen Datensätze enthalten. Durch das Zurückweisen eines Datensatzes aufgrund einer Verletzung der Bereichsvorgabe wird die Anzahl der Warnungen für Zeilen nicht erhöht. Es wird eine einzige Nachricht (SQL0327N) in die Nachrichtendatei des Dienstprogramms LOAD geschrieben, die angibt, dass Bereichsverletzungen festgestellt wurden. Es wird jedoch nicht für jeden Datensatz eine eigene Nachricht protokolliert. Zusätzlich zu allen Spalten der Zieltabelle enthält die Ausnahmetabelle auch Spalten, die den Typ des Verstoßes beschreibt, der für eine bestimmte Zeile aufgetreten ist. Zeilen, die ungültige Daten enthalten (einschließlich Daten, die nicht partitioniert werden können), werden in die Speicherauszugsdatei geschrieben.

Da Einfügungen (INSERT-Operationen) in die Ausnahmetabelle ressourcenintensiv sind, können Sie steuern, welche Verstöße gegen Integritätsbedingungen in die Ausnahmetabelle eingefügt werden sollen. Das Standardverhalten des Dienstprogramms LOAD ist beispielsweise, Zeilen in die Ausnahmetabelle einzufügen, die aufgrund einer Verletzung der Bereichsvorgabe oder der eindeutigen Integritätsbedingung zurückgewiesen wurden, ansonsten jedoch gültig sind. Sie können dieses Verhalten inaktivieren, indem Sie jeweils NORANGEEXC oder NOUNIQUEEXC mit der Ausnahmebedingung FOR EXCEPTION angeben. Wenn Sie angeben, dass diese Verletzungen der Bereichsvorgabe nicht in die Ausnahmetabelle eingefügt werden sollen, oder wenn Sie keine Ausnahmetabelle angeben, gehen alle Informationen zu Zeilen, die eine Bereichsvorgabe oder eine eindeutige Integritätsbedingung verletzen, verloren.

Protokolldatei

Bei partitionierten Zieltabellen enthält der entsprechende Eintrag in der Protokolldatei keine Liste der Tabellenbereiche, die die Zieltabelle umfasst. Eine andere Kennung für die Granularität von Operationen ('R' anstelle von 'T') gibt an, dass eine Ladeoperation für eine partitionierte Tabelle ausgeführt wurde.

Beenden einer Ladeoperation

Beim Beenden einer LOAD REPLACE-Operation werden alle sichtbaren Datenpartitionen vollständig abgeschnitten; beim Beenden einer LOAD INSERT-Operation werden alle sichtbaren Datenpartitionen auf ihre jeweilige Länge vor der Ladeoperation abgeschnitten. Beim Beenden einer ALLOW READ ACCESS-Ladeoperation, die in der LOAD COPY-Phase fehlschlug, werden die entsprechenden Indizes ungültig gemacht. Ein Index wird ebenfalls ungültig gemacht, wenn eine Ladeoperation mit der Option ALLOW NO ACCESS beendet wird, die den Index geändert hat. (Der Index wird ungültig gemacht, weil der Indexierungsmodus REBUILD (erneut erstellen) ist oder weil ein Schlüssel während der inkrementellen Verwaltung eingefügt wurde und den Index in einem inkonsistenten Status belassen hat). Das Laden von Daten in mehrere Ziele hat keinen Einfluss auf Recoveryoperationen, mit der Ausnahme, dass die Ladeoperation von einem Konsistenzpunkt, der während der LOAD-Phase erstellt wurde, nicht erneut gestartet werden kann. In diesem Fall wird die LOAD-Option `SAVECOUNT` ignoriert, wenn die Zieltabelle partitioniert ist. Dieses Verhalten entspricht dem Verhalten beim Laden von Daten in einer MDC-Zieltabelle.

Generierte Spalten

Befindet sich eine generierte Spalte in einem der Partitionierungs-, Dimensions- oder Verteilungsschlüssel, wird der Änderungswert `generatedoverride` für den Dateityp ignoriert, und das Dienstprogramm LOAD generiert Werte, als wäre der Änderungswert `generatedignore` für den Dateityp angegeben. Das Laden eines falschen Wertes einer generierten Spalte kann in diesem Fall dazu führen, dass der Datensatz in die falsche physische Position (wie beispielsweise in die falsche Datenpartition, den falschen MDC-Block oder die falsche Datenbankpartition) gestellt wird. Beispiel: Befindet sich ein Datensatz in einer falschen Datenpartition, muss die Operation SET INTEGRITY zum Festlegen der Integrität den Satz in eine andere physische Position versetzen, was im Verlauf von SET INTEGRITY-Operationen, die online ausgeführt werden, nicht bewerkstelligt werden kann.

Datenverfügbarkeit

Der aktuelle Algorithmus für ALLOW READ ACCESS-Ladeoperationen erstreckt sich auf partitionierte Tabellen. Eine ALLOW READ ACCESS-Ladeoperation ermöglicht den gleichzeitigen Lesezugriff auf die gesamte Tabelle. Dies schließt sowohl Ladedatenpartitionen als auch Nicht-Ladedatenpartitionen ein.

Wichtig: Ab Version 10.1 Fixpack 1 gilt der Parameter ALLOW READ ACCESS als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Weitere Informationen hierzu finden Sie im Abschnitt „Parameter ALLOW READ ACCESS im Befehl LOAD gilt als veraltet" in .

Das Dienstprogramm INGEST unterstützt auch partitionierte Tabellen und ist besser geeignet, den gleichzeitigen Zugriff auf Daten und die Verfügbarkeit der Daten zu ermöglichen, als der Befehl LOAD mit dem Parameter ALLOW READ ACCESS. Es kann zum Versetzen großer Datenvolumen aus Dateien und Pipes ohne Sperren der Zieltabelle verwendet werden. Darüber hinaus werden hierbei die Daten sofort nach dem Commit auf der Basis der verstrichenen Zeit bzw. der Anzahl der Zeilen zugänglich.

Datenpartitionsstatus

Nach einer erfolgreichen Ladeoperation können sichtbare Datenpartitionen entweder in den Tabellenstatus Set Integrity Pending oder Read Access Only oder beide wechseln. Dies hängt von bestimmten Bedingungen ab. So können Datenpartitionen in diese Status versetzt werden, wenn für die Tabelle Integritätsbedingungen bestehen, die von der Ladeoperation nicht verwaltet werden können. Dabei kann es sich z. B. um Integritätsbedingungen handeln, die Prüfungen auf Integritätsbedingungen und freigegebene MQTs (Materialized Query Table) beinhalten. Schlägt eine Ladeoperation fehl, verbleiben alle sichtbaren Datenpartitionen im Tabellenstatus Load Pending ('Laden anstehend').

Fehlerisolation

Eine Fehlerisolation auf Datenpartitionsebene wird nicht unterstützt. Das Isolieren von Fehlern bedeutet, Ladeoperationen für Datenpartitionen ohne Fehler fortzusetzen und Ladeoperationen für Datenpartitionen mit Fehlern zu stoppen. Fehler können in verschiedenen Datenbankpartitionen übergreifend isoliert werden. Das Dienstprogramm LOAD kann jedoch keine Transaktionen in einer Untergruppe von sichtbaren Datenpartitionen festschreiben und in den verbleibenden sichtbaren Datenpartitionen rückgängig machen.

Sonstige Aspekte

- Eine inkrementelle Indexierung wird nicht unterstützt, wenn einer der Indizes als ungültig markiert ist. Ein Index gilt als ungültig, wenn er erneut erstellt werden muss oder wenn freigegebene abhängige Objekte mit der Anweisung SET INTEGRITY überprüft werden müssen.
- Das Laden von Daten in Tabellen, deren Partitionierung anhand einer beliebigen Kombination aus Algorithmen für das Partitionieren nach Bereich, das Verteilen nach Hash oder das Organisieren nach Dimension erfolgte, wird ebenfalls unterstützt.
- Die Größe von Protokollsätzen, die eine Liste der von der Ladeoperation betroffenen Objekt- und Tabellenbereichs-IDs umfassen (Protokollsätze LOAD START und COMMIT (PENDING LIST)) kann stark anwachsen und so den für andere Anwendungen zur Verfügung stehenden aktiven Protokollspeicher reduzieren.
- Wenn eine Tabelle sowohl partitioniert als auch verteilt ist, kann es sein, dass sich eine Ladeoperation für partitionierte Datenbanken nicht auf alle Datenbankpartitionen auswirkt. Nur die Objekte in den Ausgabedatenbankpartitionen werden geändert.
- Während einer Ladeoperation nimmt die Speicherbelegung für partitionierte Tabellen mit der Anzahl der Tabellen zu. Bitte beachten Sie, dass der Gesamtanstieg nicht linear ist, da nur ein geringer Prozentsatz des Gesamtspeicherbedarfs proportional zur Anzahl der Datenpartitionen ist.

Laden von Daten in einer Umgebung mit partitionierten Datenbanken

Übersicht zum Laden - Umgebungen mit partitionierten Datenbanken

In einer Mehrpartitionsdatenbank befinden sich große Datenmengen auf vielen verschiedenen Datenbankpartitionen. Mit Verteilungsschlüsseln können Sie die Datenbankpartition festlegen, auf der die einzelnen Datenteile jeweils gespeichert werden sollen. Die Daten müssen *verteilt* werden, bevor sie auf die richtige Datenbankpartition geladen werden können.

Beim Laden von Tabellen in Mehrpartitionsdatenbanken stellt das Dienstprogramm LOAD das folgende Funktionsspektrum zur Verfügung:

- Paralleles Verteilen von Eingabedaten
- Gleichzeitiges Laden von Daten auf entsprechende Datenbankpartitionen
- Übertragen von Daten von einem System an ein anderes System

Das Laden von Daten in eine Mehrpartitionsdatenbank erfolgt in zwei Phasen: In einer *SETUP-Phase* werden Datenbankpartitionsressourcen wie beispielsweise Tabellensperren angefordert, und in einer *LOAD-Phase* werden die Daten in die Da-

tenbankpartitionen geladen. Mit der Option ISOLATE_PART_ERRS des Befehls **LOAD** können Sie auswählen, wie Fehler in diesen beiden Phasen jeweils verarbeitet werden und wie sich Fehler in einer oder mehreren der Datenbankpartitionen auf die Ladeoperation für diejenigen Datenbankpartitionen auswirken, die fehlerfrei sind.

Beim Laden von Daten in eine Mehrpartitionsdatenbank können die folgenden Modi verwendet werden:

PARTITION_AND_LOAD
Die Daten werden verteilt (möglicherweise in Parallelverarbeitung) und gleichzeitig auf die entsprechenden Datenbankpartitionen geladen.

PARTITION_ONLY
Die Daten werden verteilt (möglicherweise in Parallelverarbeitung), und die Ausgabe wird in Dateien an einer angegebenen Speicherposition auf jeder Ladedatenbankpartition geschrieben. Jede Datei enthält Partitionskopfdaten, die angeben, wie die Daten über die Datenbankpartitionen verteilt wurden, und dass die Datei unter Verwendung des Modus LOAD_ONLY in die Datenbank geladen werden kann.

LOAD_ONLY
Es wird davon ausgegangen, dass die Daten bereits über die Datenbankpartitionen verteilt sind. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen.

LOAD_ONLY_VERIFY_PART
Es wird davon ausgegangen, dass die Daten bereits über die Datenbankpartitionen verteilt sind, aber die Datendatei enthält keine Partitionskopfdaten. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen. Während der Ladeoperation wird für jede Zeile geprüft, ob sie sich auf der korrekten Datenbankpartition befindet. Zeilen, die Datenbankpartitionsverstöße enthalten, werden in eine Speicherauszugsdatei gestellt, sofern der Änderungswert `dumpfile` für den Dateityp angegeben wurde. Andernfalls werden die Zeilen gelöscht. Wenn für eine bestimmte Ladedatenbankpartition Datenbankpartitionsverstöße vorliegen, wird für die betreffende Datenbankpartition eine einzige Warnung in die LOAD-Nachrichtendatei geschrieben.

ANALYZE
Es wird eine optimale Verteilungszuordnung mit einer gleichmäßigen Verteilung auf alle Datenbankpartitionen generiert.

Konzepte und Terminologie

Im Zusammenhang mit der Funktionsweise und der Ausführung des Dienstprogramms LOAD in einer Umgebung mit partitionierten Datenbanken mit mehreren Datenbankpartitionen wird die folgende Terminologie verwendet:

- Die *Koordinatorpartition* ist die Datenbankpartition, zu der der Benutzer eine Verbindung herstellt, um die Ladeoperation auszuführen. In den Modi PARTITION_AND_LOAD, PARTITION_ONLY und ANALYZE wird davon ausgegangen, dass sich die Datendatei auf dieser Datenbankpartition befindet, sofern nicht die Option `CLIENT` des Befehls **LOAD** angegeben ist. Mit der Option `CLIENT` wird angegeben, dass sich die zu ladenden Daten auf einem Client mit Fernverbindung befinden.
- In den Modi PARTITION_AND_LOAD, PARTITION_ONLY und ANALYZE liest der *Agent für Partitionierungsvorbereitung* die Benutzerdaten und teilt sie reihum

den *Partitionierungsagenten* zu, die die Daten verteilen. Dieser Prozess findet immer auf der Koordinatorpartition statt. Bei allen Ladeoperationen ist pro Datenbankpartition maximal ein Partitionierungsagent zulässig.

- In den Modi PARTITION_AND_LOAD, LOAD_ONLY und LOAD_ONLY_VERIFY_PART werden auf jeder Ausgabedatenbankpartition *Ladeagenten* ausgeführt. Sie koordinieren das Laden der Daten auf diese Datenbankpartition.
- *Dateiladeagenten* werden während einer Ladeoperation im Modus PARTITION_ONLY auf jeder Ausgabedatenbankpartition ausgeführt. Sie empfangen Daten von Partitionierungsagenten und schreiben sie in eine Datei auf ihrer Datenbankpartition.
- Die Option `SOURCEUSEREXIT` stellt eine Funktion bereit, über die das Dienstprogramm LOAD ein angepasstes Script bzw. eine angepasste ausführbare Datei ausführen kann. Dieses Script bzw. diese Datei wird im vorliegenden Handbuch als *Benutzerexit* bezeichnet.

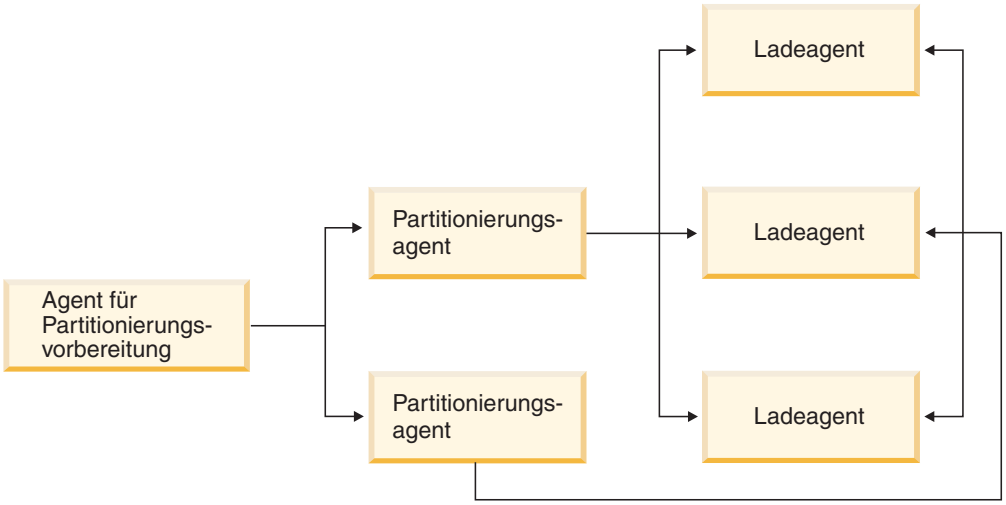


Abbildung 39. Übersicht über das Laden in partitionierte Datenbanken. Die Quellendaten werden durch den Agenten für Partitionierungsvorbereitung gelesen. Ungefähr die Hälfte der Daten wird jeweils an zwei Partitionierungsagenten gesendet, die die Daten verteilen und an eine der drei Datenbankpartitionen senden. Auf den einzelnen Datenbankpartitionen werden die Daten durch den jeweiligen Ladeagenten geladen.

Hinweise und Tipps für das Laden von Daten in einer Umgebung mit partitionierten Datenbanken

Bevor Sie eine Tabelle in eine Mehrpartitionsdatenbank laden, sollten Sie die folgenden Punkte beachten:

- Machen Sie sich mit den Konfigurationsoptionen des Dienstprogramms LOAD vertraut, indem Sie das Dienstprogramm mit kleinen Datenmengen verwenden.
- Wenn die Eingabedaten bereits sortiert sind oder in einer bestimmten Reihenfolge vorliegen und Sie diese Reihenfolge während des Ladeprozesses beibehalten möchten, darf nur eine einzige Datenbankpartition zum Verteilen verwendet werden. Bei paralleler Verteilung kann nicht sichergestellt werden, dass die Daten in derselben Reihenfolge geladen werden, in der sie empfangen wurden. Das Dienstprogramm LOAD wählt standardmäßig einen einzigen Partitionierungsagenten aus, wenn der Änderungswert `anyorder` im Befehl **`LOAD`** nicht angegeben wird.
- Wenn große Objekte (LOBs) aus separaten Dateien geladen werden (wenn Sie also den Änderungswert `lobsinfile` über das Dienstprogramm LOAD verwen-

den), müssen alle Datenbankpartitionen, auf denen Ladevorgänge stattfinden, Lesezugriff auf alle Verzeichnisse mit LOB-Dateien haben. Der LOAD-Parameter *lob-pfad* muss beim Arbeiten mit LOBs vollständig qualifiziert sein.

- Sie können die Fortsetzung eines Jobs, der in einer Mehrpartitionsdatenbank ausgeführt wird, erzwingen, selbst wenn die Ladeoperation (beim Start) feststellt, dass sich einige Ladedatenbankpartitionen oder zugeordnete Tabellenbereiche oder Tabellen im Offlinemodus befinden. Hierzu setzen Sie die Option ISOLATE_PART_ERRS auf SETUP_ERRS_ONLY oder SETUP_AND_LOAD_ERRS.
- Mit der LOAD-Konfigurationsoption STATUS_INTERVAL kann der Status eines Jobs überwacht werden, der in einer Mehrpartitionsdatenbank ausgeführt wird. Die Ladeoperation gibt in den angegebenen Intervallen Nachrichten zurück, aus denen ersichtlich ist, wie viele Megabyte Daten durch den Agenten für die Partitionierungsvorbereitung gelesen wurden. Diese Nachrichten werden in der Nachrichtendatei des Agenten für die Partitionierungsvorbereitung gespeichert. Um den Inhalt dieser Datei während der Ladeoperation anzuzeigen, stellen Sie eine Verbindung zur Koordinatorpartition her und setzen den Befehl **LOAD QUERY** für die Zieltabelle ab.
- Sie können mit besserer Leistung rechnen, wenn sich die am Verteilungsprozess beteiligten Datenbankpartitionen (durch die Option PARTITIONING_DBPARTNUMS definiert) von den Ladedatenbankpartitionen (durch die Option OUTPUT_DBPARTNUMS definiert) unterscheiden, da dann weniger Konkurrenzsituationen für CPU-Zyklen auftreten. Wenn Daten in eine Mehrpartitionsdatenbank geladen werden, sollte das Dienstprogramm LOAD auf einer Datenbankpartition aufgerufen werden, die weder an der Verteilungs- noch an der Ladeoperation beteiligt ist.
- Bei Angabe des Parameters MESSAGES im Befehl **LOAD** werden die Nachrichtendateien der Agenten für die Partitionierungsvorbereitung, der Partitionierungs- und der Ladeagenten am Ende der Ladeoperation zu Referenzzwecken gespeichert. Um den Inhalt dieser Dateien während einer Ladeoperation anzuzeigen, stellen Sie eine Verbindung zu der entsprechenden Datenbankpartition her und setzen einen Befehl **LOAD QUERY** für die Zieltabelle ab.
- Das Dienstprogramm LOAD wählt nur eine Ausgabedatenbankpartition für die Erfassung von Statistikdaten aus. Mit der Datenbankkonfigurationsoption RUN_STAT_DBPARTNUM können Sie die Datenbankpartition angeben.
- Führen Sie vor dem Laden von Daten in eine Mehrpartitionsdatenbank den Designadvisor aus, um die jeweils beste Partition für die einzelnen Tabellen zu ermitteln. Weitere Informationen hierzu finden Sie in „Designadvisor" in *Fehlerbehebung und Optimieren der Datenbankleistung*.

Fehlerbehebung

Wenn das Dienstprogramm LOAD blockiert, haben Sie folgende Möglichkeiten:

- Mit dem Parameter STATUS_INTERVAL kann der Status einer Ladeoperation für Mehrpartitionsdatenbanken überwacht werden. Die Statusintervalldaten werden in der Nachrichtendatei des Agenten für die Partitionierungsvorbereitung auf der Koordinatorpartition gespeichert.
- Überprüfen Sie in der Nachrichtendatei des Partitionierungsagenten den Status der Prozesse des Partitionierungsagenten auf den einzelnen Datenbankpartitionen. Wenn das Laden ohne Fehler fortgesetzt wird und die Option TRACE definiert wurde, sollten diese Nachrichtendateien Tracenachrichten für eine Reihe von Datensätzen enthalten.
- Überprüfen Sie, ob die LOAD-Nachrichtendatei Fehlernachrichten enthält.

Anmerkung: Damit diese Dateien vorhanden sind, müssen Sie die Option MESSAGES des Befehls **LOAD** angeben.

- Unterbrechen Sie die aktuelle Ladeoperation, wenn Sie Fehler finden, die darauf schließen lassen, dass bei einem der Ladeprozesse Fehler aufgetreten sind.

Laden von Daten in einer Umgebung mit partitionierten Datenbanken

Dienstprogramm LOAD zum Laden von Daten in eine Umgebung mit partitionierten Datenbanken verwenden

Vorbereitende Schritte

Bevor Sie eine Tabelle in eine Mehrpartitionsdatenbank laden, sollten Sie die folgenden Punkte beachten:

- Stellen Sie sicher, dass der Konfigurationsparameter **svcename** des Datenbankmanagers und die Profilregistrierdatenbankvariable **DB2COMM** richtig definiert sind. Dieser Schritt ist wichtig, weil das Dienstprogramm LOAD TCP/IP verwendet, um Daten von den Agenten für Partitionierungsvorbereitung an die Partitionierungsagenten und von den Partitionierungsagenten an die Ladedatenbankpartitionen zu übertragen.
- Bevor Sie das Dienstprogramm LOAD aufrufen, müssen Sie mit der Datenbank verbunden sein, in die die Daten geladen werden sollen, oder in der Lage sein, implizit eine Verbindung zu dieser Datenbank herzustellen.
- Da das Dienstprogramm LOAD eine Anweisung COMMIT absetzt, sollten Sie vor dem Beginn der Ladeoperation alle Transaktionen beenden und alle Sperren aufheben, indem Sie eine Anweisung COMMIT oder ROLLBACK absetzen. Wird der Modus PARTITION_AND_LOAD, PARTITION_ONLY oder ANALYZE verwendet, muss sich die geladene Datendatei auf dieser Datenbankpartition befinden. Ausgenommen sind die folgenden Fälle:
  1. Der Parameter **CLIENT** wurde angegeben. In diesem Fall müssen sich die Daten auf der Clientmaschine befinden.
  2. Der Eingabequellentyp ist CURSOR. In diesem Fall gibt es keine Eingabedatei.
- Führen Sie den **Designadvisor** aus, um die jeweils beste Datenbankpartition für die einzelnen Tabellen zu ermitteln. Weitere Informationen hierzu finden Sie in „Designadvisor" in *Fehlerbehebung und Optimieren der Datenbankleistung*.

Einschränkungen

Bei der Verwendung des Dienstprogramms LOAD zum Laden von Daten in eine Mehrpartitionsdatenbank gelten die folgenden Einschränkungen:

- Die Eingabedateien für die Ladeoperation dürfen nicht auf einer Bandeinheit gespeichert sein.
- Der Parameter **ROWCOUNT** wird nur dann unterstützt, wenn der Modus ANALYZE verwendet wird.
- Wenn die Zieltabelle über eine Identitätsspalte verfügt, die zur Verteilung notwendig ist, und der Änderungswert **identityoverride** für den Dateityp nicht angegeben ist, oder wenn Sie mehrere Datenbankpartitionen verwenden, um die Daten zu verteilen und anschließend zu laden, wird ein Wert für **SAVECOUNT**, der größer als 0 ist, im Befehl **LOAD** nicht unterstützt.
- Wenn eine Identitätsspalte einen Teil des Verteilungsschlüssels darstellt, wird nur der Modus PARTITION_AND_LOAD unterstützt.

- Der Modus LOAD_ONLY und der Modus LOAD_ONLY_VERIFY_PART können nicht mit dem Parameter **CLIENT** des Befehls **LOAD** kombiniert werden.
- Der Modus LOAD_ONLY_VERIFY_PART kann nicht mit dem Eingabequellentyp CURSOR verwendet werden.
- Die Modi LOAD_ERRS_ONLY und SETUP_AND_LOAD_ERRS für die Isolation von Verteilungsfehlern können nicht zusammen mit den Parametern **ALLOW READ ACCESS** und **COPY YES** des Befehls **LOAD** verwendet werden.
- Mehrere Ladeoperationen können gleichzeitig Daten in dieselbe Tabelle laden, wenn sich die durch die Optionen **OUTPUT_DBPARTNUMS** und **PARTITIONING_DBPARTNUMS** angegebenen Datenbankpartitionen nicht überlappen. Beispiel: Wenn eine Tabelle für die Datenbankpartitionen 0 bis 3 definiert ist, kann eine Ladeoperation Daten in die Datenbankpartitionen 0 und 1 laden, während eine zweite Ladeoperation Daten in die Datenbankpartitionen 2 und 3 lädt. Wenn die von den **PARTITIONING_DBPARTNUMS**-Optionen angegebenen Datenbankpartitionen überlappen, wählt die Ladeoperation automatisch einen Parameter **PARTITIONING_DBPARTNUMS** aus, bei dem noch kein Ladepartitionierungssubagent für die Tabelle ausgeführt wird, oder sie schlägt fehl, wenn keiner verfügbar ist.

  Ab Version 9.7 Fixpack 6 gilt, dass bei Überlappen der von den **PARTITIONING_DBPARTNUMS**-Optionen angegebenen Datenbankpartitionen das Dienstprogramm LOAD automatisch versucht, einen Parameter **PARTITIONING_DBPARTNUMS** aus den Datenbankpartitionen auszuwählen, die durch **OUTPUT_DBPARTNUMS** angegeben werden und bei denen noch kein Ladepartitionierungsagent für die Tabelle ausgeführt wird, oder es schlägt fehlt, wenn keiner verfügbar ist.

  Wenn Sie mithilfe der Option **PARTITIONING_DBPARTNUMS** Partitionen explizit angeben, wird sehr empfohlen, dass Sie diese Option mit allen gleichzeitig ablaufenden **LOAD**-Befehlen verwenden, wobei jeder Befehl unterschiedliche Partitionen angibt. Wenn Sie nur für einige der gleichzeitig ablaufenden LOAD-Befehle **PARTITIONING_DBPARTNUMS** angeben oder wenn Sie überlappende Partitionen angeben, muss der Befehl **LOAD** für zumindest einige der gleichzeitig ablaufenden Ladevorgänge alternierende Partitionierungsknoten auswählen und in seltenen Fällen kann der Befehl fehlschlagen (SQL2038N).
- Nur ASC-Dateien (ASCII-Format mit universellen Zeilenbegrenzern) und DEL-Dateien (ASCII-Format ohne universelle Zeilenbegrenzer) können über Tabellen auf mehreren Datenbankpartitionen verteilt werden. PC/IXF-Dateien können nicht verteilt werden, Sie können eine PC/IXF-Datei jedoch in eine über mehrere Datenbankpartitionen verteilte Tabelle laden, indem Sie die Ladeoperation im Modus LOAD_ONLY_VERIFY_PART verwenden.

Beispiel

Die folgenden Beispiele veranschaulichen, wie Sie mit dem Befehl **LOAD** unterschiedliche Typen von Ladeoperationen einleiten können. Die in den folgenden Beispielen verwendete Datenbank enthält fünf Datenbankpartitionen: 0, 1, 2, 3 und 4. Jede Datenbankpartition verfügt über das lokale Verzeichnis /db2/data/. Die beiden Tabellen TABLE1 und TABLE2 sind auf den Datenbankpartitionen 0, 1, 3 und 4 definiert. Beim Laden von einem Client aus greift der Benutzer auf einen fernen Client zu, bei dem es sich nicht um eine der Datenbankpartitionen handelt.

Beispiel für Verteilen und Laden

In diesem Szenario besteht eine Verbindung zu einer Datenbankpartition, auf der die Tabelle TABLE1 definiert sein kann oder auch nicht. Die Datendatei load.del befindet sich im aktuellen Arbeitsverzeichnis dieser Daten-

bankpartition. Um die Daten aus der Datei `load.del` auf alle Datenbankpartitionen zu laden, auf denen die Tabelle TABLE1 definiert ist, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

Anmerkung: In diesem Beispiel werden für alle Konfigurationsparameter für Umgebungen mit partitionierten Datenbanken die Standardwerte verwendet: Der Parameter **MODE** verwendet den Standardwert PARTITION_AND-_LOAD. Der Parameter **OUTPUT_DBPARTNUMS** verwendet standardmäßig alle Datenbankpartitionen, für die TABLE1 definiert ist. Der Parameter **PARTITIONING_DBPARTNUMS** verwendet standardmäßig die Gruppe der Datenbankpartitionen, die auf der Basis der Regeln für den Befehl **LOAD** zur Auswahl von Datenbankpartitionen ausgewählt werden, wenn keine Angabe erfolgt.

Um eine Ladeoperation auszuführen, bei der Daten über die Datenbankpartitionen 3 und 4 verteilt werden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
 PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

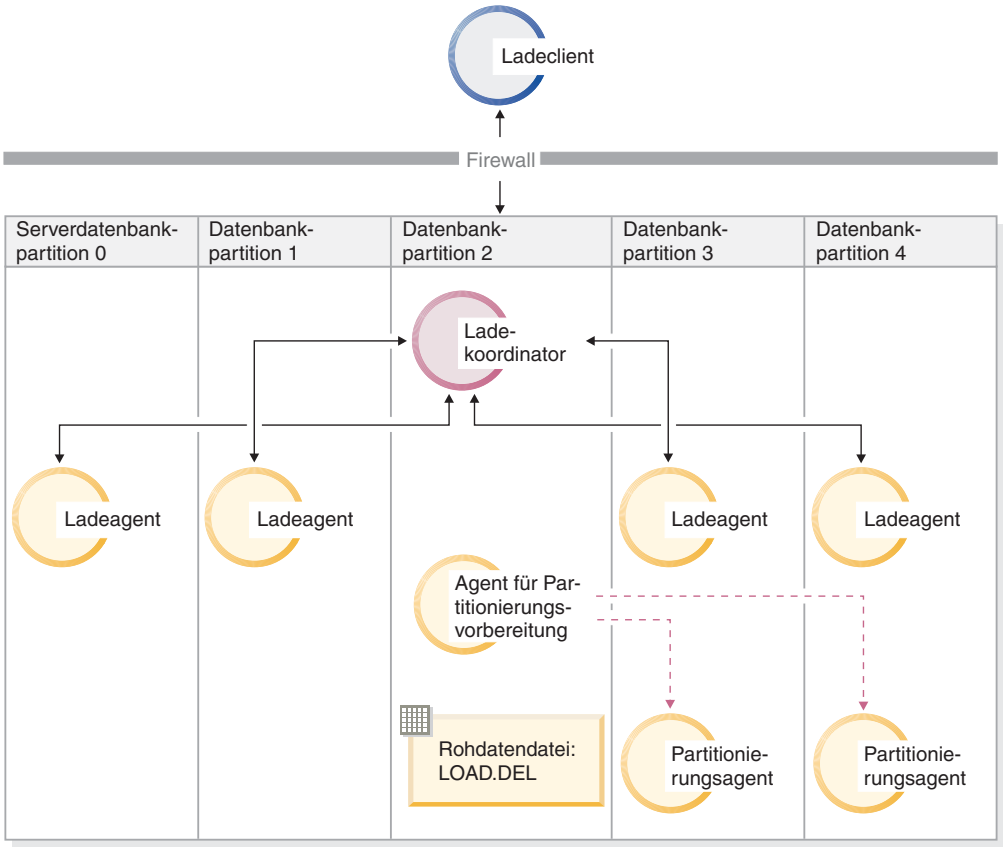


Abbildung 40. Laden von Daten in die Datenbankpartitionen 3 und 4. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die Daten werden in die Datenbankpartitionen 3 und 4 geladen.

Beispiel für Verteilen ohne Laden

In diesem Szenario besteht eine Verbindung zu einer Datenbankpartition, auf der die Tabelle TABLE1 definiert sein kann oder auch nicht. Die Datendatei `load.del` befindet sich im aktuellen Arbeitsverzeichnis dieser Datenbankpartition. Um die Datei `load.del` auf alle Datenbankpartitionen, auf denen TABLE1 definiert ist, zu verteilen (jedoch nicht zu laden) und dafür die Datenbankpartitionen 3 und 4 zu verwenden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE PARTITION_ONLY
 PART_FILE_LOCATION /db2/data
 PARTITIONING_DBPARTNUMS (3,4)
```

Dieser Befehl bewirkt, dass eine Datei mit dem Namen `load.del.`*xxx* im Verzeichnis `/db2/data` auf jeder Datenbankpartition gespeichert wird. Hierbei steht *xxx* für eine dreistellige Darstellung der Datenbankpartitionsnummer.

Um die Datei `load.del` auf die Datenbankpartitionen 1 und 3 zu verteilen und hierfür nur einen auf der Partition 0 aktiven Partitionierungsagenten zu verwenden (Standardwert für **PARTITIONING_DBPARTNUMS**), verwenden Sie den folgenden Befehl:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE PARTITION_ONLY
 PART_FILE_LOCATION /db2/data
 OUTPUT_DBPARTNUMS (1,3)
```

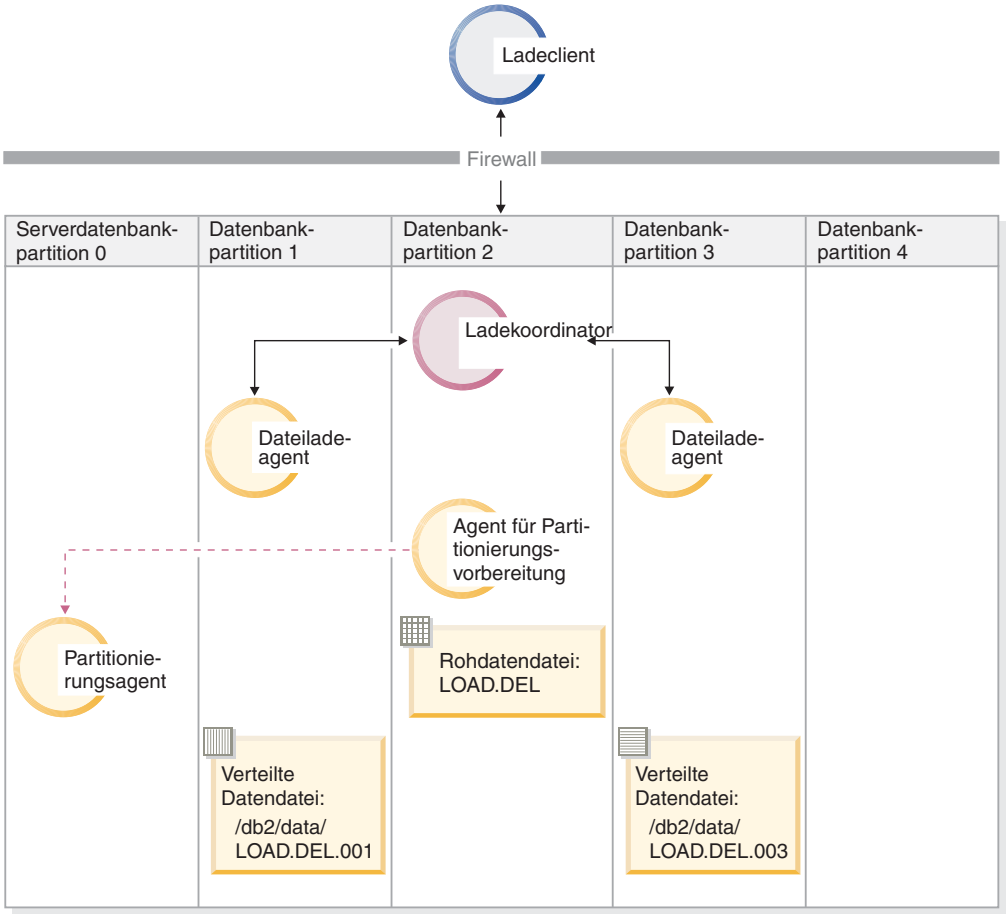


Abbildung 41. Laden von Daten in die Datenbankpartitionen 1 und 3 mithilfe eines Partitionierungsagenten. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die Daten werden auf die Datenbankpartitionen 1 und 3 geladen, wobei ein Partitionierungsagent verwendet wird, der auf Datenbankpartition 0 aktiv ist.

Beispiel für Laden ohne Verteilen

Wenn Sie bereits eine Ladeoperation im Modus `PARTITION_ONLY` ausgeführt haben und die partitionierten Dateien im Verzeichnis `/db2/data` jeder Ladedatenbankpartition auf alle Datenbankpartitionen laden wollen, auf denen TABLE1 definiert ist, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
 PART_FILE_LOCATION /db2/data
```

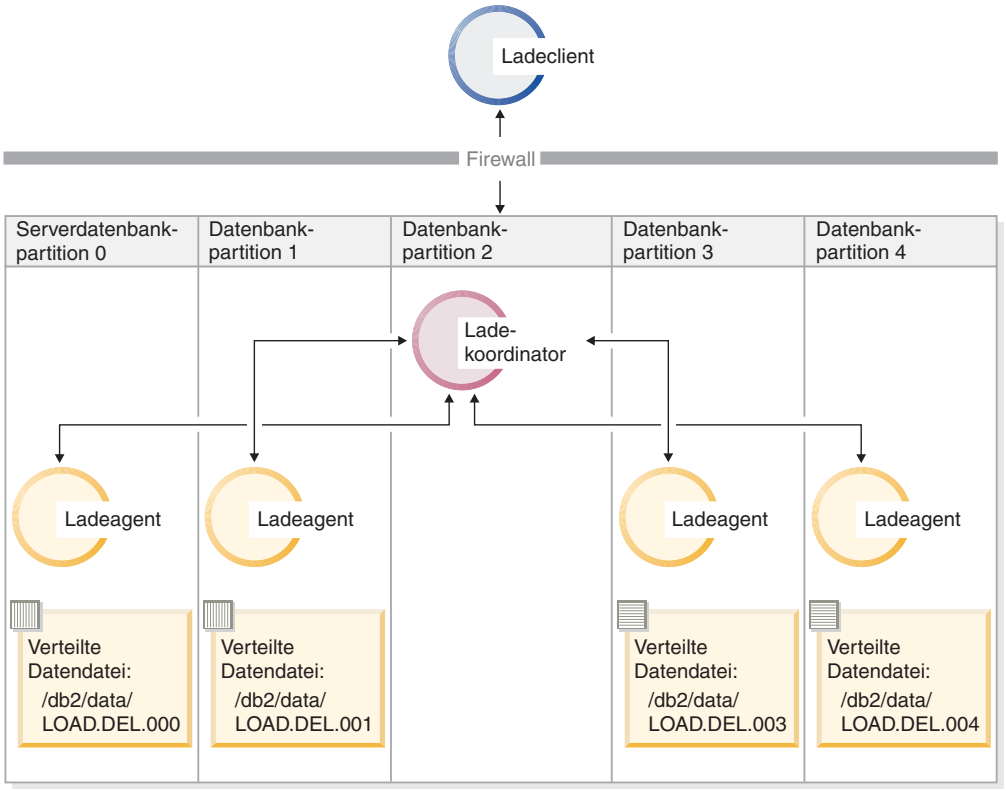


Abbildung 42. Laden von Daten in alle Datenbankpartitionen mit einer bestimmten definierten Tabelle. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die verteilten Daten werden auf alle Datenbankpartitionen geladen, auf denen TABLE1 definiert ist.

Um Daten nur auf Datenbankpartition 4 zu laden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
 PART_FILE_LOCATION /db2/data
 OUTPUT_DBPARTNUMS (4)
```

Laden von vorverteilten Dateien ohne Kopfdaten für Verteilungszuordnung

Mit dem Befehl **LOAD** können Sie Datendateien ohne Verteilungskopfdaten direkt auf mehrere Datenbankpartitionen laden. Wenn die Datendateien im Verzeichnis `/db2/data` aller Datenbankpartitionen, auf denen TABLE1 definiert ist, bereits vorhanden sind und ihr Name `load.del.`*xxx* lautet (hierbei ist *xxx* die Datenbankpartitionsnummer), können die Dateien mit dem folgenden Befehl geladen werden:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
 PART_FILE_LOCATION /db2/data
```

Um die Daten nur auf Datenbankpartition 1 zu laden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
 PART_FILE_LOCATION /db2/data
 OUTPUT_DBPARTNUMS (1)
```

Anmerkung: Zeilen, die nicht zu der Datenbankpartition gehören, von der aus sie geladen wurden, werden zurückgewiesen und in die Speicherauszugsdatei gestellt, sofern eine Speicherauszugsdatei angegeben wurde.

Laden von einem fernen Client in eine Mehrpartitionsdatenbank

Um Daten aus einer Datei, die sich auf einem fernen Client befindet, in eine Mehrpartitionsdatenbank zu laden, müssen Sie den Parameter **CLIENT** des Befehls **LOAD** angeben. Dieser Parameter gibt an, dass die Datendatei sich nicht auf einer Serverpartition befindet. Beispiel:

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

Anmerkung: Der Modus LOAD_ONLY oder LOAD_ONLY_VERIFY_PART kann nicht zusammen mit dem Parameter **CLIENT** verwendet werden.

Laden über einen Cursor

Wie bei Einzelpartitionsdatenbanken können Sie Daten über einen Cursor in eine Mehrpartitionsdatenbank laden. In diesem Beispiel muss bei den Modi PARTITION_ONLY und LOAD_ONLY der Parameter **PART_FILE_LOCATION** einen vollständig qualifizierten Dateinamen angeben. Dieser Name ist der vollständig qualifizierte Basisdateiname der verteilten Dateien, die auf jeder Ausgabedatenbankpartition erstellt oder geladen werden. Wenn die Zieltabelle LOB-Spalten enthält, können mehrere Dateien mit dem angegebenen Basisnamen erstellt werden.

Um alle Zeilen in der Antwortgruppe der Anweisung SELECT * FROM TABLE1 auf eine Datei auf allen Datenbankpartitionen mit dem Namen /db2/data/select.out.*xxx* zu verteilen (*xxx* ist die Datenbankpartitionsnummer), damit diese Zeilen später in die Tabelle TABLE2 geladen werden können, setzen Sie die folgenden Befehle ab:

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
PARTITIONED DB CONFIG MODE PARTITION_ONLY
 PART_FILE_LOCATION /db2/data/select.out
```

Die durch die vorherige Operation erzeugten Datendateien können anschließend durch Absetzen des folgenden Befehls **LOAD** geladen werden:

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
PARTITIONED CB CONFIG MODE LOAD_ONLY
 PART_FILE_LOCATION /db2/data/select.out
```

Überwachen einer Ladeoperation in einer Umgebung mit partitionierten Datenbanken mit LOAD QUERY

Während einer Ladeoperation in einer Umgebung mit partitionierten Datenbanken werden von einigen Ladeprozessen Nachrichtendateien auf den Datenbankpartitionen erstellt, auf denen sie ausgeführt werden.

In den Nachrichtendateien werden alle Informationen, Warnungen und Fehlernachrichten gespeichert, die während der Ausführung der Ladeoperation erzeugt wurden. Bei den Ladeprozessen, die vom Benutzer anzeigbare Nachrichtendateien erzeugen, handelt es sich um den Ladeagenten, den Agenten für die Partitionierungsvorbereitung und den Partitionierungsagenten. Der Inhalt der Nachrichtendatei steht erst nach Abschluss der Ladeoperation zur Verfügung.

Sie können während einer Ladeoperation eine Verbindung zu einzelnen Datenbankpartitionen herstellen und den Befehl **LOAD QUERY** für die Zieltabelle absetzen. Wird dieser Befehl über den Befehlszeilenprozessor (CLP) abgesetzt, zeigt er den

Inhalt der Nachrichtendateien an, die sich gegenwärtig für die im Befehl **LOAD QUERY** angegebene Tabelle auf dieser Datenbankpartition befinden.

Beispiel: Die Tabelle TABLE1 ist auf den Datenbankpartitionen 0 bis 3 in der Datenbank WSDB definiert. Sie sind mit Datenbankpartition 0 verbunden und setzen den folgenden Befehl **LOAD** ab:

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

Dieser Befehl leitet eine Ladeoperation ein, die das Ausführen von Ladeagenten auf den Datenbankpartitionen 0, 1, 2 und 3 umfasst. Außerdem wird ein Partitionierungsagent auf Datenbankpartition 1 und ein Agent für Partitionierungsvorbereitung auf Datenbankpartition 0 ausgeführt.

Datenbankpartition 0 enthält eine Nachrichtendatei für den Agenten für Partitionierungsvorbereitung und eine Nachrichtendatei für den Ladeagenten auf dieser Datenbankpartition. Um den Inhalt dieser Dateien gleichzeitig anzuzeigen, starten Sie eine neue Sitzung und setzen Sie die folgenden Befehle am Befehlszeilenprozessor ab:

```
set client connect_node 0
connect to wsdb
load query table table1
```

Datenbankpartition 1 enthält eine Datei für den Ladeagenten und eine Datei für den Partitionierungsagenten. Um den Inhalt dieser Dateien anzuzeigen, starten Sie eine neue Sitzung, und setzen Sie die folgenden Befehle am Befehlszeilenprozessor ab:

```
set client connect_node 1
connect to wsdb
load query table table1
```

Anmerkung: Die durch die LOAD-Konfigurationsoption STATUS_INTERVAL generierten Nachrichten werden in der Nachrichtendatei des Agenten für die Partitionierungsvorbereitung angezeigt. Um diese Nachrichten während einer Ladeoperation anzuzeigen, müssen Sie eine Verbindung zur Koordinatorpartition herstellen und den Befehl **LOAD QUERY** absetzen.

Sichern des Inhalts von Nachrichtendateien

Wenn eine Ladeoperation über die API **'db2Load'** eingeleitet wird, muss die Nachrichtenoption (piLocalMsgFileName) angegeben werden. Dann werden die Nachrichtendateien vom Server an den Client übertragen und dort gespeichert, damit Sie die Nachrichten anzeigen können.

Bei Ladeoperationen für Mehrpartitionsdatenbanken, die über den Befehlszeilenprozessor (CLP) eingeleitet werden, werden die Nachrichtendateien nicht auf der Konsole angezeigt oder beibehalten. Um den Inhalt dieser Dateien zu sichern oder anzuzeigen, nachdem das Laden in eine Mehrpartitionsdatenbank abgeschlossen ist, muss die Option MESSAGES des Befehls **LOAD** angegeben werden. Bei Verwendung dieser Option werden die Nachrichtendateien auf jeder Datenbankpartition nach Abschluss der Ladeoperation an die Clientmaschine übertragen und dort unter dem Basisnamen in Dateien gespeichert, der in der Option MESSAGES angegeben wurde. Die folgende Tabelle enthält die Dateinamen, die dem Ladeprozess, der die jeweilige Datei erstellt hat, entsprechen (bei Ladeoperationen für Mehrpartitionsdatenbanken):

Prozesstyp	Dateiname
Ladeagent	<*nachrichtendateiname*>.LOAD.<*dbpartitionsnr*>
Partitionierungsagent	<*nachrichtendateiname*>.PART.<*dbpartitionsnr*>
Agent für Partitionierungsvorbereitung	<*nachrichtendateiname*>.PREP.<*dbpartitionsnr*>

Wenn in der Option MESSAGES beispielsweise der Wert /wsdb/messages/load angegeben ist, lautet der Name der Nachrichtendatei des Ladeagenten auf Datenbankpartition 2 /wsdb/messages/load.LOAD.002.

Anmerkung: Es wird dringend empfohlen, bei Ladeoperationen für Mehrpartitionsdatenbanken, die über den Befehlszeilenprozessor (CLP) eingeleitet werden, die Option MESSAGES zu verwenden.

Fortsetzen, erneutes Starten oder Beenden von Ladeoperationen in einer Umgebung mit partitionierten Datenbanken

Die Arbeitsschritte, die in einer Umgebung mit partitionierten Datenbanken nach fehlgeschlagenen Ladeoperationen ausgeführt werden müssen, hängen davon ab, wann der Fehler aufgetreten ist.

Der Ladeprozess in einer Mehrpartitionsdatenbank besteht aus zwei Phasen:

1. In der *SETUP-Phase* werden Ressourcen auf Datenbankpartitionsebene angefordert. Hierzu gehören z. B. Tabellensperren für Datenbankpartitionen, die für die Ausgabe verwendet werden.

   Wenn während der SETUP-Phase ein Fehler auftritt, sind in der Regel keine Neustart- oder Beendigungsoperationen erforderlich. Die erforderlichen Maßnahmen hängen von dem Fehlerisolationsmodus ab, der für die fehlgeschlagene Ladeoperation angegeben wurde.

   Wenn für die Ladeoperation angegeben wurde, dass Fehler während der SETUP-Phase nicht isoliert werden sollen, wird die gesamte Ladeoperation abgebrochen und der Status der Tabelle wird auf allen Datenbankpartitionen mittels Rollback auf den Status zurückgesetzt, der vor der Ladeoperation gültig war.

   Wenn für die Ladeoperation angegeben wurde, dass Fehler während der SETUP-Phase isoliert werden sollen, wird die Ladeoperation auf den Datenbankpartitionen fortgesetzt, auf denen die SETUP-Phase erfolgreich ausgeführt werden konnte. Die Tabellen auf den Datenbankpartitionen, auf denen ein Fehler auftrat, werden mittels Rollback auf den Status zurückgesetzt, der vor der Ladeoperation gültig war. Dies bedeutet, dass eine einzige Ladeoperation in unterschiedlichen Phasen fehlschlagen kann, wenn einige Partitionen während der SETUP-Phase und andere während der LOAD-Phase fehlschlagen.
2. Die *LOAD-Phase*, während der die Daten formatiert und in Tabellen auf den Datenbankpartitionen geladen werden.

   Wenn eine Ladeoperation für eine Mehrpartitionsdatenbank während der LOAD-Phase auf mindestens einer Datenbankpartition fehlschlägt, muss ein Befehl **LOAD RESTART** oder ein Befehl **LOAD TERMINATE** abgesetzt werden. Dies ist notwendig, weil das Laden von Daten in eine Mehrpartitionsdatenbank in einer einzigen Transaktion erfolgt.

   Wählen Sie einen Befehl **LOAD RESTART** aus, wenn Sie die Probleme, die zum Fehlschlagen der Ladeoperation führten, beheben können. Dies spart Zeit. Wenn eine LOAD RESTART-Operation eingeleitet wird, wird die Ladeoperation auf allen Datenbankpartitionen dort fortgesetzt, wo sie abgebrochen wurde.

Wählen Sie einen Befehl **LOAD TERMINATE** aus, wenn die Tabelle in den Status zurückversetzt werden soll, in der sie sich vor der ursprünglichen Ladeoperation befand.

Ermitteln des Fehlerzeitpunkts einer Ladeoperation

Wenn Ihre Ladeoperation in einer partitionierten Umgebung fehlschlägt, müssen Sie zuerst ermitteln, auf welchen Partitionen der Fehler aufgetreten ist und in welcher Phase die Verarbeitung fehlschlug. Hierzu müssen Sie die Zusammenfassung für die Partition überprüfen. Wenn der Befehl **LOAD** über den Befehlszeilenprozessor (CLP) eingegeben wurde, wird die Partitionszusammenfassung am Ende des Ladevorgangs angezeigt (siehe folgendes Beispiel). Wenn der Befehl **LOAD** über die API 'db2Load' eingegeben wurde, dann ist die Partitionszusammenfassung im Feld **poAgentInfoList** der Struktur 'db2PartLoadOut' enthalten.

Wenn für "Agent Typ" für eine bestimmte Partition ein Eintrag "LOAD" vorhanden ist, hat diese Partition die LOAD-Phase erreicht. Andernfalls trat der Fehler während der SETUP-Phase auf. Ein negativer SQL-Code gibt an, dass ein Fehler aufgetreten ist. Im folgenden Beispiel schlug die Ladeoperation in Partition 1 während der LOAD-Phase fehl.

```
    Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________________
    LOAD           000      +00000000    Erfolg.
_____________________________________________________________
    LOAD           001      -00000289    Fehler. Möglicherweise
Neustart (RESTART) erforderlich.
_____________________________________________________________
    LOAD           002      +00000000    Erfolg.
_____________________________________________________________
    LOAD           003      +00000000    Erfolg.
.
.
.
```

Fortsetzen, erneutes Starten oder Beenden von fehlgeschlagenen Ladeoperationen

Nur Ladeoperationen mit der Option ISOLATE_PART_ERRS, für die SETUP_ERRS_ONLY oder SETUP_AND_LOAD_ERRS angegeben wurde, können während der SETUP-Phase fehlschlagen. Bei Ladeoperationen, für die auf mindestens einer Ausgabedatenbankpartition ein Fehler in dieser Phase aufgetreten ist, können Sie den Befehl **LOAD REPLACE** oder **LOAD INSERT** eingeben. Verwenden Sie die Option OUTPUT_DBPARTNUMS, um nur die Datenbankpartitionen anzugeben, auf denen ein Fehler aufgetreten ist.

Bei Ladeoperationen, für die auf mindestens einer Ausgabedatenbankpartition ein Fehler in der LOAD-Phase aufgetreten ist, müssen Sie den Befehl **LOAD RESTART** oder **LOAD TERMINATE** eingeben.

Bei Ladeoperationen, für die auf mindestens einer Ausgabedatenbankpartition in der SETUP-Phase und in der LOAD-Phase ein Fehler aufgetreten ist, müssen Sie zwei Ladeoperationen ausführen, um die fehlgeschlagene Ladeoperation fortzusetzen. Hierbei muss eine für die Fehler während der SETUP-Phase und eine für die Fehler während der LOAD-Phase verwendet werden. Um eine auf diese Weise fehlgeschlagene Ladeoperation rückgängig zu machen, müssen Sie den Befehl **LOAD TERMINATE** eingeben. Nach der Eingabe des Befehls müssen Sie allerdings alle Partitionen berücksichtigen, da in der Tabelle für keine der Partitionen, auf denen in der SETUP-Phase ein Fehler aufgetreten ist, Änderungen vorgenommen wurden, und weil alle Änderungen auf Partitionen rückgängig gemacht wurden, die wäh-

rend der LOAD-Phase fehlgeschlagen sind.

Beispiel: Die Tabelle TABLE1 ist auf den Datenbankpartitionen 0 bis 3 in der Datenbank WSDB definiert. Der folgende Befehl wird abgesetzt:

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

Während der SETUP-Phase tritt auf der Ausgabedatenbankpartition 1 ein Fehler auf. Da Fehler während der SETUP-Phase isoliert werden, wird die Ladeoperation fortgesetzt, während der LOAD-Phase tritt jedoch auf Partition 3 ebenfalls ein Fehler auf. Um die Ladeoperation fortzusetzen, sollten Sie die folgenden Befehle eingeben:

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

Anmerkung: Bei LOAD RESTART-Operationen gelten die im Befehl **LOAD RESTART** angegebenen Optionen. Es ist daher wichtig, in diesem Befehl dieselben Optionen wie im ursprünglichen Befehl **LOAD** anzugeben.

LOAD-Konfigurationsoptionen für Umgebungen mit partitionierten Datenbanken

Es gibt eine Reihe von Konfigurationsoptionen, mit deren Hilfe eine Ladeoperation in einer Umgebung mit partitionierten Datenbanken geändert werden kann.

MODE *X*

Gibt den Modus an, in dem die Ladeoperation beim Laden einer Mehrpartitionsdatenbank stattfindet. PARTITION_AND_LOAD ist der Standardwert. Gültige Werte:

- PARTITION_AND_LOAD. Die Daten werden verteilt (möglicherweise in Parallelverarbeitung) und gleichzeitig auf die entsprechenden Datenbankpartitionen geladen.
- PARTITION_ONLY. Die Daten werden verteilt (möglicherweise in Parallelverarbeitung), und die Ausgabe wird in Dateien an einer angegebenen Speicherposition auf jeder Ladedatenbankpartition geschrieben. Bei anderen Dateitypen als CURSOR lautet das Format des Namens der Ausgabedatei in der betreffenden Datenbankpartition *dateiname.xxx*, wobei *dateiname* der im Befehl **LOAD** angegebene Name der Eingabedatei und *xxx* die dreistellige Datenbankpartitionsnummer ist. Beim Dateityp CURSOR wird der Name der Ausgabedatei auf jeder Datenbankpartition durch die Option PART_FILE_LOCATION festgelegt. Der Abschnitt zur Option PART_FILE_LOCATION enthält weitere Informationen dazu, wie die Position der Verteilungsdatei für die betreffende Datenbankpartition angegeben wird.

  Anmerkung:

  1. Dieser Modus kann nicht für CLI-Ladeoperationen verwendet werden.
  2. Enthält die Tabelle eine Identitätsspalte, die für die Verteilung erforderlich ist, wird dieser Modus nicht unterstützt, es sei denn, der Änderungswert **identityoverride** für den Dateityp wird angegeben.
  3. Verteilungsdateien, die für den Dateityp CURSOR generiert werden, sind nicht mit verschiedenen DB2-Releases kompatibel. Dies bedeutet, dass Verteilungsdateien vom Dateityp CURSOR, die in einem Vorgängerrelease generiert wurden, nicht mit dem Modus LOAD_ONLY geladen werden kön-

nen. Ebenso gilt, dass Verteilungsdateien vom Dateityp CURSOR, die im aktuellen Release generiert wurden, in einem zukünftigen Release nicht mit dem Modus LOAD_ONLY geladen werden können.

- LOAD_ONLY. Es wird davon ausgegangen, dass die Daten bereits verteilt sind. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen. Bei anderen Dateitypen als CURSOR muss das Format des Namens der Eingabedatei für die betreffende Datenbankpartition *dateiname.xxx* lauten, wobei *dateiname* der im Befehl **LOAD** angegebene Name der Datei und *xxx* die dreistellige Datenbankpartitionsnummer ist. Beim Dateityp CURSOR wird der Name der Eingabedatei auf jeder Datenbankpartition durch die Option PART_FILE_LOCATION festgelegt. Der Abschnitt zur Option PART_FILE_LOCATION enthält weitere Informationen dazu, wie die Position der Verteilungsdatei für die betreffende Datenbankpartition angegeben wird.

  Anmerkung:

  1. Dieser Modus kann nicht für CLI-Ladeoperationen verwendet werden und nicht, wenn der Parameter **CLIENT** des Befehls **LOAD** angegeben ist.
  2. Enthält die Tabelle eine Identitätsspalte, die für die Verteilung erforderlich ist, wird dieser Modus nicht unterstützt, es sei denn, der Änderungswert **identityoverride** für den Dateityp wird angegeben.

- LOAD_ONLY_VERIFY_PART. Es wird davon ausgegangen, dass die Daten bereits verteilt sind, aber die Datendatei enthält keine Partitionskopfdaten. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen. Während der Ladeoperation wird für jede Zeile geprüft, ob sie sich auf der korrekten Datenbankpartition befindet. Zeilen, die Datenbankpartitionsverstöße enthalten, werden in eine Speicherauszugsdatei gestellt, sofern der Änderungswert **dumpfile** für den Dateityp angegeben wurde. Andernfalls werden die Zeilen gelöscht. Wenn für eine bestimmte Ladedatenbankpartition Datenbankpartitionsverstöße vorliegen, wird für die betreffende Datenbankpartition eine einzige Warnung in die LOAD-Nachrichtendatei geschrieben. Das Format des Namens der Eingabedatei für die betreffende Datenbankpartition muss *dateiname.xxx* lauten, wobei *dateiname* der im Befehl **LOAD** angegebene Name der Datei und *xxx* die dreistellige Datenbankpartitionsnummer ist. Der Abschnitt zur Option PART_FILE_LOCATION enthält weitere Informationen dazu, wie die Position der Verteilungsdatei für die betreffende Datenbankpartition angegeben wird.

  Anmerkung:

  1. Dieser Modus kann nicht für CLI-Ladeoperationen verwendet werden und nicht, wenn der Parameter **CLIENT** des Befehls **LOAD** angegeben ist.
  2. Enthält die Tabelle eine Identitätsspalte, die für die Verteilung erforderlich ist, wird dieser Modus nicht unterstützt, es sei denn, der Änderungswert **identityoverride** für den Dateityp wird angegeben.

- ANALYZE. Es wird eine optimale Verteilungszuordnung mit einer gleichmäßigen Verteilung auf alle Datenbankpartitionen generiert.

PART_FILE_LOCATION *X*

In den Modi PARTITION_ONLY, LOAD_ONLY und LOAD_ONLY_VERIFY_PART kann mit diesem Parameter die Position der verteilten Dateien angegeben werden. Diese Position muss auf jeder Datenbankpartition vorhanden sein, die mit der Option OUTPUT_DBPARTNUMS angegeben wurde. Handelt es sich bei der angegebenen Position um den Namen eines relativen Pfads, wird der Pfad an das aktuelle Verzeichnis angehängt, um die Position für die verteilten Dateien zu erstellen.

Beim Dateityp CURSOR muss diese Option angegeben werden, und die Position muss sich auf einen vollständig qualifizierten Dateinamen beziehen. Dieser Name ist der vollständig qualifizierte Basisdateiname der verteilten Dateien, die auf jeder Ausgabedatenbankpartition erstellt werden (beim Modus PARTITION_ONLY), bzw. die Position der Dateien, aus denen für jede Datenbankpartition gelesen werden soll (beim Modus LOAD_ONLY). Bei Verwendung des Modus PARTITION_ONLY können mehrere Dateien mit dem angegebenen Basisnamen erstellt werden, wenn die Zieltabelle LOB-Spalten enthält.

Bei anderen Dateitypen als CURSOR wird das aktuelle Verzeichnis für die verteilten Dateien verwendet, falls diese Option nicht angegeben ist.

OUTPUT_DBPARTNUMS *X*
X ist eine Liste mit Datenbankpartitionsnummern. Die Datenbankpartitionsnummern geben die Datenbankpartitionen an, auf denen die Ladeoperation ausgeführt werden soll. Die Datenbankpartitionsnummern müssen eine Untermenge der Datenbankpartitionen sein, auf denen die Tabelle definiert ist. Standardmäßig werden alle Datenbankpartitionen ausgewählt. Die Liste muss in runde Klammern gesetzt werden. Die einzelnen Listeneinträge müssen mit einem Komma voneinander abgegrenzt werden. Die Angabe von Bereichen ist zulässig (Beispiel: (0, 2 to 10, 15)).

PARTITIONING_DBPARTNUMS *X*
X ist eine Liste der Datenbankpartitionsnummern, die im Verteilungsprozess verwendet werden. Die Liste muss in runde Klammern gesetzt werden. Die einzelnen Listeneinträge müssen mit einem Komma voneinander abgegrenzt werden. Die Angabe von Bereichen ist zulässig (Beispiel: (0, 2 to 10, 15)). Die für den Verteilungsprozess angegebenen Datenbankpartitionen können sich von den Datenbankpartitionen, die geladen werden, unterscheiden. Wird die Option PARTITIONING_DBPARTNUMS nicht angegeben, ermittelt das Dienstprogramm LOAD, wie viele Datenbankpartitionen erforderlich sind und welche Datenbankpartitionen im Hinblick auf eine optimale Leistung verwendet werden.

Falls der Änderungswert **anyorder** für den Dateityp im Befehl **LOAD** nicht angegeben wird, wird in der LOAD-Sitzung nur ein Partitionierungsagent verwendet. Des Weiteren wird - wenn nur eine Datenbankpartition für die Option OUTPUT_DBPARTNUMS angegeben wird oder die Koordinatorpartition der Ladeoperation kein Element von OUTPUT_DBPARTNUMS ist - die Koordinatorpartition der Ladeoperation im Verteilungsprozess verwendet. Andernfalls wird die erste in OUTPUT_DBPARTNUMS angegebene Datenbankpartition (nicht die Koordinatorpartition) im Verteilungsprozess verwendet.

Ist der Änderungswert **anyorder** für den Dateityp angegeben, wird die Anzahl der im Verteilungsprozess verwendeten Datenbankpartitionen wie folgt ermittelt: (Anzahl der Partitionen in OUTPUT_DBPARTNUMS geteilt durch 4) plus 1.

MAX_NUM_PART_AGENTS *X*
Gibt an, wie viele Partitionierungsagenten in einer LOAD-Sitzung maximal verwendet werden. Der Standardwert ist 25.

ISOLATE_PART_ERRS *X*
Gibt an, wie die Ladeoperation auf Fehler reagiert, die auf einzelnen Datenbankpartitionen auftreten. Der Standardwert ist LOAD_ERRS_ONLY, sofern nicht sowohl der Parameter **ALLOW READ ACCESS** als auch der Parameter **COPY YES** des Befehls **LOAD** angegeben sind. In diesem Fall ist der Standardwert NO_ISOLATION. Gültige Werte:

- SETUP_ERRS_ONLY. Bei Fehlern, die während der SETUP-Phase auf einer Datenbankpartition auftreten (beispielsweise Fehler beim Zugriff auf eine Da-

tenbankpartition oder Probleme beim Zugriff auf einen Tabellenbereich bzw. auf eine Tabelle auf einer Datenbankpartition), wird die Ladeoperation auf der Datenbankpartition mit dem Fehler gestoppt, auf den übrigen Datenbankpartitionen jedoch fortgesetzt. Fehler, die während des Ladens von Daten auf einer Datenbankpartition auftreten, bewirken das Fehlschlagen der gesamten Operation.

- LOAD_ERRS_ONLY. Fehler, die während der SETUP-Phase auf einer Datenbankpartition auftreten, bewirken das Fehlschlagen der gesamten Ladeoperation. Wenn beim Laden der Daten ein Fehler auftritt, stoppt die Ladeoperation für die Datenbankpartition, auf der der Fehler aufgetreten ist. Die Ladeoperation wird für die verbleibenden Datenbankpartitionen fortgesetzt, bis ein Fehler auftritt oder bis alle Daten geladen wurden. Die neu geladenen Daten sind auf allen Datenbankpartitionen so lange nicht sichtbar, bis eine LOAD RESTART-Operation ausgeführt und erfolgreich beendet wird.

  Anmerkung: Dieser Modus kann nicht verwendet werden, wenn sowohl der Parameter **ALLOW READ ACCESS** als auch der Parameter **COPY YES** des Befehls **LOAD** angegeben sind.
- SETUP_AND_LOAD_ERRS. In diesem Modus führen Fehler auf Datenbankpartitionsebene während der SETUP-Phase oder beim Laden von Daten dazu, dass die Verarbeitung nur auf den betroffenen Datenbankpartitionen gestoppt wird. Wie im Modus LOAD_ERRS_ONLY sind die neu geladenen Daten - falls beim Laden der Daten Partitionsfehler auftreten - auf allen Datenbankpartitionen so lange nicht sichtbar, bis eine LOAD RESTART-Operation ausgeführt und erfolgreich beendet wird.

  Anmerkung: Dieser Modus kann nicht verwendet werden, wenn sowohl die Option **ALLOW READ ACCESS** als auch die Option **COPY YES** des Befehls **LOAD** angegeben sind.
- NO_ISOLATION. In diesem Modus bewirkt jeder Fehler während der Ladeoperation, dass die Ladeoperation insgesamt fehlschlägt.

STATUS_INTERVAL *X*
: *X* gibt an, wie häufig der Benutzer eine Benachrichtigung über den Umfang der gelesenen Daten empfängt. Die Maßeinheit ist Megabyte (MB). Der Standardwert ist 100 MB. Gültige Werte sind ganze Zahlen von 1 bis 4000.

PORT_RANGE *X*
: X gibt den Bereich von TCP-Ports an, die zur Erstellung von Sockets für die interne Kommunikation verwendet werden. Der Standardbereich liegt zwischen 49152 und 65535. Wenn die Registrierdatenbankvariable **DB2ATLD_PORTS** zum Zeitpunkt des Aufrufs definiert ist, überschreibt ihr Wert den Wert, der für die LOAD-Konfigurationsoption PORT_RANGE angegeben ist. Der Bereich für die Registrierdatenbank-Variable **DB2ATLD_PORTS** sollte in dem folgenden Format angegeben werden:

  <niedrige_portnummer:hohe_portnummer>

  Das Format für den Befehlszeilenprozessor lautet:

  (niedrige_portnummer, hohe_portnummer)

CHECK_TRUNCATION
: Gibt an, dass das Programm bei der Ein-/Ausgabe prüfen soll, ob Datensätze abgeschnitten werden. In der Standardeinstellung werden die Daten bei der Ein-/Ausgabe nicht auf ein Abschneiden hin überprüft.

MAP_FILE_INPUT *X*
: *X* gibt den Namen der Eingabedatei für die Verteilungszuordnung an. Dieser

Parameter muss angegeben werden, wenn die Verteilungszuordnung angepasst wurde, da er auf die Datei verweist, in der sich die angepasste Verteilungszuordnung befindet. Eine angepasste Verteilungszuordnung kann erstellt werden, indem die Zuordnung mit dem Programm **db2gpmap** aus der Tabelle mit dem Datenbanksystemkatalog extrahiert wird oder indem mit dem Modus ANALYZE des Befehls **LOAD** eine optimale Zuordnung generiert wird. Die mit dem Modus ANALYZE generierte Zuordnung muss auf alle Datenbankpartitionen in der Datenbank versetzt werden, bevor die Ladeoperation fortgesetzt werden kann.

MAP_FILE_OUTPUT *X*
X ist der Name der Ausgabedatei für die Verteilungszuordnung. Die Ausgabedatei wird auf der Datenbankpartition erstellt, die den Befehl **LOAD** absetzt, wobei davon ausgegangen wird, dass diese Datenbankpartition zu der Datenbankpartitionsgruppe gehört, in der die Partitionierung stattfindet. Wird der Befehl **LOAD** auf einer Datenbankpartition aufgerufen, die nicht an der Partitionierung beteiligt ist (wie von **PARTITIONING_DBPARTNUMS** definiert), wird die Ausgabedatei auf der ersten Datenbankpartition erstellt, die mit dem Parameter **PARTITIONING_DBPARTNUMS** definiert ist. Betrachten Sie das folgende Setup einer Umgebung mit partitionierten Datenbanken:

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

Bei Ausführung des folgenden **LOAD**-Befehls auf Server 3 (serv3) wird die Verteilungszuordnung auf Server 1 (serv1) erstellt.

```
LOAD FROM datei OF ASC METHOD L ( ...) INSERT INTO tabelle CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

Dieser Parameter sollte verwendet werden, wenn der Modus ANALYZE angegeben wird. Es wird eine optimale Verteilungszuordnung mit einer gleichmäßigen Verteilung auf alle Datenbankpartitionen generiert. Falls dieser Parameter nicht angegeben und der Modus ANALYZE angegeben ist, wird das Programm mit einem Fehler beendet.

TRACE *X*
Gibt die Anzahl der Datensätze an, für die ein Trace erstellt werden soll, wenn ein Speicherauszug des Datenkonvertierungsprozesses überprüft werden soll und eine Ausgabe der Hash-Werte erforderlich ist. Der Standardwert ist 0.

NEWLINE
Diese Option wird verwendet, wenn es sich bei der Eingabedatendatei um eine ASC-Datei handelt, in der jeder Datensatz durch ein Zeilenvorschubzeichen begrenzt wird, und der Änderungswert **reclen** für den Dateityp im Befehl **LOAD** angegeben wird. Bei Angabe dieser Option wird jeder Datensatz daraufhin geprüft, ob er ein Zeilenvorschubzeichen enthält. Die Satzlänge, wie im Änderungswert **reclen** für den Dateityp angegeben, wird außerdem geprüft.

DISTFILE *X*
Wird diese Option angegeben, generiert das Dienstprogramm LOAD eine Datenbankpartitionsverteilungsdatei mit dem angegebenen Namen. Die Datenbankpartitionsverteilungsdatei enthält 32.768 Integer: ein Integer für jeden Eintrag in der Verteilungszuordnung der Zieltabelle. Jedes Integer in der Datei stellt die Anzahl der Zeilen in den geladenen Eingabedateien dar, für die zum entsprechenden Eintrag der Verteilungszuordnung ein Hashverfahren ausgeführt wird. Diese Informationen können Ihnen nicht nur dabei helfen, ungleiche Verteilungen in Ihren Daten zu ermitteln, sondern auch zu entscheiden, ob

mithilfe des Modus ANALYZE des Dienstprogramms eine neue Verteilungszuordnung für die Tabelle generiert werden soll. Wenn diese Option nicht angegeben wird, ist es das Standardverhalten des Dienstprogramms LOAD, die Verteilungsdatei nicht zu generieren.

Anmerkung: Bei Angabe dieser Option wird maximal ein Partitionierungsagent für die Ladeoperation verwendet. Selbst wenn Sie explizit mehrere Partitionierungsagenten anfordern, wird nur einer verwendet.

OMIT_HEADER
Gibt an, dass die Verteilungsdatei keine Kopfdaten für die Verteilungszuordnung enthalten soll. Wenn diese Option nicht angegeben wird, werden Kopfdaten generiert.

RUN_STAT_DBPARTNUM *X*
Falls der Parameter **STATISTICS USE PROFILE** im Befehl **LOAD** angegeben ist, werden Statistikdaten nur auf einer Datenbankpartition erfasst. Dieser Parameter gibt an, auf welcher Datenbankpartition Statistikdaten erfasst werden sollen. Wenn dieser Wert -1 lautet oder gar nicht angegeben wird, werden Statistikdaten auf der ersten Datenbankpartition in der Liste der Ausgabedatenbankpartitionen erfasst.

LOAD-Sitzungen in einer Umgebung mit partitionierten Datenbanken - CLP-Beispiele

In den folgenden Beispielen wird das Laden von Daten in eine Mehrpartitionsdatenbank dargestellt.

Die Datenbank hat vier Datenbankpartitionen, die von 0 bis 3 nummeriert sind. Die Datenbank WSDB ist auf allen Datenbankpartitionen definiert. Die Tabelle TABLE1 befindet sich in der Standarddatenbankpartitionsgruppe, die ebenfalls auf allen Datenbankpartitionen definiert ist.

Beispiel 1

Um Daten aus der Benutzerdatendatei load.del, die sich auf Datenbankpartition 0 befindet, in die Tabelle TABLE1 zu laden, stellen Sie eine Verbindung zur Datenbankpartition 0 her, und setzen Sie anschließend den folgenden Befehl ab:

```
load from load.del of del replace into table1
```

Wenn die Ladeoperation erfolgreich durchgeführt wird, erhalten Sie die folgende Ausgabe:

```
Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________
  LOAD           000      +00000000    Erfolg.
_____________________________________________________
  LOAD           001      +00000000    Erfolg.
_____________________________________________________
  LOAD           002      +00000000    Erfolg.
_____________________________________________________
  LOAD           003      +00000000    Erfolg.
_____________________________________________________
  PARTITION      001      +00000000    Erfolg.
_____________________________________________________
  PRE_PARTITION  000      +00000000    Erfolg.
_____________________________________________________
  ERGEBNISSE:    4 von 4 LOAD-Operationen wurden
                   erfolgreich beendet.
_____________________________________________________
```

```
Zusammenfassung für Partitionierungsagenten
Gelesene Zeilen                 = 100000
Zurückgewiesene Zeilen          = 0
Partitionierte Zeilen           = 100000

Zusammenfassung für LOAD-Agenten:
Anzahl gelesener Zeilen         = 100000
Anzahl übersprungener Zeilen    = 0
Anzahl geladener Zeilen         = 100000
Anzahl zurückgewiesener Zeilen  = 0
Anzahl gelöschter Zeilen        = 0
Anzahl festgeschriebener Zeilen = 100000
```

Die Ausgabe gibt an, dass sich auf jeder Datenbankpartition ein Ladeagent befand, der jeweils erfolgreich ausgeführt wurde. Außerdem zeigt die Ausgabe, dass auf der Koordinatorpartition ein Agent für Partitionierungsvorbereitung ausgeführt wurde sowie ein Partitionierungsagent auf Datenbankpartition 1. Diese Prozesse wurden erfolgreich mit einem normalen SQL-Rückkehrcode 0 abgeschlossen. Die Zusammenfassung der Statistikdaten ergibt, dass der Agent für Partitionierungsvorbereitung 100.000 Zeilen gelesen hat, dass der Partitionierungsagent 100.000 Zeilen verteilt hat und dass durch die Ladeagenten insgesamt 100.000 Zeilen geladen wurden.

Beispiel 2

Im folgenden Beispiel werden die Daten in die Tabelle TABLE1 geladen, wobei der Modus PARTITION_ONLY verwendet wird. Die verteilten Ausgabedateien werden auf allen Ausgabedatenbankpartitionen jeweils im Verzeichnis /db/data gespeichert:

```
load from load.del of del replace into table1 partitioned db config mode
partition_only part_file_location /db/data
```

Der Befehl LOAD hat die folgende Ausgabe:

```
Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________
LOAD_TO_FILE   000      +00000000    Erfolg.
_____________________________________________________
LOAD_TO_FILE   001      +00000000    Erfolg.
_____________________________________________________
LOAD_TO_FILE   002      +00000000    Erfolg.
_____________________________________________________
LOAD_TO_FILE   003      +00000000    Erfolg.
_____________________________________________________
PARTITION      001      +00000000    Erfolg.
_____________________________________________________
PRE_PARTITION  000      +00000000    Erfolg.
_____________________________________________________

   Zusammenfassung für Partitionierungsagenten
   Gelesene Zeilen                 = 100000
   Zurückgewiesene Zeilen          = 0
   Partitionierte Zeilen           = 100000
```

Die Ausgabe macht deutlich, dass auf jeder Ausgabedatenbankpartition jeweils ein Dateiladeagent aktiv war und dass diese Agenten erfolgreich ausgeführt wurden. Auf der Koordinatorpartition wurde ein Agent für Partitionierungsvorbereitung ausgeführt, und auf Datenbankpartition 1 wurde ein Partitionierungsagent ausgeführt. Die Zusammenfassung der Statistikdaten ergibt, dass der Agent für Partitionierungsvorbereitung 100.000 Zeilen erfolgreich gelesen hat und dass 100.000 Zeilen vom Partitionierungsagenten erfolgreich verteilt wurden. Da keine Zeilen in die Tabelle geladen wurden, wird keine Zusammenfassung über die Anzahl der geladenen Zeilen angezeigt.

Beispiel 3

Mit dem folgenden Befehl können Sie die Dateien laden, die mit der zuvor dargestellten Ladeoperation im Modus PARTITION_ONLY generiert wurden:

```
load from load.del of del replace into table1 partitioned db config mode
 load_only part_file_location /db/data
```

Der Befehl LOAD hat die folgende Ausgabe:

```
Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________
   LOAD           000      +00000000    Erfolg.
_____________________________________________________
   LOAD           001      +00000000    Erfolg.
_____________________________________________________
   LOAD           002      +00000000    Erfolg.
_____________________________________________________
   LOAD           003      +00000000    Erfolg.
_____________________________________________________
   ERGEBNISSE:    4 von 4 LOAD-Operationen wurden
                  erfolgreich beendet.
_____________________________________________________

   Zusammenfassung für LOAD-Agenten:
   Anzahl gelesener Zeilen         = 100000
   Anzahl übersprungener Zeilen    = 0
   Anzahl geladener Zeilen         = 100000
   Anzahl zurückgewiesener Zeilen  = 0
   Anzahl gelöschter Zeilen        = 0
   Anzahl festgeschriebener Zeilen = 100000
```

Die Ausgabe besagt, dass die Ladeagenten auf allen Ausgabedatenbankpartitionen jeweils erfolgreich ausgeführt wurden und dass insgesamt 100.000 Zeilen durch alle Ladeagenten geladen wurden. Die Zusammenfassung enthält keine Angaben zu verteilten Zeilen, da keine Verteilung erfolgte.

Beispiel 4

Beim Absetzen des LOAD-Befehls

```
load from load.del of del replace into table1
```

wird eventuell die folgende Ausgabe zurückgegeben, wenn während der Ladeoperation auf einer der Ladedatenbankpartitionen nicht genügend Speicherbereich im Tabellenbereich verfügbar ist:

```
   SQL0289N  Dem Tabellenbereich "DMS4KT" konnten keine neuen Seiten
zugeordnet werden.
   SQLSTATE=57011

Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________
   LOAD           000      +00000000    Erfolg.
_____________________________________________________
   LOAD           001      -00000289    Fehler. Möglicherweise
Neustart (RESTART) erforderlich.
_____________________________________________________
   LOAD           002      +00000000    Erfolg.
_____________________________________________________
   LOAD           003      +00000000    Erfolg.
_____________________________________________________
   PARTITION      001      +00000000    Erfolg.
_____________________________________________________
   PRE_PARTITION  000      +00000000    Erfolg.
_____________________________________________________
```

```
ERGEBNISSE:    3 von 4 LOAD-Operationen wurden
erfolgreich beendet.
```

```
Zusammenfassung für Partitionierungsagenten
Gelesene Zeilen                 = 0
Zurückgewiesene Zeilen          = 0
Partitionierte Zeilen           = 0

Zusammenfassung für LOAD-Agenten:
Anzahl gelesener Zeilen         = 0
Anzahl übersprungener Zeilen    = 0
Anzahl geladener Zeilen         = 0
Anzahl zurückgewiesener Zeilen  = 0
Anzahl gelöschter Zeilen        = 0
Anzahl festgeschriebener Zeilen = 0
```

Die Ausgabe gibt an, dass die Ladeoperation den Fehler SQL0289 zurückgegeben hat. Die Zusammenfassung für die Datenbankpartitionen besagt, dass auf Datenbankpartition 1 nicht genügend Speicherbereich vorhanden war. Wird zusätzlicher Speicherbereich zu den Containern des Tabellenbereichs auf Datenbankpartition 1 hinzufügt, können Sie die Ladeoperation wie folgt erneut starten:

```
load from load.del of del restart into table1
```

Migration und Versionskompatibilität

Die Registrierdatenbankvariable **DB2_PARTITIONEDLOAD_DEFAULT** kann verwendet werden, um in einer Datenbank mit mehreren Partitionen zum Ladeverhalten von DB2 Universal Database vor Version 8 zurückzukehren.

Anmerkung: Die Registrierdatenbankvariable **DB2_PARTITIONEDLOAD_DEFAULT** ist veraltet und wird in einem späteren Release möglicherweise entfernt.

Durch die Rückkehr zu dem Verhalten des Befehls **LOAD** von DB2 UDB vor Version 8 in einer Datenbank mit mehreren Partitionen können Sie eine Datei mit gültigen Verteilungskopfdaten in eine einzige Datenbankpartition laden, ohne hierbei zusätzliche Konfigurationsoptionen für partitionierte Datenbanken anzugeben. Hierzu können Sie den Wert von **DB2_PARTITIONEDLOAD_DEFAULT** auf NO setzen. Die Verwendung dieser Option ist beispielsweise sinnvoll, wenn Sie Änderungen an vorhandenen Scripts vermeiden möchten, die den Befehl **LOAD** für einzelne Datenbankpartitionen absetzen. Um beispielsweise eine Verteilungsdatei in die Datenbankpartition 3 einer Tabelle zu laden, die zu einer Datenbankpartitionsgruppe mit vier Datenbankpartitionen gehört, setzen Sie den folgenden Befehl ab:

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

Anschließend setzen Sie die folgenden Befehle über den DB2-Befehlszeilenprozessor (CLP) ab:

```
CONNECT RESET

SET CLIENT CONNECT_NODE 3

CONNECT TO DB MYDB

LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

In einer Mehrpartitionsdatenbank findet die Ladeoperation für alle Datenbankpartitionen, in denen die Tabelle definiert ist, statt, wenn keine LOAD-Konfigurationsoptionen für das Laden in Mehrpartitionsdatenbanken angegeben werden. Die Eingabedatei muss keine Verteilungskopfdaten enthalten, und die Option MODE nimmt

standardmäßig den Wert PARTITION_AND_LOAD an. Um das Laden für eine einzelne Datenbankpartition auszuführen, muss die Option OUTPUT_DBPARTNUMS angegeben werden.

Migration von Umgebungen mit partitionierten Datenbanken

Migrieren von partitionierten Datenbanken

Die Migration von Umgebungen mit partitionierten Datenbanken setzt voraus, dass Sie das neueste Relase des Datenbankprodukts auf allen Datenbankpartitionsservern installieren. Anschließend migrieren Sie die Instanzen und dann die Datenbanken.

Datenbankpartitionsserver können vom Katalogdatenbankpartitionsserver oder von einem anderen Datenbankpartitionsserver aus migriert werden. Falls der Migrationsprozess fehlschlägt, können Sie die Migration vom Katalogdatenbankpartitionsserver oder von einem anderen Datenbankpartitionsserver aus erneut versuchen.

Da eine Migration dieser Art eine Operation von erheblicher Tragweite ist, würde eine Beschreibung der Migrationsprozedur sowie der Voraussetzungen und Einschränkungen über den Rahmen dieses Handbuchs hinausgehen. Eine detaillierte Beschreibung finden Sie im Thema „Migrieren von Umgebungen mit partitionierten Datenbanken" im Handbuch *Migration*. Dort finden Sie auch Querverweise auf zahlreiche weitere Themen, mit denen Sie sich vertraut machen sollten, bevor Sie eine Migration durchführen.

Verwenden von Momentaufnahme- und Ereignismonitoren

Überwachen der Reorganisation einer partitionierten Tabelle mit Snapshot Monitor-Daten

Im Folgenden werden einige der nützlichsten Methoden zur Überwachung des globalen Status einer Tabellenreorganisation erläutert.

Informationen zu diesem Vorgang

Es gibt keine eigene Datengruppe, die den allgemeinen Status der Reorganisation einer partitionierten Tabelle angibt. 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. Der globale Status einer Tabellenreorganisation lässt sich jedoch aus den Werten der Elemente in den einzelnen reorganisierten Datengruppen der Datenpartitionen ableiten. Im Folgenden werden einige der nützlichsten Methoden zur Überwachung des globalen Status einer Tabellenreorganisation erläutert.

Ermittlung der Anzahl der reorganisierten Datenpartitionen

Die Gesamtzahl der für eine Tabelle reorganisierten Datenpartitionen lässt sich ermitteln, indem die Anzahl der Überwachungsdatenblöcke für Tabellendaten gezählt werden, die denselben Tabellen- und Schemanamen haben. Dieser Wert gibt die Anzahl der Datenpartitionen an, auf denen die Reorganisation gestartet wurde. Die Beispiele 1 und 2 zeigen, dass drei Partitionen reorganisiert werden.

Ermitteln der reorganisierten Datenpartition

Die Datenpartition, die zum jeweiligen Zeitpunkt gerade reorganisiert wird, lässt sich aus der Startzeit der Reorganisationsphase (reorg_phase_start) ableiten. Während der Phase SORT/BUILD/REPLACE geben die Überwachungsdaten der gerade reorganisierten Datenpartition die neueste Startzeit der betreffenden Phase an. Während der Phase INDEX_RECREATE (Index erneut erstellen) ist die Startzeit der Phase für alle Datenpartitionen gleich. In den Beispielen 1 und 2 wird die Phase INDEX_RECREATE angegeben, sodass die Startzeit für alle Datenpartitionen gleich ist.

Ermitteln des Bedarfs für einen Index-Rebuild

Sie können ermitteln, ob ein Index erneut erstellt werden muss, indem Sie den Wert des Elements für die maximale Reorganisationsphase (reorg_max_phase) abrufen, das einer der reorganisierten Datenpartitionen entspricht. Hat das Element 'reorg_max_phase' einen Wert von 3 oder 4, ist ein Index-Rebuild erforderlich. In den Beispielen 1 und 2 hat 'reorg_max_phase' den Wert 3; es ist also ein Index-Rebuild erforderlich.

Beispiele

Die folgende Beispielausgabe stammt von einem Server mit drei Knoten, der eine Tabelle mit drei Datenpartitionen enthält:

```
CREATE TABLE  sales (c1 INT, c2 INT, c3 INT)
    PARTITION BY RANGE (c1)
        (PART P1 STARTING FROM (1)  ENDING AT (10) IN parttbs,
         PART P2 STARTING FROM (11) ENDING AT (20) IN parttbs,
         PART P3 STARTING FROM (21) ENDING AT (30) IN parttbs)
    DISTRIBUTE BY (c2)
```

Ausgeführte Anweisung:

```
REORG TABLE sales ALLOW NO ACCESS ON ALL DBPARTITIONNUMS
```

Beispiel 1:

```
GET SNAPSHOT FOR TABLES ON DPARTDB GLOBAL
```

Die Ausgabe wurde geändert und enthält lediglich die Informationen für die relevante Tabelle.

```
                        Momentaufnahme einer Tabelle

Zeitmarke für erste Datenbankverbindung = 06/28/2005 13:46:43.061690
Zeitmarke für letzte Zurücksetzung      = 06/28/2005 13:46:47.440046
Zeitmarke für Momentaufnahme            = 06/28/2005 13:46:50.964033
Datenbankname                               = DPARTDB
Datenbankpfad                               = /work/sales/NODE0000/SQL00001/
Aliasname der Eingabedatenbank              = DPARTDB
Anzahl Tabellen im Zugriff              = 5

Tabellenverzeichnis
  Tabellenschema         = NEWTON
  Tabellenname           = SALES
  Tabellenart            = User
  Datenpartitions-ID     = 0
  Datenobjektseiten      = 3
  Gelesene Zeilen        = 12
  Geschriebene Zeilen    = 1
  Überläufe              = 0
  Seitenreorganisationen = 0
  Informationen zur Tabellenreorganisation:
      Knotennummer                          = 0
      Reorganisationstyp                    =
                Wiederherstellen
```

```
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:49.816883
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.362918
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.821244

Informationen zur Tabellenreorganisation:
    Knotennummer                          = 1
    Reorganisationstyp                    =
                Wiederherstellen
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:49.822701
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.420741
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.899543


Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
                Wiederherstellen
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:49.814813
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619


Tabellenschema         = NEWTON
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 1
Datenobjektseiten      = 3
Gelesene Zeilen        = 8
```

```
Geschriebene Zeilen    = 1
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                         = 0
    Reorganisationstyp                   =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                 = 0
    Tabellenbereich für Reorg.           = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                            = 06/28/2005 13:46:50.014617
    Reorganisationsphase                 = 3 - Index erneut erstellen
    Max. Phase                           = 3
    Phasenstartzeit                      = 06/28/2005 13:46:50.362918
    Status                               = Beendet
    Aktueller Zähler                     = 0
    Max. Zähler                          = 0
    Beendigungsstatus                    = 0
    Endzeit                              = 06/28/2005 13:46:50.821244

Informationen zur Tabellenreorganisation:
    Knotennummer                         = 1
    Reorganisationstyp                   =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                 = 0
    Tabellenbereich für Reorg.           = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                            = 06/28/2005 13:46:50.026278
    Reorganisationsphase                 = 3 - Index erneut erstellen
    Max. Phase                           = 3
    Phasenstartzeit                      = 06/28/2005 13:46:50.420741
    Status                               = Beendet
    Aktueller Zähler                     = 0
    Max. Zähler                          = 0
    Beendigungsstatus                    = 0
    Endzeit                              = 06/28/2005 13:46:50.899543

Informationen zur Tabellenreorganisation:
    Knotennummer                         = 2
    Reorganisationstyp                   =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                 = 0
    Tabellenbereich für Reorg.           = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                            = 06/28/2005 13:46:50.006392
    Reorganisationsphase                 = 3 - Index erneut erstellen
    Max. Phase                           = 3
    Phasenstartzeit                      = 06/28/2005 13:46:50.344277
    Status                               = Beendet
    Aktueller Zähler                     = 0
    Max. Zähler                          = 0
    Beendigungsstatus                    = 0
    Endzeit                              = 06/28/2005 13:46:50.803619
```

```
Tabellenschema         = NEWTON
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 2
Datenobjektseiten      = 3
Gelesene Zeilen        = 4
Geschriebene Zeilen    = 1
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                          = 0
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.199971
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.362918
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.821244

Informationen zur Tabellenreorganisation:
    Knotennummer                          = 1
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.223742
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.420741
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.899543

Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.179922
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
```

```
     Aktueller Zähler                    = 0
     Max. Zähler                         = 0
     Beendigungsstatus                   = 0
     Endzeit                             = 06/28/2005 13:46:50.803619
```

Beispiel 2:

```
GET SNAPSHOT FOR TABLES ON DPARTDB AT DBPARTITIONNUM 2
```

Die Ausgabe wurde geändert und enthält lediglich die Informationen für die relevante Tabelle.

```
                        Momentaufnahme einer Tabelle

Zeitmarke für erste Datenbankverbindung     = 06/28/2005 13:46:43.617833
Zeitmarke für letzte Zurücksetzung          =
Zeitmarke für Momentaufnahme                = 06/28/2005 13:46:51.016787
Datenbankname                               = DPARTDB
Datenbankpfad                               = /work/sales/NODE0000/SQL00001/
Aliasname der Eingabedatenbank              = DPARTDB
Anzahl Tabellen im Zugriff                  = 3

Tabellenverzeichnis
  Tabellenschema         = NEWTON
  Tabellenname           = SALES
  Tabellenart            = User
  Datenpartitions-ID     = 0
  Datenobjektseiten      = 1
  Gelesene Zeilen        = 0
  Geschriebene Zeilen    = 0
  Überläufe              = 0
  Seitenreorganisationen = 0
  Informationen zur Tabellenreorganisation:
      Knotennummer                          = 2
      Reorganisationstyp                    =
                Wiederherstellen
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
      Reorganisationsindex                  = 0
      Tabellenbereich für Reorg.            = 3
      ID des temporären LOB-Speicherbereichs = 3
      Startzeit                             = 06/28/2005 13:46:49.814813
      Reorganisationsphase                  = 3 - Index erneut erstellen
      Max. Phase                            = 3
      Phasenstartzeit                       = 06/28/2005 13:46:50.344277
      Status                                = Beendet
      Aktueller Zähler                      = 0
      Max. Zähler                           = 0
      Beendigungsstatus                     = 0
      Endzeit                               = 06/28/2005 13:46:50.803619


  Tabellenschema         = NEWTON
  Tabellenname           = SALES
  Tabellenart            = User
  Datenpartitions-ID     = 1
  Datenobjektseiten      = 1
  Gelesene Zeilen        = 0
  Geschriebene Zeilen    = 0
  Überläufe              = 0
  Seitenreorganisationen = 0
  Informationen zur Tabellenreorganisation:
      Knotennummer                          = 2
      Reorganisationstyp                    =
                Wiederherstellen
```

```
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.006392
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619


Tabellenschema         = NEWTON
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 2
Datenobjektseiten      = 1
Gelesene Zeilen        = 4
Geschriebene Zeilen    = 1
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
                Wiederherstellen
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.179922
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619
```

Beispiel 3:

```
SELECT * FROM SYSIBMADM.SNAPLOCK WHERE tabname = 'SALES';
```

Die Ausgabe wurde geändert und enthält lediglich einen Teil der Informationen für die relevante Tabelle.

```
...   TBSP_NAME TABNAME LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_STATUS ...
     --------- ------- ------------------ ---------- -----------
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
```

```
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...

 9 Satz/Sätze ausgewählt.
```

Ausgabe zu der Abfrage (Forts.)

```
... LOCK_ESCALATION LOCK_ATTRIBUTES DATA_PARTITION_ID DBPARTITIONNUM
   --------------- --------------- ----------------- --------------
...              0 INSERT          2                 2
...              0 NONE            -                 2
...              0 NONE            2                 2
...              0 INSERT          0                 0
...              0 NONE            -                 0
...              0 NONE            0                 0
...              0 INSERT          1                 1
...              0 NONE            -                 1
...              0 NONE            1                 1
```

Beispiel 4:

```
SELECT * FROM SYSIBMADM.SNAPTAB WHERE tabname = 'SALES';
```

Die Ausgabe wurde geändert und enthält lediglich einen Teil der Informationen für die relevante Tabelle.

```
... TABSCHEMA TABNAME TAB_FILE_ID TAB_TYPE   DATA_OBJECT_PAGES ROWS_WRITTEN ...
... --------- ------- ----------- ---------- ----------------- ------------ ...
... NEWTON    SALES             2 USER_TABLE                 1            1 ...
... NEWTON    SALES             4 USER_TABLE                 1            1 ...
... NEWTON    SALES             3 USER_TABLE                 1            1 ...

      3 Satz/Sätze ausgewählt.
```

Ausgabe zu der Abfrage (Forts.)

```
... OVERFLOW_ACCESSES PAGE_REORGS DBPARTITIONNUM TBSP_ID DATA_PARTITION_ID
... ----------------- ----------- -------------- ------- -----------------
...                 0           0              0       3                 0
...                 0           0              2       3                 2
...                 0           0              1       3                 1
```

Beispiel 5:

```
SELECT * FROM SYSIBMADM.SNAPTAB_REORG WHERE tabname = 'SALES';;
```

Die Ausgabe wurde geändert und enthält lediglich einen Teil der Informationen für die relevante Tabelle.

```
REORG_PHASE   REORG_MAX_PHASE REORG_TYPE                                     ...
 ------------- --------------- ------------------------------------------
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...

 9 Satz/Sätze ausgewählt.
```

Ausgabe zu der Abfrage (Forts.)

```
... REORG_STATUS REORG_TBSPC_ID DBPARTITIONNUM DATA_PARTITION_ID
    ------------------------- -------------- ----------------
... COMPLETED                3              2                 0
... COMPLETED                3              2                 1
... COMPLETED                3              2                 2
... COMPLETED                3              1                 0
... COMPLETED                3              1                 1
... COMPLETED                3              1                 2
... COMPLETED                3              0                 0
... COMPLETED                3              0                 1
... COMPLETED                3              0                 2
```

Beispiel 6:

Die Informationen zur Tabellenreorganisation enthalten Informationen zur Freigabe von Speicherbereichen als Teil einer Reorganisationsoperation. Das nachstehende Beispiel zeigt die relevante Ausgabe.

```
db2 -v "get snapshot for tables on wsdb"

  Informationen zur Tabellenreorganisation:
      Reorganisationstyp                    =
        Freigabe von Speicherbereichen
        Schreibzugriff zulassen
      Reorganisationsindex                  = 0
   Tabellenbereich für Reorganisation       = 0
   Startzeit                                = 10/22/2008 15:49:35.477532
   Reorganisationsphase                     = 12 - Release
      Max. Phase                            = 3
```

Anmerkung: Bei Anforderungen von Momentaufnahmen über einen Monitor vor Version SQLM_DBMON_VERSION9_7 wird kein Freigabestatus der Reorganisation an den anfordernden Client zurückgegeben.

Globale Momentaufnahmen auf partitionierten Datenbanksystemen

Auf einem partitionierten Datenbanksystem können Sie mithilfe von Snapshot Monitor eine Momentaufnahme der aktuellen Partition, einer bestimmten Partition oder aller Partitionen erstellen. Beim Erstellen einer globalen Momentaufnahme für alle Partitionen einer partitionierten Datenbank werden die Daten zusammengefasst, bevor die Ergebnisse zurückgegeben werden.

Die Daten werden für die verschiedenen Elementtypen wie folgt zusammengefasst:

- **Zähler, Zeit und Wertangaben**

  Die Daten enthalten die Summe aller gleichen Werte, die in den einzelnen Partitionen der Instanz erfasst wurden. Die Anweisung GET SNAPSHOT FOR DATABASE XYZ ON TEST GLOBAL beispielsweise würde die Anzahl der aus der Datenbank gelesenen Zeilen (rows_read) für alle Partitionen in der partitionierten Datenbankinstanz zurückgeben.

- **Grenzwerte**

  Gibt den höchsten Wert (bei oberer Grenze) bzw. den niedrigsten Wert (bei unterer Grenze) an, der in einer beliebigen Partition im partitionierten Datenbanksystem gefunden wurde. Gibt der zurückgegebene Wert Anlass zu Besorgnis, können Momentaufnahmen für die einzelnen Partitionen erstellt werden, um zu ermitteln, ob eine bestimmte Partition überlastet ist oder ob das Problem für die gesamte Instanz relevant ist.

- **Zeitmarke**

Die Daten werden auf den Wert der Zeitmarke für die Partition gesetzt, mit der der Agent der Snapshot Monitor-Instanz verbunden ist. Bitte beachten Sie, dass sämtliche Zeitmarkenwerte vom Monitorschalter für `Zeitmarken` (TIMESTAMP) gesteuert werden.

- **Information**

  Gibt die jeweils höchstwertige Information für eine Partition zurück, die unter Umständen die Arbeit behindert. Beispiel für das Element `appl_status`: Ist der Status in einer Partition 'UOW wird ausgeführt' und in einer anderen 'Wartestatus für Sperre', würde 'Wartestatus für Sperre' zurückgegeben werden, da dies der Status ist, der die Ausführung der Anwendung verzögert.

Sie können auch Zähler zurücksetzen, Monitorschalter einstellen und Monitorschalterstellungen abrufen. Dies ist entweder für einzelnen Partitionen oder alle Partitionen in der partitionierten Datenbank möglich.

Anmerkung: Wird eine globale Momentaufnahme erstellt und tritt dabei in einer oder mehreren der Partitionen ein Fehler auf, werden von den Partitionen, in denen die Momentaufnahme erfolgreich war, die Daten erfasst, und gleichzeitig wird eine Warnung (SQLCODE-Wert 1629) zurückgegeben. Schlägt ein globaler Abruf bzw. eine globale Aktualisierung von Monitorschaltern oder das globale Zurücksetzen von Zählern in einer oder mehreren der Partitionen fehl, werden in den betreffenden Partitionen die Schalter nicht gesetzt bzw. die Daten nicht zurückgesetzt.

Erstellen eines Ereignismonitors für partitionierte Datenbanken oder für Datenbanken in einer DB2 pureScale-Umgebung

Im Allgemeinen ist die Funktionsweise von Ereignismonitoren für partitionierte Datenbanksysteme oder Ereignismonitoren in einer DB2 pureScale-Umgebung ähnlich der von Ereignismonitoren, die in Datenbanken mit einem einzigen Member ausgeführt werden. Es bestehen jedoch einige Unterschiede, die Sie kennen sollten, wenn Sie einen Ereignismonitor für diese Umgebungen erstellen.

Vorgehensweise

- Geben Sie die Partition an, die überwacht werden soll.

  ```
  CREATE EVENT MONITOR tabmon FOR TABLES
                      WRITE TO FILE '/tmp/tabevents'
                      ON PARTITION 3
  ```

  `tabmon` stellt den Namen des Ereignismonitors dar.

  `/tmp/tab events` ist der Name des Verzeichnispfads (unter UNIX), in den der Ereignismonitor die Ereignisdateien schreiben soll.

  `3` ist die Nummer der zu überwachenden Partition.

- Geben Sie an, ob die Ereignismonitordaten auf lokaler oder globaler Ebene (LOCAL bzw. GLOBAL) erfasst werden sollen. Setzen Sie die folgende Anweisung ab, um beispielsweise Ereignismonitorberichte von allen Partitionen zu erfassen:

  ```
  CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                      WRITE TO FILE '/tmp/dlevents'
                                          ON PARTITION 3 GLOBAL
  ```

  Anmerkung: Nur Ereignismonitore für Deadlocks und Deadlocks mit Details können als GLOBAL definiert werden.
  Alle Partitionen melden Ereignisdatensätze im Zusammenhang mit Deadlocks an Partition 3.

- Um Ereignisüberwachungsberichte nur von der lokalen Partition zu erfassen, setzen Sie die folgende Anweisung ab:

```
CREATE EVENT MONITOR dlmon FOR TABLES
                    WRITE TO FILE '/tmp/dlevents'
                                        ON PARTITION 3 LOCAL
```

Dies ist das Standardverhalten für Datei- und Pipe-Ereignismonitore in partitionierten Datenbanken. Die Klauseln LOCAL und GLOBAL werden für Ereignismonitore mit der Klausel WRITE TO TABLE ignoriert.

- Die Werte für die Monitorpartition und den Überwachungsumfang können für vorhandene Ereignismonitore überprüft werden. Hierfür wird die Tabelle SYSCAT.EVENTMONITORS mithilfe der folgenden Anweisung abgefragt:

  ```
  SELECT EVMONNAME, NODENUM, MONSCOPE FROM SYSCAT.EVENTMONITORS
  ```

Ergebnisse

Nachdem ein Ereignismonitor erstellt und aktiviert wurde, erfasst er Überwachungsdaten zu den angegebenen Ereignissen, sobald diese auftreten.

Entwickeln einer geeigneten Backup- und Recovery-Strategie

Recovery nach Systemabsturz

Transaktionen (bzw. UOWs), die für eine Datenbank ausgeführt werden, können auf unerwartete Weise unterbrochen werden. Wenn eine Störung auftritt, bevor alle Änderungen, die Bestandteil der UOW (Unit of Work - Arbeitseinheit) sind, beendet, festgeschrieben und auf Platte geschrieben wurden, verbleibt die Datenbank in einem inkonsistenten und unbrauchbaren Status.

Die *Recovery nach einem Systemabsturz* versetzt die Datenbank wieder in einen konsistenten und verwendbaren Status. Dies geschieht durch Rollback der unvollständigen Transaktionen und Beenden der festgeschriebenen Aktionen, die sich zum Zeitpunkt des Systemabsturzes noch im Hauptspeicher befanden (Abb. 43). Wenn sich eine Datenbank in einem konsistenten und verwendbaren Status befindet, hat sie einen Punkt erreicht, der als *Konsistenzzustand* bezeichnet wird.

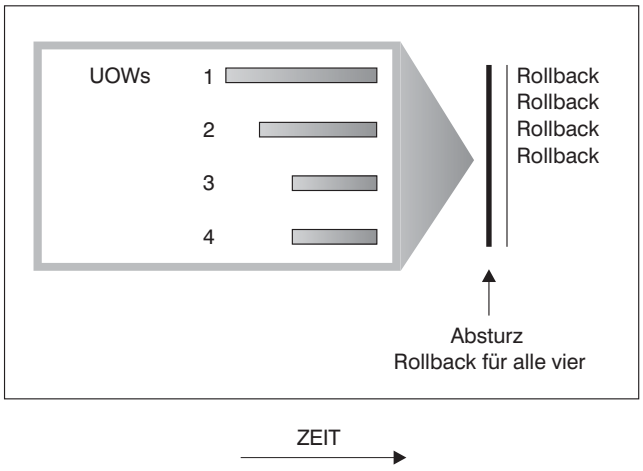


Abbildung 43. Rollback von UOWs (Recovery nach einem Systemabsturz)

Bei Verwendung von IBM DB2 pureScale Feature stehen zwei Typen von Recoverys nach Systemabsturz zur Verfügung, die Sie kennen sollten: *Recovery nach dem Absturz eines Members* und *Recovery nach dem Absturz einer Gruppe*. Die Recovery

nach dem Absturz eines Members bezeichnet den Prozess für das Durchführen einer Recovery für einen Teil der Datenbank mithilfe eines einzigen Protokolldatenstroms des Members nach einem Fehlschlagen des Members. Die Recovery nach dem Absturz eines Members, die in der Regel automatisch als Teil des Neustarts des Members initialisiert wird, ist eine Onlineoperation. Dies bedeutet, dass andere Member weiterhin auf die Datenbank zugreifen können. Es kann für mehrere Member gleichzeitig eine Recovery nach Systemabsturz eines Members durchgeführt werden. Die Recovery nach dem Absturz einer Gruppe bezeichnet den Prozess für das Durchführen einer Recovery für eine Datenbank mithilfe von Protokolldatenströmen von mehreren Members nach einem Fehlschlagen, durch das im Cluster keine funktionsfähige Cluster-Caching-Funktion mehr vorhanden ist. Die Recovery nach dem Absturz einer Gruppe wird ebenfalls automatisch initialisiert (als Teil eines Gruppenneustarts). Wie bei DB2-Operationen für die Recovery nach dem Systemabsturz außerhalb einer DB2 pureScale-Umgebung ist während der Recovery nach dem Absturz einer Gruppe der Zugriff auf die Datenbank nicht möglich.

Wenn die Datenbank oder der Datenbankmanager fehlschlägt, verbleibt die Datenbank möglicherweise in einem inkonsistenten Status. Im Inhalt der Datenbank sind möglicherweise Änderungen vorhanden, die von Transaktionen vorgenommen wurden, die während des Fehlschlagens unvollständig waren. Möglicherweise fehlen der Datenbank auch Änderungen, die von Transaktionen vorgenommen wurden, die vor dem Fehlschlagen abgeschlossen, aber noch nicht auf Platte geschrieben wurden. Es muss eine Recoveryoperation nach einem Systemabsturz ausgeführt werden, um für die nur teilweise abgeschlossenen Transaktionen einen Rollback durchzuführen und um die Änderungen von abgeschlossenen Transaktionen, die bisher nur im Hauptspeicher vorgenommen wurden, auf Platte zu schreiben.

Zu den Bedingungen, die eine Recovery nach einem Systemabsturz erforderlich machen, gehören Folgende:

- Ein Stromausfall auf der Maschine, durch den der Datenbankmanager und die Datenbankpartitionen abnormal beendet werden.
- Ein Hardwarefehler, z. B. ein Hauptspeicher-, Platten, CPU- oder Netzfehler.
- Ein schwer wiegender Betriebssystemfehler, durch den die DB2-Instanz abnormal beendet wird.

Wenn Sie möchten, dass die Recovery nach einem Systemabsturz vom Datenbankmanager automatisch vorgenommen werden soll, aktivieren Sie den Datenbankkonfigurationsparameter für automatischen Neustart (**autorestart**), indem Sie ihn auf ON setzen. (Dies ist der Standardwert.) Wenn Sie den automatischen Neustart inaktivieren möchten, setzen Sie den Datenbankkonfigurationsparameter **autorestart** auf OFF. Ist der automatische Neustart inaktiviert, müssen Sie im Fall eines Datenbankfehlers den Befehl **RESTART DATABASE** absetzen. Wenn die Datenbank-E/A vor dem Auftreten des Absturzes ausgesetzt wurde, müssen Sie mit dem Befehl **RESTART DATABASE** die Option **WRITE RESUME** angeben, damit die Recovery nach dem Systemabsturz fortgesetzt wird. Der Neustart der Datenbank wird im Protokoll mit den Benachrichtigungen für die Systemverwaltung aufgezeichnet.

Wenn auf einer Datenbank, für die die aktualisierende Recovery aktiviert ist (d. h., der Konfigurationsparameter **logarchmeth1** ist auf einen anderen Wert als OFF gesetzt), eine Recovery nach einem Systemabsturz durchgeführt wird und während dieser Recovery ein Fehler auftritt, der auf einen einzelnen Tabellenbereich zurückzuführen ist, wird dieser Tabellenbereich in den Offlinestatus versetzt. Auf ihn

kann erst wieder zugegriffen werden, nachdem er repariert wurde. Die Recovery nach dem Systemabsturz wird für andere Tabellenbereiche fortgesetzt. Nach der Beendigung der Recovery nach Systemabsturz kann auf die anderen Tabellenbereiche in der Datenbank zugegriffen werden, und es können Verbindungen zur Datenbank hergestellt werden. Wenn jedoch der in den Offlinestatus versetzte Tabellenbereich die Systemkataloge enthält, muss er repariert werden, bevor Verbindungen hergestellt werden können. Dieses Verhalten gilt nicht für DB2 pureScale-Umgebungen. Wenn während der Recovery nach dem Absturz eines Members oder einer Gruppe ein Fehler auftritt, schlägt die Operation für die Recovery nach dem Systemabsturz fehl.

Recovery nach Transaktionsfehlern in einer Umgebung mit partitionierten Datenbanken

Tritt ein Transaktionsfehler in einer Umgebung mit partitionierten Datenbanken auf, ist in der Regel eine Datenbankrecovery sowohl auf dem ausgefallenen Datenbankpartitionsserver als auch auf allen anderen, an der Transaktion beteiligten Datenbankpartitionsservern erforderlich.

Es gibt zwei Arten der Datenbankrecovery:

- Eine Recovery nach Systemabsturz wird auf dem ausgefallenen Datenbankpartitionsserver durchgeführt, nachdem die Fehlerbedingung korrigiert wurde.
- Die *Recovery der Datenbankpartitionen nach einem Fehler* erfolgt auf den anderen (weiterhin aktiven) Datenbankpartitionsservern unmittelbar nach Feststellung des Fehlers.

In einer Umgebung mit partitionierten Datenbanken ist der Datenbankpartitionsserver, auf dem die Transaktion übergeben wird, die Koordinatorpartition und der erste Agent, der die Transaktion verarbeitet, ist der Koordinatoragent. Der Koordinatoragent ist für die Verteilung der Arbeit auf andere Datenbankpartitionsserver verantwortlich und protokolliert, welche Datenbankpartitionsserver an der Transaktion beteiligt sind. Wenn die Anwendung eine COMMIT-Anweisung für eine Transaktion ausführt, schreibt der Koordinatoragent die Transaktion mithilfe des Protokolls zum zweiphasigen Commit fest. Während der ersten Phase sendet die Koordinatorpartition eine PREPARE-Anforderung an alle anderen an der Transaktion beteiligten Datenbankpartitionsserver. Die Server antworten daraufhin wie folgt:

READ-ONLY
: Auf diesem Server gab es keine Datenänderung.

YES
: Auf diesem Server gab es eine Datenänderung.

NO
: Aufgrund eines Fehlers wurde der Server nicht für das Commit vorbereitet.

Wenn einer der Server mit `NO` antwortet, wird die Transaktion rückgängig gemacht. Andernfalls beginnt die Koordinatorpartition mit der zweiten Phase.

Während der zweiten Phase schreibt die Koordinatorpartition einen Commitprotokollsatz und sendet anschließend eine Commitanforderung an alle Server, die mit `YES` geantwortet haben. Wenn alle anderen Datenbankpartitionsserver das Commit durchgeführt haben, senden sie eine Bestätigung des Commits an die Koordinatorpartition. Die Transaktion ist abgeschlossen, wenn der Koordinatoragent von allen beteiligten Servern die Commitbestätigungen empfangen hat. Wenn dies der Fall ist, schreibt der Koordinatoragent einen FORGET-Protokollsatz.

Recovery auf einem aktiven Datenbankpartitionsserver nach Transaktionsfehler

Wenn ein Datenbankpartitionsserver feststellt, dass ein anderer Server inaktiv ist, werden alle Arbeiten, an denen der ausgefallene Datenbankpartitionsserver beteiligt ist, gestoppt:

- Wenn der noch aktive Datenbankpartitionsserver die Koordinatorpartition für eine Anwendung ist und die Anwendung auf dem ausgefallenen Datenbankpartitionsserver ausgeführt wird (und nicht zum Commit bereit ist), wird der Koordinatoragent unterbrochen, um Recoverymaßnahmen durchzuführen. Wenn der Koordinatoragent in der zweiten Phase der Commitverarbeitung ist, empfängt die Anwendung die Nachricht SQL0279N und verliert die Datenbankverbindung. Ansonsten sendet der Koordinatoragent eine Rollback-Anforderung an alle an der Transaktion beteiligten Server und die Anwendung empfängt die Nachricht SQL1229N.
- Wenn der ausgefallene Datenbankpartitionsserver die Koordinatorpartition für die Anwendung war, werden Agenten, die noch für die Anwendung auf den aktiven Servern arbeiten, unterbrochen, um Recoverymaßnahmen durchzuführen. Die Transaktion wird lokal auf jeder Datenbankpartition rückgängig gemacht, auf der sich die Transaktion nicht im Status vorbereiteter Transaktionen befindet. Auf den Datenbankpartitionen, auf denen sich die Transaktion im Status vorbereiteter Transaktionen befindet, wird die Transaktion zu einer unbestätigten Transaktion. Die Koordinatordatenbankpartition ist nicht darüber informiert, dass die Transaktion auf einigen Datenbankpartitionen unbestätigt ist, weil sie nicht verfügbar ist.
- Wenn die Anwendung die Verbindung zu dem ausgefallenen Datenbankpartitionsserver (bevor er ausfiel) herstellte, aber weder der lokale Datenbankpartitionsserver noch der ausgefallene Datenbankpartitionsserver die Koordinatorpartition ist, werden Agenten, die für diese Anwendung aktiv sind, unterbrochen. Die Koordinatorpartition sendet eine Nachricht über eine Rollback-Operation (ROLLBACK) oder eine Trennoperation (DISCONNECT) an die anderen Datenbankpartitionsserver. Die Transaktion ist nur auf den Datenbankpartitionsservern unbestätigt, die weiterhin aktiv sind, wenn die Koordinatorpartition die Nachricht SQL0279 zurückgibt.

Jeder Prozess (wie z. B. ein Agent oder ein Deadlock-Detektor), der versucht, eine Anforderung an den ausgefallenen Server zu senden, wird informiert, dass er die Anforderung nicht senden kann.

Recovery nach Transaktionsfehler auf dem ausgefallenen Datenbankpartitionsserver

Wenn der Transaktionsfehler zu einer abnormalen Beendigung des Datenbankmanagers führt, können Sie den Befehl **db2start** mit der Option RESTART angeben, um den Datenbankmanager nach dem Neustart der Datenbankpartition erneut zu starten. Falls der Neustart der Datenbankpartition nicht möglich ist, können Sie den Befehl **db2start** auch in einer anderen Datenbankpartition zum Starten des Datenbankmanagers verwenden.

Die abnormale Beendigung des Datenbankmanagers kann zur Inkonsistenz einiger Datenbankpartitionen auf dem Server führen. Um diese Datenbankpartitionen wieder in einen konsistenten Status zu versetzen, kann auf einem Datenbankpartitionsserver wie folgt eine Recovery nach einem Systemabsturz ausgelöst werden:

- Explizit durch den Befehl **RESTART DATABASE**

- Implizit durch eine CONNECT-Anforderung, wenn der Datenbankkonfigurationsparameter *autorestart* auf ON gesetzt ist

Bei der Recovery nach einem Systemabsturz werden die Protokollsätze in den aktiven Protokolldateien erneut angewendet, um sicherzustellen, dass die Ergebnisse aller vollständigen Transaktionen in der Datenbank vorhanden sind. Nachdem alle Änderungen erneut angewendet wurden, werden alle nicht festgeschriebenen Transaktionen *außer* unbestätigten Transaktionen lokal rückgängig gemacht. In einer Umgebung mit partitionierten Datenbanken gibt es zwei Typen unbestätigter Transaktionen:

- Auf einem Datenbankpartitionsserver, der nicht die Koordinatorpartition ist, gilt eine Transaktion als unbestätigt, wenn sie zwar vorbereitet (PREPARE), aber noch nicht festgeschrieben (COMMIT) wurde.
- In der Koordinatorpartition ist eine Transaktion unbestätigt, wenn sie festgeschrieben (COMMIT), aber noch nicht als abgeschlossen protokolliert wurde (d. h., der Protokollsatz FORGET wurde noch nicht geschrieben). Diese Situation tritt ein, wenn der Koordinatoragent noch nicht alle Commitbestätigungen von allen Servern empfangen hat, die für die Anwendung aktiv waren.

Bei der Recovery nach einem Systemabsturz wird versucht, alle unbestätigten Transaktionen durch eine der folgenden Aktionen aufzulösen. Die durchgeführte Aktion ist davon abhängig, ob der Datenbankpartitionsserver die Koordinatorpartition für eine Anwendung war:

- Wenn der Server, der erneut gestartet wird, nicht die Koordinatorpartition für die Anwendung ist, sendet er eine Abfragenachricht an den Koordinatoragenten, um das Ergebnis der Transaktion festzustellen.
- Wenn der erneut gestartete Server die Koordinatorpartition für die Anwendung *ist*, sendet er eine Nachricht an alle anderen Agenten (untergeordneten Agenten), von denen der Koordinatoragent immer noch Commitbestätigungen erwartet.

Es ist möglich, dass durch eine Recovery nach einem Systemabsturz nicht alle unbestätigten Transaktionen aufgelöst werden können. Einige der Datenbankpartitionsserver sind z. B. möglicherweise nicht verfügbar. Wenn die Koordinatorpartition die Recovery nach einem Systemabsturz vor den anderen, an der Transaktion beteiligen Datenbankpartitionen abschließt, kann die Recovery nach einem Systemabsturz die unbestätigten Transaktionen nicht auflösen. Dies ist die erwartete Funktionsweise, da die Recovery nach einem Systemabsturz von jeder Datenbankpartition unabhängig durchgeführt wird. In diesem Fall wird die SQL-Warnung SQL1061W zurückgegeben. Unbestätigte Transaktionen belegen Ressourcen, z. B. Sperren und Speicherbereich für aktive Protokolle. Daher ist es möglich, dass ein Punkt erreicht wird, an dem keine Änderungen an der Datenbank mehr durchgeführt werden können, weil der Speicherbereich für die aktiven Protokolldateien durch unbestätigte Transaktionen belegt ist. Aus diesem Grund sollten Sie nach einer Recovery nach einem Systemabsturz feststellen, ob unbestätigte Transaktionen verblieben sind, und alle Datenbankpartitionsserver, die zur Auflösung der unbestätigten Transaktionen erforderlich sind, so schnell wie möglich wieder verfügbar machen.

Anmerkung: In einer Umgebung mit partitionierten Datenbankservern wird der Befehl RESTART DATABASE auf jedem Knoten einzeln ausgeführt. Um sicherzustellen, dass die Datenbank auf allen Knoten erneut gestartet wird, verwenden Sie den folgenden empfohlenen Befehl:

```
db2_all "db2 restart database <datenbankname>"
```

Wenn mindestens ein Server, der zur Auflösung einer unbestätigten Transaktion benötigt wird, nicht rechtzeitig wieder verfügbar gemacht werden kann und der Zugriff auf Datenbankpartitionen auf anderen Servern erforderlich ist, können Sie die unbestätigte Transaktion durch eine heuristische Entscheidung manuell auflösen. Mithilfe des Befehls **LIST INDOUBT TRANSACTIONS** können Sie die unbestätigte Transaktion auf dem Server abfragen, festschreiben oder rückgängig machen.

Anmerkung: Der Befehl **LIST INDOUBT TRANSACTIONS** wird auch in einer verteilten Transaktionsumgebung verwendet. Zur Unterscheidung zwischen den beiden Typen unbestätigter Transaktionen enthält das Feld für die Quelle (*Originator*) in der Ausgabe des Befehls **LIST INDOUBT TRANSACTIONS** eine der folgenden Angaben:

- DB2 Enterprise Server Edition. Dies zeigt an, dass die Transaktion aus einer Umgebung mit partitionierten Datenbanken stammt.
- XA. Dies zeigt an, dass die Transaktion aus einer verteilten Umgebung stammt.

Identifizieren des ausgefallenen Datenbankpartitionsservers

Wenn ein Datenbankpartitionsserver ausfällt, empfängt die Anwendung normalerweise einen der folgenden SQLCODE-Werte. Die Methode zum Identifizieren des jeweils ausgefallenen Datenbankmanagers hängt vom empfangenen SQLCODE-Wert ab:

SQL0279N
Dieser SQLCODE-Wert wird empfangen, wenn ein Datenbankpartitionsserver, der an einer Transaktion beteiligt ist, während der Commitverarbeitung beendet wird.

SQL1224N
Dieser SQLCODE-Wert wird empfangen, wenn der ausgefallene Datenbankpartitionsserver die Koordinatorpartition für die Transaktion ist.

SQL1229N
Dieser SQLCODE-Wert wird empfangen, wenn der ausgefallene Datenbankpartitionsserver nicht die Koordinatorpartition für die Transaktion ist.

Welcher Datenbankpartitionsserver ausgefallen ist, kann in zwei Schritten festgestellt werden.

1. Suchen Sie im SQL-Kommunikationsbereich den Partitionsserver, der den Fehler festgestellt hat. Der SQL-Kommunikationsbereich, der zum SQLCODE-Wert SQL1229N gehört, enthält in der sechsten Feldposition des Felds *sqlerrd* die Knotennummer des Servers, der den Fehler erkannte. (Die Knotennummer, die für den Server geschrieben wird, entspricht der Knotennummer in der Datei `db2nodes.cfg`.)
2. Ermitteln Sie für den in Schritt 1 gefundenen Server im Protokoll mit Benachrichtigungen für die Systemverwaltung die Knotennummer des ausgefallenen Servers.

Anmerkung: Wenn mehrere logische Knoten auf einem Prozessor verwendet werden, kann der Ausfall eines logischen Knotens den Ausfall anderer logischer Knoten auf demselben Prozessor verursachen.

Recovery nach dem Ausfall eines Datenbankpartitionsservers

Führen Sie eine Recovery nach dem Ausfall eines Datenbankpartitionsservers durch, indem Sie den zugrunde liegenden Fehler ermitteln und beheben.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Recovery nach dem Ausfall eines Datenbankpartitionsservers auszuführen.

1. Beheben Sie den Fehler, der den Ausfall verursachte.
2. Starten Sie den Datenbankmanager erneut, indem Sie auf einem beliebigen Datenbankpartitionsserver den Befehl **db2start** absetzen.
3. Starten Sie die Datenbank erneut, indem Sie auf dem bzw. den ausgefallenen Datenbankpartitionsserver(n) den Befehl **RESTART DATABASE** absetzen.

Erneutes Erstellen von partitionierten Datenbanken

Zur erneuten Erstellung (Rebuild) einer partitionierten Datenbank erstellen Sie jede Datenbankpartition separat erneut. Für jede Datenbankpartition, angefangen bei der Katalogpartition, stellen Sie zunächst alle Tabellenbereiche, die Sie benötigen, mit RESTORE wieder her. Alle Tabellenbereiche, die nicht wiederhergestellt werden, werden in den Status *Restore anstehend* versetzt.

Wenn alle Datenbankpartitionen wiederhergestellt sind, setzen Sie den Befehl **ROLLFORWARD DATABASE** in der Katalogpartition ab, um eine aktualisierende Recovery für alle Datenbankpartitionen auszuführen.

Informationen zu diesem Vorgang

Anmerkung: Wenn Sie zu einem späteren Zeitpunkt Tabellenbereiche wiederherstellen müssen, die ursprünglich nicht in die Rebuildphase mit eingeschlossen waren, müssen Sie sicherstellen, dass bei der nachfolgenden aktualisierenden Recovery des Tabellenbereichs das Dienstprogramm für aktualisierende Recovery (Rollforward) alle Daten über die Datenbankpartitionen hinweg synchron hält. Wenn ein Tabellenbereich bei der ursprünglichen Restore- und aktualisierenden Recoveryoperation versehentlich unberücksichtigt bleibt, wird sein Fehlen unter Umständen erst erkannt, wenn bei einem Versuch, auf die Daten zuzugreifen, ein Datenzugriffsfehler auftritt. Sie müssen den fehlenden Tabellenbereich mit Restore wiederherstellen und eine aktualisierende Recovery für ihn ausführen, um ihn mit den übrigen Partitionen zu resynchronisieren.

Betrachten Sie das folgende Beispiel im Hinblick auf die erneute Erstellung einer partitionierten Datenbank mithilfe von Backup-Images, die auf Tabellenbereichsebene erstellt wurden.

Für dieses Beispiel wird angenommen, dass eine wiederherstellbare Datenbank mit dem Namen SAMPLE vorhanden ist und drei Datenbankpartitionen enthält:

- Datenbankpartition 1 enthält die Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2 und ist die Katalogpartition.
- Datenbankpartition 2 enthält die Tabellenbereiche USERSP1 und USERSP3.
- Datenbankpartition 3 enthält die Tabellenbereiche USERSP1, USERSP2 und USERSP3.

Die folgenden Backups wurden erstellt, wobei BK*xy* die Backupnummer *x* in Partition *y* bezeichnet:

- BK11 ist ein Backup der Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2.
- BK12 ist ein Backup der Tabellenbereiche USERSP2 und USERSP3.
- BK13 ist ein Backup der Tabellenbereiche USERSP1, USERSP2 und USERSP3.

- BK21 ist ein Backup des Tabellenbereichs USERSP1.
- BK22 ist ein Backup des Tabellenbereichs USERSP1.
- BK23 ist ein Backup des Tabellenbereichs USERSP1.
- BK31 ist ein Backup des Tabellenbereichs USERSP2.
- BK33 ist ein Backup des Tabellenbereichs USERSP2.
- BK42 ist ein Backup des Tabellenbereichs USERSP3.
- BK43 ist ein Backup des Tabellenbereichs USERSP3.

Die folgende Prozedur veranschaulicht die Verwendung der Befehle **RESTORE DATABASE** und **ROLLFORWARD DATABASE** über den Befehlszeilenprozessor (CLP) zur erneuten Erstellung der gesamten Datenbank bis zum Ende der Protokolle.

Vorgehensweise

1. Führen Sie in Datenbankpartition 1 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK31 without prompting
   ```

2. Führen Sie in Datenbankpartition 2 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db sample rebuild with tablespaces in database
         taken at BK42 without prompting
   ```

3. Führen Sie in Datenbankpartition 3 einen Befehl **RESTORE DATABASE** mit der Option **REBUILD** aus:

   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK43 without prompting
   ```

4. Führen Sie in der Katalogpartition einen Befehl **ROLLFORWARD DATABASE** mit der Option **TO END OF LOGS** aus:

   ```
   db2 rollforward db sample to end of logs
   ```

5. Führen Sie einen Befehl **ROLLFORWARD DATABASE** mit der Option **STOP** aus:

   ```
   db2 rollforward db sample stop
   ```

Nächste Schritte

An diesem Punkt kann eine Verbindung zur Datenbank in allen Datenbankpartitionen hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Datenrecovery mit 'db2adutl'

Sie können eine knotenübergreifende Recovery mit dem Befehl **db2adutl** sowie mithilfe der Datenbankkonfigurationsparameter **logarchopt1** und **vendoropt** ausführen. Diese Recovery wird in Beispielen in verschiedenen TSM-Umgebungen (TSM = Tivoli Storage Manager) ausgeführt.

In den folgenden Beispielen besitzt Computer 1 den Namen `bar` und wird unter dem Betriebssystem AIX betrieben. Der Benutzer dieser Maschine ist `roecken`. Die Datenbank auf der Maschine `bar` heißt `zample`. Computer 2 besitzt den Namen `dps`. Dieser Computer wird ebenfalls unter dem Betriebssystem AIX von dem Benutzer `regress9` ausgeführt.

Beispiel 1: Automatische Kennwortverwaltung durch den TSM-Server (die Option PASSWORDACCESS ist auf GENERATE gesetzt)

Dieses Beispiel für knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

1. Um die Datenbank für Protokollarchivierung für den Computer bar auf dem TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter **logarchmeth1** für die Datenbank zample mit dem folgenden Befehl:

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```

2. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:

   ```
   db2 force applications all
   ```

3. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:

   ```
   db2 list applications
   ```

   Sie müssten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen.

4. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:

   ```
   db2 backup db zample use tsm
   ```

   Informationen ähnlich den folgenden werden zurückgegeben:

   ```
   Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
   ```

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen. Die Reihenfolge, in der Sie diesen Schritt in den Datenbankpartitionen ausführen, variiert, je nachdem, ob Sie ein Online-Backup oder ein Offline-Backup durchführen. Weitere Informationen hierzu finden Sie in „Backup von Daten" auf Seite 475.

5. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank zample her:

   ```
   db2 connect to zample
   ```

6. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace
       into employee copy yes use tsm
   ```

   In diesem Beispiel hat die Tabelle den Namen employee, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen mr hat. Die Option **COPY** YES wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option **USE TSM** wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

Anmerkung: Sie können die Option **COPY** YES nur angeben, wenn die Datenbank für die aktualisierende Recovery aktiviert ist. Das heißt, der Datenbankkonfigurationsparameter **logarchmeth1** muss auf den Wert USEREXIT, LOGRETAIN, DISK oder TSM gesetzt sein.

Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

```
SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
"/home/roecken/mr".

SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
15:12:13.392633").

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
aus der Eingabedatei gelesen.

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
15:12:13.445718").


   Anzahl gelesener Zeilen         = 1
   Anzahl übersprungener Zeilen    = 0
   Anzahl geladener Zeilen         = 1
   Anzahl zurückgewiesener Zeilen  = 0
   Anzahl gelöschter Zeilen        = 0
   Anzahl festgeschriebener Zeilen = 1
```

7. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank zample ausführen:

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl query db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
           Sessions: 1

   Retrieving INCREMENTAL DATABASE BACKUP information.
       No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

   Retrieving DELTA DATABASE BACKUP information.
       No DELTA DATABASE BACKUP images found for ZAMPLE

   Retrieving TABLESPACE BACKUP information.
       No TABLESPACE BACKUP images found for ZAMPLE

   Retrieving INCREMENTAL TABLESPACE BACKUP information.
       No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

   Retrieving DELTA TABLESPACE BACKUP information.
       No DELTA TABLESPACE BACKUP images found for ZAMPLE

   Retrieving LOAD COPY information.
    1 Time: 20090216151213

   Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
   ```

8. Um die knotenübergreifende Recovery zu aktivieren, müssen Sie den Zugriff auf die zugeordneten Objekte des Computers bar für einen anderen Computer

und ein anderes Benutzerkonto erteilen. Erteilen Sie in diesem Beispiel den Zugriff für den Computer dps und den Benutzer regress9 mit dem folgenden Befehl:

```
bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
         on nodename dps for db zample
```

Die folgenden Informationen werden zurückgegeben:

```
Successfully added permissions for regress9 to access ZAMPLE on node dps.
```

Anmerkung: Sie können die Ergebnisse der GRANT-Operation für **db2adutl** überprüfen, indem Sie mit dem folgenden Befehl die aktuelle Zugriffsliste für den aktuellen Knoten abrufen:

```
bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
```

Die folgenden Informationen werden zurückgegeben:

```
Node                Username            Database Name   Type
-----------------------------------------------------------------
DPS                 regress9            ZAMPLE          A
-----------------------------------------------------------------
 Access Types:  B - backup images   L - logs   A - both
```

9. In diesem Beispiel ist der zweite Computer (dps) noch nicht für die knotenübergreifende Recovery der Datenbank zample konfiguriert. Vergewissern Sie sich mit dem folgenden Befehl, dass diesem Benutzer und diesem Computer keine Daten auf dem TSM-Server zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
     --- Database directory is empty ---
   Warning: There are no file spaces created by DB2 on the ADSM server
   Warning: No DB2 backup images found in ADSM for any alias.
   ```

10. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank zample ab, die dem Benutzer roecken und dem Computer bar zugeordnet sind:

    ```
    dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
               bar owner roecken
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
        --- Database directory is empty ---

    Query for database ZAMPLE

       Retrieving FULL DATABASE BACKUP information.
        1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
               Sessions: 1

       Retrieving INCREMENTAL DATABASE BACKUP information.
           No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

       Retrieving DELTA DATABASE BACKUP information.
           No DELTA DATABASE BACKUP images found for ZAMPLE

       Retrieving TABLESPACE BACKUP information.
           No TABLESPACE BACKUP images found for ZAMPLE

       Retrieving INCREMENTAL TABLESPACE BACKUP information.
           No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

       Retrieving DELTA TABLESPACE BACKUP information.
           No DELTA TABLESPACE BACKUP images found for ZAMPLE

       Retrieving LOAD COPY information.
        1 Time: 20090216151213
    ```

```
Retrieving LOG ARCHIVE information.
Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
Taken at: 2009-02-16-15.10.38
```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Computer dps wiederhergestellt werden kann.

11. Stellen Sie mit dem folgenden Befehl die Datenbank zample vom TSM-Server auf dem Computer dps wieder her:

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
```

Die folgenden Informationen werden zurückgegeben:

```
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Anmerkung: Wenn die Datenbank zample auf dps bereits vorhanden wäre, würde der Parameter **OPTIONS** weggelassen, und der Datenbankkonfigurationsparameter **vendoropt** würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter **OPTIONS** für eine BACKUP- oder RESTORE-Operation.

12. Führen Sie eine aktualisierende Recovery durch, um die Transaktionen anzuwenden, die beim Erstellen einer neuen Tabelle und beim Laden neuer Daten in der Protokolldatei für die Datenbank zample erfasst wurden. In diesem Beispiel schlägt die aktualisierende Recovery (ROLLFORWARD) fehl, weil das Dienstprogramm für die aktualisierende Recovery die Protokolldateien nicht finden kann, da keine Benutzer- und Computerinformationen angegeben sind:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Der Befehl gibt den folgenden Fehler zurück:

```
SQL4970N  Die aktualisierende Recovery der Datenbank "ZAMPLE"
kann wegen fehlender Protokolldatei(en) auf Knoten "0" nicht den angegebenen
Endpunkt (Protokollende oder angegebener Zeitpunkt) erreichen.
```

Veranlassen Sie das Dienstprogramm für aktualisierende Recovery, die Protokolldateien für einen anderen Computer zu suchen, indem Sie den entsprechenden Wert für **logarchopt** angeben. Verwenden Sie in diesem Beispiel den folgenden Befehl, um den Datenbankkonfigurationsparameter **logarchopt1** festzulegen und nach Protokolldateien für den Benutzer roecken und den Computer bar zu suchen:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
```

13. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Recovery die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter **vendoropt** festlegen:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
```

14. Mit dem folgenden Befehl können Sie die knotenübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                                Status der aktualisierenden Recovery

    Aliasname der Eingabedatenbank             = zample
Anzahl der Member, die Status zurückgaben   = 1


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
```

```
            0      nicht anstehend          S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC

  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 2: Kennwortverwaltung durch den Benutzer (die Option PASSWORDACCESS ist auf PROMPT gesetzt)

Dieses Beispiel für knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter von den Benutzern verwaltet werden. In diesen Umgebungen sind zusätzliche Informationen erforderlich. Dies sind insbesondere der TSM-Knotenname und das Kennwort für den Computer, auf dem die Objekte erstellt wurden.

1. Fügen Sie in der Datei dsm.sys die folgende Zeile hinzu, weil der Quellencomputer den Namen bar trägt:

   ```
   NODENAME bar
   ```

   Anmerkung: Auf Windows-Betriebssystemen heißt diese Datei dsm.opt. Nach dem Aktualisieren dieser Datei müssen Sie für Ihr System einen Warmstart durchführen, damit die Änderungen wirksam werden.

2. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte ab, die dem Benutzer roecken und dem Computer bar zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
      owner roecken password *******
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Query for database ZAMPLE

      Retrieving FULL DATABASE BACKUP information.
       1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
              Sessions: 1

      Retrieving INCREMENTAL DATABASE BACKUP information.
          No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

      Retrieving DELTA DATABASE BACKUP information.
          No DELTA DATABASE BACKUP images found for ZAMPLE

      Retrieving TABLESPACE BACKUP information.
          No TABLESPACE BACKUP images found for ZAMPLE

      Retrieving INCREMENTAL TABLESPACE BACKUP information.
          No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

      Retrieving DELTA TABLESPACE BACKUP information.
          No DELTA TABLESPACE BACKUP images found for ZAMPLE

      Retrieving LOAD COPY information.
       1 Time: 20090216151213

      Retrieving LOG ARCHIVE information.
      Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
      Taken at: 2009-02-16-15.10.38
   ```

3. Führen Sie die folgenden Schritte aus, wenn die Datenbank zample auf dem Computer dps noch nicht vorhanden ist:

   a. Erstellen Sie mit dem folgenden Befehl eine leere Datenbank zample:

      ```
      dps:/home/regress9> db2 create db zample
      ```

b. Aktualisieren Sie den Datenbankkonfigurationsparameter **tsm_nodename** mit dem folgenden Befehl:

```
dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
```

c. Aktualisieren Sie den Datenbankkonfigurationsparameter **tsm_password** mit dem folgenden Befehl:

```
dps:/home/regress9> db2 update db cfg for zample using
   tsm_password ********
```

4. Versuchen Sie mit dem folgenden Befehl, die Datenbank zample wiederherzustellen:

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
```

Die Restoreoperation wird erfolgreich ausgeführt, jedoch wird eine Warnung ausgeben:

```
   SQL2540W  RESTORE wurde erfolgreich ausgeführt, es wurde jedoch eine
Warnung "2523" beim Restore der Datenbank im Modus
'No Interrupt' festgestellt.
```

5. Führen Sie mit dem folgenden Befehl eine aktualisierende Recovery durch:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Weil in diesem Beispiel die Datenbankkonfigurationsdatei durch die Restoreoperation ersetzt wurde, kann das Dienstprogramm zur aktualisierenden Recovery nicht die richtigen Protokolldateien finden und die folgende Fehlernachricht wird zurückgegeben:

```
SQL1268N  Die aktualisierende Recovery wurde nach dem
Fehler "-2112880618" beendet, während die Protokolldatei "S0000000.LOG"
für die Datenbank "ZAMPLE" auf dem Knoten "0" abgerufen wurde.
```

Setzen Sie die folgenden TSM-Datenbankkonfigurationsparameter auf die richtigen Werte zurück:

a. Legen Sie mit dem folgenden Befehl den Konfigurationsparameter **tsm_nodename** fest:

```
dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
```

b. Legen Sie mit dem folgenden Befehl den Datenbankkonfigurationsparameter **tsm_password** fest:

```
dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
```

c. Legen Sie den Datenbankkonfigurationsparameter **logarchopt1** so fest, dass vom Dienstprogramm für aktualisierende Recovery die richtigen Protokolldateien gefunden werden; verwenden Sie hierfür den folgenden Befehl:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
```

d. Legen Sie mit dem folgenden Befehl den Datenbankkonfigurationsparameter **vendoropt** so fest, dass die Recoverydatei auch während der aktualisierenden Recovery verwendet werden kann:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
```

6. Mit dem folgenden Befehl können Sie die knotenübergreifende Recovery durch das Ausführen der aktualisierenden Recovery beenden:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                                Status der aktualisierenden Recovery

    Aliasname der Eingabedatenbank              = zample
Anzahl der Member, die Status zurückgaben   = 1


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
```

```
              0      nicht anstehend          S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC

      DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 3: Konfigurieren des TSM-Servers für die Verwendung von Client-Proxy-Knoten

Dieses Beispiel für die knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern als Proxy-Knoten für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

In diesem Beispiel werden die Computer bar und dps unter dem Proxy-Namen clusternode registriert. Die Computer sind bereits als Proxy-Knoten konfiguriert.

1. Registrieren Sie mit den folgenden Befehlen die Computer bar und dps als Proxy-Knoten auf dem TSM-Server:
   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=bar,dps
   ```
2. Um die Datenbank für die Protokollarchivierung für den TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter **logarchmeth1** für die Datenbank zample mit dem folgenden Befehl:
   ```
   bar:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```
   Die folgenden Informationen werden zurückgegeben:
   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```
3. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:
   ```
   db2 force applications all
   ```
4. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:
   ```
   db2 list applications
   ```
   Sie müssten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen.
5. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:
   ```
   db2 backup db zample use tsm options "'-asnodename=clusternode'"
   ```
   Informationen ähnlich den folgenden werden zurückgegeben:
   ```
   Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
   ```
   Anstatt die Option **-asnodename** im Befehl **BACKUP DATABASE** anzugeben, können Sie den Datenbankkonfigurationsparameter **vendoropt** aktualisieren.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen. Die Reihenfolge, in der Sie diesen Schritt an den Datenbankpartitionen ausführen, variiert, je nachdem, ob Sie ein Online-Backup oder ein Offline-Backup durchführen. Weitere Informationen hierzu finden Sie in „Backup von Daten" auf Seite 475.

6. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank `zample` her:

   ```
   db2 connect to zample
   ```

7. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace       into employee copy yes use
   ```

   In diesem Beispiel hat die Tabelle den Namen `employee`, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen `mr` hat. Die Option **COPY** `YES` wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option **USE TSM** wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

   Anmerkung: Sie können die Option **COPY** `YES` nur angeben, wenn die Datenbank für die aktualisierende Recovery aktiviert ist. Das heißt, der Datenbankkonfigurationsparameter **logarchmeth1** muss auf den Wert `USEREXIT`, `LOGRETAIN`, `DISK` oder `TSM` gesetzt sein.

   Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

   ```
   SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
   "/home/roecken/mr".

   SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
   15:12:13.392633").

   SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

   SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

   SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
   aus der Eingabedatei gelesen.

   SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

   SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

   SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
   15:12:13.445718").


      Anzahl gelesener Zeilen         = 1
      Anzahl übersprungener Zeilen    = 0
      Anzahl geladener Zeilen         = 1
      Anzahl zurückgewiesener Zeilen  = 0
      Anzahl gelöschter Zeilen        = 0
      Anzahl festgeschriebener Zeilen = 1
   ```

8. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank `zample` ausführen:

   ```
      bar:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
        Retrieving FULL DATABASE BACKUP information.
         1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
                Sessions: 1
   ```

```
Retrieving INCREMENTAL DATABASE BACKUP information.
    No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
    No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
    No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
    No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
    No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
 1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
Taken at: 2009-02-16-15.10.38
```

9. In diesem Beispiel ist der zweite Computer (dps) noch nicht für die knotenübergreifende Recovery der Datenbank zample konfiguriert. Vergewissern Sie sich mit dem folgenden Befehl, dass diesem Benutzer und diesem Computer keine Daten zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
     --- Database directory is empty ---
   Warning: There are no file spaces created by DB2 on the ADSM server
   Warning: No DB2 backup images found in ADSM for any alias.
   ```

10. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank zample ab, die dem Proxy-Knoten clusternode zugeordnet sind:

    ```
    dps:/home/regress9/sqllib/adsm> db2adutl query db zample
       options="-asnodename=clusternode"
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
    --- Database directory is empty ---

  Query for database ZAMPLE

    Retrieving FULL DATABASE BACKUP information.
     1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
            Sessions: 1

    Retrieving INCREMENTAL DATABASE BACKUP information.
        No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

    Retrieving DELTA DATABASE BACKUP information.
        No DELTA DATABASE BACKUP images found for ZAMPLE

    Retrieving TABLESPACE BACKUP information.
        No TABLESPACE BACKUP images found for ZAMPLE

    Retrieving INCREMENTAL TABLESPACE BACKUP information.
        No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

    Retrieving DELTA TABLESPACE BACKUP information.
        No DELTA TABLESPACE BACKUP images found for ZAMPLE

    Retrieving LOAD COPY information.
     1 Time: 20090216151213

    Retrieving LOG ARCHIVE information.
    Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
    Taken at: 2009-02-16-15.10.38
    ```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Computer dps wiederhergestellt werden kann.

11. Stellen Sie mit dem folgenden Befehl die Datenbank zample vom TSM-Server auf dem Computer dps wieder her:

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-asnodename=clusternode'" without prompting
```

Die folgenden Informationen werden zurückgegeben:

```
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Anmerkung: Wenn die Datenbank zample auf dps bereits vorhanden wäre, würde der Parameter **OPTIONS** weggelassen, und der Datenbankkonfigurationsparameter **vendoropt** würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter **OPTIONS** für eine BACKUP- oder RESTORE-Operation.

12. Führen Sie eine aktualisierende Recovery durch, um die Transaktionen anzuwenden, die beim Erstellen einer neuen Tabelle und beim Laden neuer Daten in der Protokolldatei für die Datenbank zample erfasst wurden. In diesem Beispiel schlägt die aktualisierende Recovery (ROLLFORWARD) fehl, weil das Dienstprogramm für die aktualisierende Recovery die Protokolldateien nicht finden kann, da keine Benutzer- und Computerinformationen angegeben sind:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Der Befehl gibt den folgenden Fehler zurück:

```
SQL4970N  Die aktualisierende Recovery der Datenbank "ZAMPLE"
kann wegen fehlender Protokolldatei(en) auf Knoten "0" nicht den angegebenen
Endpunkt (Protokollende oder angegebener Zeitpunkt) erreichen.
```

Veranlassen Sie das Dienstprogramm für aktualisierende Recovery, die Protokolldateien auf einem anderen Computer zu suchen, indem Sie den entsprechenden Wert für **logarchopt** angeben. Verwenden Sie in diesem Beispiel den folgenden Befehl, um den Datenbankkonfigurationsparameter **logarchopt1** festzulegen und nach Protokolldateien für den Benutzer roecken und den Computer bar zu suchen:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-asnodename=clusternode'"
```

13. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Recovery die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter **vendoropt** festlegen:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

14. Mit dem folgenden Befehl können Sie die knotenübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                                Status der aktualisierenden Recovery

    Aliasname der Eingabedatenbank             = zample
Anzahl der Member, die Status zurückgaben   = 1


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
        0      nicht anstehend           S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 4: Konfigurieren des TSM-Servers für die Verwendung von Client-Proxy-Knoten in einer DB2 pureScale-Umgebung

Dieses Beispiel zeigt die Vorgehensweise zum Einrichten von zwei Membern als Proxy-Knoten für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

In diesem Beispiel werden die Member member1 und member2 unter dem Proxy-Namen clusternode registriert. In DB2 pureScale-Umgebungen können Sie Backup- oder Datenrecoveryoperationen von jedem Member aus ausführen. In diesem Beispiel wird die Datenrevocery von dem Member member2 aus ausgeführt.

1. Registrieren Sie mit den folgenden Befehlen die Member member1 und member2 als Proxy-Knoten auf dem TSM-Server:

   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=member1,member2
   ```

2. Um die Datenbank für die Protokollarchivierung für den TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter **logarchmeth1** für die Datenbank zample mit dem folgenden Befehl:

   ```
   member1:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```

   Anmerkung: In DB2 pureScale-Umgebungen können Sie die globalen **logarchmeth1**-Datenbankkonfigurationsparameter einmal (1) von einem beliebigen Member aus festlegen.

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```

3. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:

   ```
   db2 force applications all
   ```

4. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:

   ```
   db2 list applications global
   ```

   Sie müssten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

5. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:

   ```
   db2 backup db zample use tsm options '-asnodename=clusternode'
   ```

   Informationen ähnlich den folgenden werden zurückgegeben:

   ```
   Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
   ```

   Anstatt die Option **-asnodename** im Befehl **BACKUP DATABASE** anzugeben, können Sie den Datenbankkonfigurationsparameter **vendoropt** aktualisieren.

Anmerkung: In DB2 pureScale-Umgebungen können Sie diesen Befehl zum Ausführen eines Backups aller Daten für die Datenbank von einem beliebigen Member aus ausführen.

6. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank zample her:

```
db2 connect to zample
```

7. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

```
member1:/home/roecken> db2 load from mr of del modified by noheader replace
      into employee copy yes use tsmwhere
```

In diesem Beispiel hat die Tabelle den Namen employee, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen mr hat. Die Option **COPY** YES wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option **USE TSM** wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

Anmerkung: Sie können die Option **COPY** YES nur angeben, wenn die Datenbank für die aktualisierende Recovery aktiviert ist. Das heißt, der Datenbankkonfigurationsparameter **logarchmeth1** muss auf den Wert USEREXIT, LOGRETAIN, DISK oder TSM gesetzt sein.

Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

```
SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
"/home/roecken/mr".

SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
15:12:13.392633").

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
aus der Eingabedatei gelesen.

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
15:12:13.445718").


   Anzahl gelesener Zeilen         = 1
   Anzahl übersprungener Zeilen    = 0
   Anzahl geladener Zeilen         = 1
   Anzahl zurückgewiesener Zeilen  = 0
   Anzahl gelöschter Zeilen        = 0
   Anzahl festgeschriebener Zeilen = 1
```

8. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank zample ausführen:

```
member1:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

Die folgenden Informationen werden zurückgegeben:

```
   Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
           Sessions: 1
```

```
   Retrieving INCREMENTAL DATABASE BACKUP information.
       No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

   Retrieving DELTA DATABASE BACKUP information.
       No DELTA DATABASE BACKUP images found for ZAMPLE

   Retrieving TABLESPACE BACKUP information.
       No TABLESPACE BACKUP images found for ZAMPLE

   Retrieving INCREMENTAL TABLESPACE BACKUP information.
       No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

   Retrieving DELTA TABLESPACE BACKUP information.
       No DELTA TABLESPACE BACKUP images found for ZAMPLE

   Retrieving LOAD COPY information.
    1 Time: 20090216151213

Retrieving LOG ARCHIVE information.

Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15

Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

9. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank `zample` ab, die dem Proxy-Knoten `clusternode` zugeordnet sind:

```
member2:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

   Die folgenden Informationen werden zurückgegeben:

```
     --- Database directory is empty ---

Query for database ZAMPLE

   Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 0
           Sessions: 1

   Retrieving INCREMENTAL DATABASE BACKUP information.
       No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

   Retrieving DELTA DATABASE BACKUP information.
       No DELTA DATABASE BACKUP images found for ZAMPLE

   Retrieving TABLESPACE BACKUP information.
       No TABLESPACE BACKUP images found for ZAMPLE

   Retrieving INCREMENTAL TABLESPACE BACKUP information.
       No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

   Retrieving DELTA TABLESPACE BACKUP information.
       No DELTA TABLESPACE BACKUP images found for ZAMPLE

   Retrieving LOAD COPY information.
    1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
```

```
Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15

Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Member member2 wiederhergestellt werden kann.

10. Stellen Sie die Datenbank zample auf dem TSM-Server vom Member member2 aus wieder her, indem Sie folgenden Befehl verwenden:

```
member2:/home/regress9> db2 restore db zample use tsm options
   '-asnodename=clusternode' without prompting
```

Die folgenden Informationen werden zurückgegeben:

```
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Anmerkung: Wenn die Datenbank zample auf member2 bereits vorhanden wäre, würde der Parameter **OPTIONS** weggelassen, und der Datenbankkonfigurationsparameter **vendoropt** würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter **OPTIONS** für eine BACKUP- oder RESTORE-Operation.

11. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Recovery die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter **vendoropt** festlegen:

```
member2:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

Anmerkung: In DB2 pureScale-Umgebungen können Sie die globalen **vendoropt**-Datenbankkonfigurationsparameter einmal (1) von einem beliebigen Member aus festlegen.

12. Mit dem folgenden Befehl können Sie die memberübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

```
member2:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                                Status der aktualisierenden Recovery

    Aliasname der Eingabedatenbank             = zample
Anzahl der Member, die Status zurückgaben = 3


Mitgldsnr.     Aktualis.    Nächstfolg.  Verarbeitete              Zuletzt festgeschriebene
               Wiederherst. Protokoll    Protokolldateien          Transaktion
               Status       zum Lesen
-------------  -----------  -----------  ------------------------  ------------------------------
           0  nicht anstehend           S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
           1  nicht anstehend           S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
           2  nicht anstehend           S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC

  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Member member2 unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf Member member1 unter dem Benutzer roecken wiederhergestellt.

Synchronisieren der Systemzeit in einer Umgebung mit partitionierten Datenbanken

Unter den Datenbankpartitionsservern sollte für eine relative hohe Synchronisation der Systemuhren gesorgt werden, um einen reibungslosen Datenbankbetrieb und eine uneingeschränkte Möglichkeit zur aktualisierenden Recovery zu gewährleisten.

Zeitunterschiede zwischen den Datenbankpartitionsservern, einschließlich aller potenziellen betriebs- und kommunikationsbedingten Verzögerungen für eine Transaktion, sollten kleiner sein als der für den Konfigurationsparameter *max_time_diff* des Datenbankmanagers angegebene Wert, mit dem die maximale Zeitdifferenz zwischen Knoten definiert wird.

Um sicherzustellen, dass die Zeitmarken der Protokollsätze die Reihenfolge von Transaktionen in einer Umgebung mit partitionierten Datenbanken richtig wiedergeben, verwendet DB2 die Systemuhr und die in der Datei SQLOGCTL.LFH gespeicherte virtuelle Zeitmarke der jeweiligen Maschine als Basis für die Zeitmarken in den Protokollsätzen. Wenn die Systemuhr jedoch vorgestellt wird, wird die Protokolluhr automatisch mit vorgestellt. Obwohl die Systemuhr wieder zurückgestellt werden kann, kann die Uhr für die Protokolle dies nicht und bleibt auf dieser *vorgestellten* Zeit, bis die Systemuhr mit dieser Zeit übereinstimmt. Dann sind die Uhren synchron. Dies hat zur Konsequenz, dass ein kurzfristiger Systemuhrfehler auf einem Datenbankknoten einen langfristigen Effekt auf die Zeitmarken von Datenbankprotokollen haben kann.

Nehmen Sie zum Beispiel an, dass die Systemuhr auf Datenbankpartitionsserver A irrtümlicherweise auf den 7. November 2005 gesetzt wird, obwohl das Jahr 2003 ist, und nehmen Sie weiter an, dass der Fehler korrigiert wurde, *nachdem* eine Transaktion zur Aktualisierung in der Datenbankpartition auf diesem Datenbankpartitionsserver festgeschrieben wurde. Wenn die Datenbank ständig verwendet und regelmäßig aktualisiert wird, bleibt jeder Zeitpunkt zwischen dem 7. November 2003 und dem 7. November 2005 durch die aktualisierende Recovery praktisch unerreichbar. Wenn die COMMIT-Operation auf Datenbankpartitionsserver A erfolgt ist, wird die Zeitmarke im Datenbankprotokoll auf 2005 gesetzt und die Uhr des Protokolls bleibt auf dem 7. November 2005 stehen, bis die Systemuhr diesen Zeitpunkt erreicht. Wenn Sie versuchen, eine aktualisierende Recovery bis zu einem Zeitpunkt innerhalb dieses Zeitrahmens durchzuführen, stoppt die Operation an der ersten Zeitmarke, die über den angegebenen Stoppzeitpunkt, d. h. den 7. November 2003, hinausgeht.

Zwar kann DB2 etwaige Aktualisierungen der Systemuhr nicht steuern, der Konfigurationsparameter des Datenbankmanagers *max_time_diff* kann jedoch die Wahrscheinlichkeit reduzieren, dass diesbezüglich Probleme auftreten:

- Die konfigurierbaren Werte für diesen Parameter reichen von 1 Minute bis zu 24 Stunden.
- Wenn die erste Verbindungsanforderung an eine Nichtkatalogpartition erfolgt, sendet der Datenbankpartitionsserver seine Zeit an die Katalogpartition für die Datenbank. Die Katalogpartition überprüft, ob die Zeit in der Datenbankpartition, die die Verbindung anfordert, und ihre eigene Zeit innerhalb des durch den Parameter *max_time_diff* definierten Bereichs liegen. Falls dieser Bereich überschritten wird, wird die Verbindung verweigert.

- Eine Aktualisierungstransaktion, die auf mehr als zwei Datenbankpartitionsserver in der Datenbank zugreift, muss überprüfen, ob die Systemuhren auf den beteiligten Datenbankpartitionsservern synchron sind, bevor die Aktualisierung festgeschrieben werden kann. Wenn zwei oder mehr Datenbankpartitionsserver eine Zeitdifferenz aufweisen, die den durch den Parameter *max_time_diff* gesetzten Grenzwert überschreitet, wird die Transaktion rückgängig gemacht, um zu verhindern, dass die falsche Zeit auf weitere Datenbankpartitionsserver übertragen wird.

Fehlerbehebung

Fehlerbehebung in Umgebungen mit partitionierten Datenbanken

Absetzen von Befehlen in Umgebungen mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken kann es wünschenswert sein, Befehle absetzen zu können, die auf Computern in der Instanz oder auf Datenbankpartitionsservern ausgeführt werden. Sie haben diese Möglichkeit mithilfe des Befehls **rah** oder des Befehls **db2_all**. Der Befehl **rah** ermöglicht Ihnen das Absetzen von Befehlen, die Sie auf Computern in der Instanz ausführen wollen.

Wenn Sie die Befehle auf Datenbankpartitionsservern in der Instanz ausführen wollen, verwenden Sie den Befehl **db2_all**. Dieser Abschnitt enthält eine Übersicht über diese Befehle. Die folgenden Informationen beziehen sich ausschließlich auf Umgebungen mit partitionierten Datenbanken.

Unter Windows müssen Sie mit einem Benutzerkonto angemeldet sein, das zur Administratorgruppe gehört, um die Befehle **rah** oder **db2_all** ausführen zu können.

Auf Linux- und UNIX-Betriebssystemen kann Ihre Anmeldeshell eine Korn-Shell oder eine beliebige andere Shell sein. Allerdings gibt es Unterschiede in der Art, wie verschiedene Shells Befehle behandeln, die Sonderzeichen enthalten.

Darüber hinaus verwendet der Befehl **rah** auf Linux- und UNIX-Betriebssystemen das Programm für die ferne Shell, das durch die Registrierdatenbankvariable **DB2RSHCMD** angegeben wird. Sie können zwischen den beiden Programmen für die ferne Shell wählen: 'ssh' (für zusätzliche Sicherheit) oder 'rsh' (bzw. 'remsh' für HP-UX). Wenn **DB2RSHCMD** nicht definiert ist, wird 'rsh' (bzw. 'remsh' für HP-UX) verwendet. Das Programm 'ssh' für die ferne Shell wird verwendet, um die Übertragung von unverschlüsselten Kennwörtern in UNIX-Betriebssystemumgebungen zu verhindern.

Wenn ein Befehl nur auf einem Datenbankpartitionsserver ausgeführt wird und Sie ihn auf allen von ihnen ausführen wollen, verwenden Sie dazu den Befehl **db2_all**. Eine Ausnahme bildet der Befehl **db2trc**, der auf allen logischen Datenbankpartitionsservern auf einem Computer ausgeführt wird. Wenn Sie den Befehl **db2trc** auf allen logischen Datenbankpartitionsservern aller Computer ausführen möchten, verwenden Sie dazu den Befehl **rah**.

Anmerkung: Der Befehl **db2_all** unterstützt keine Befehle, die eine interaktive Benutzereingabe erfordern.

Teil 4. Leistungsaspekte

Kapitel 13. Leistungsaspekte beim Datenbankentwurf

Leistungsverbessernde Funktionalität

Tabellenpartitionierung und MDC-Tabellen

In einer Tabelle, die sowohl MDC- als auch Datenpartitionstabelle ist, können Spalten in der Bereichspartitionsspezifikation (Range-Partition-Spec) und auch im MDC-Schlüssel für die Tabellenpartitionierung verwendet werden. Eine Tabelle, die diese beiden Merkmale aufweist, ermöglicht eine exaktere Differenzierung des Datenpartitions- und Blockausschlusses, als dies bei Verwendung nur einer dieser Funktionalitäten der Fall wäre.

Es gibt auch zahlreiche Anwendungen, in denen es sinnvoll ist, für den MDC-Schlüssel andere Spalten als die für die Tabellenpartitionierung genutzten Spalten anzugeben. Hierbei ist zu beachten, dass die Tabellenpartitionierung mit mehreren Spalten arbeitet, während beim MDC mit mehreren Dimensionen gearbeitet wird.

Merkmale eines normalen DB2 Data Warehouse

Die folgenden Empfehlungen beziehen sich auf typische, normale Data Warehouses, die in DB2 Version 9.1 neu implementiert wurden. Die folgenden Merkmale werden angenommen:

- Die Datenbank arbeitet auf mehreren Systemen oder in mehreren logischen AIX-Partitionen.
- Umgebungen mit partitionierten Datenbanken werden verwendet (Tabellen werden mithilfe der Klausel DISTRIBUTE BY HASH erstellt).
- Es wurden zwischen vier und 50 Datenpartitionen definiert.
- Die Tabelle, für die die Möglichkeit zur Verwendung des MDC und der Tabellenpartitionierung geprüft wird, ist eine der zentralen Fakttabellen.
- Die Tabelle umfasst zwischen 100.000.000 und 100.000.000.000 Zeilen.
- Neue Daten werden in unterschiedlichen Zeitrahmen geladen: Jeweils über Nacht, wöchentlich, monatlich.
- Das tägliche Aufnahmevolumen liegt zwischen 10.000 und 10 Millionen Datensätzen.
- Die Datenvolumen schwanken: Hierbei liegt das Volumen des Spitzenmonats um das Fünffache höher als das des Monats mit dem geringsten Datenaufkommen. Entsprechend beläuft sich auch der Umfang der größten Dimensionen (Produktlinie, Bereich) auf das Fünffache des Umfangs der kleinsten Dimensionen.
- Die Datenbank enthält detaillierte Datenbestände der letzten 1 - 5 Jahre.
- Abgelaufene Daten werden in monatlichen oder vierteljährlichen Abständen mit einer Rollout-Operation ausgelagert.
- Tabellen verwenden eine Vielzahl von Abfragetypen. Die Workload besteht jedoch größtenteils aus analytischen Abfragen, die in Bezug auf OLTP-Workloads die folgenden Merkmale aufweisen:
  - Es gibt umfangreichere Ergebnismengen mit bis zu 2 Millionen Zeilen.
  - Die Mehrzahl oder alle Abfragen beziehen sich auf Sichten und nicht auf Basistabellen.

- SQL-Klauseln zur Auswahl von Daten nach Bereichen (Klausel BETWEEN), Elementen in Listen etc.

Merkmale einer Fakttabelle eines normalen DB2 Data Warehouse der Version 9.1

Eine normale Data Warehouse-Fakttabelle kann z. B. das folgende Design verwenden:

- Erstellung von Datenpartitionen über die Spalte 'Month'.
- Definition einer Datenpartition für jeden Rollout-Zeitraum, z. B. 1 Monat, 3 Monate.
- Erstellung von MDC-Dimensionen für 'Day' und für 1 - 4 weitere Dimensionen. Typische Dimensionen sind: Produktlinie und Bereich.
- Alle Datenpartitionen und MDC-Cluster sind über alle Datenpartitionen verteilt.

Das MDC und die Tabellenpartitionierung bieten teilweise dieselben Vorteile. Die folgende Tabelle enthält mögliche Anforderungen innerhalb Ihres Unternehmens und Empfehlungen zu einem Organisationsschema, die auf der Basis der zuvor festgestellten Merkmale gegeben werden.

Tabelle 15. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Empfehlung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Tabellenpartitionierung und MDC	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines traditionellen Offlinezeitfensters	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines Mikro-Offlinezeitfensters (weniger als 1 Minute)	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.

Tabelle 15. Verwendung der Tabellenpartitionierung mit MDC-Tabellen (Forts.)

Problemstellung	Empfohlenes Schema	Empfehlung
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums bei gleichzeitiger Erhaltung der Tabellenverfügbarkeit für Geschäftsbenutzer, die Abfragen ohne Serviceverluste weiterhin übergeben können	MDC	MDC kann nicht alle in diesem Zusammenhang geltenden Anforderungen voll erfüllen. Die Tabellenpartitionierung eignet sich hier nicht, da die Tabelle nur für kurze Zeit in den Offlinemodus versetzt wird.
Tägliches Laden von Daten (Befehl **LOAD** oder **INGEST**)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Kontinuierliches Laden von Daten (Befehl **LOAD** mit ALLOW READ ACCESS oder Befehl **INGEST**)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Leistung bei der Ausführung von Abfragen für traditionelle BI-Abfragen	Tabellenpartitionierung und MDC	MDC eignet sich besonders gut für Abfragen in Datenkuben und mehreren Dimensionen. Die Tabellenpartitionierung bietet Unterstützung über den Partitionsausschluss.
Minimierung des Reorganisationsaufwandes durch Vermeidung des Reorganisationsbedarfs oder Reduzierung des Aufwands für die Ausführung der Task	MDC	Beim MDC wird das Clustering beibehalten, sodass sich die Notwendigkeit von Reorganisationen verringert. Wenn das MDC verwendet wird, dann bietet die Datenpartitionierung auch in Teilbereichen keine Vorteile. Allerdings ermöglicht die Tabellenpartitionierung bei Nichtverwendung des MDC die Reduzierung des Reorganisationsbedarfs, indem auf Partitionsebene ein etwas groberes Clustering beibehalten wird.

Beispiel 1:

Sie arbeiten mit einer Tabelle mit den Schlüsselspalten 'YearAndMonth' und 'Province'. Bei der Planung dieser Tabelle wäre eine Partitionierung nach Datum mit einer Zeitspanne von zwei Monaten pro Datenpartition sinnvoll. Darüber hinaus können Sie die Daten auch nach der Spalte 'Province' organisieren, sodass alle Zeilen für ein bestimmtes Gebiet innerhalb eines Datumsbereichs von zwei Monaten in einer Gruppe zusammengefasst werden. Diese Vorgehensweise ist in Abb. 6 auf Seite 37 dargestellt.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

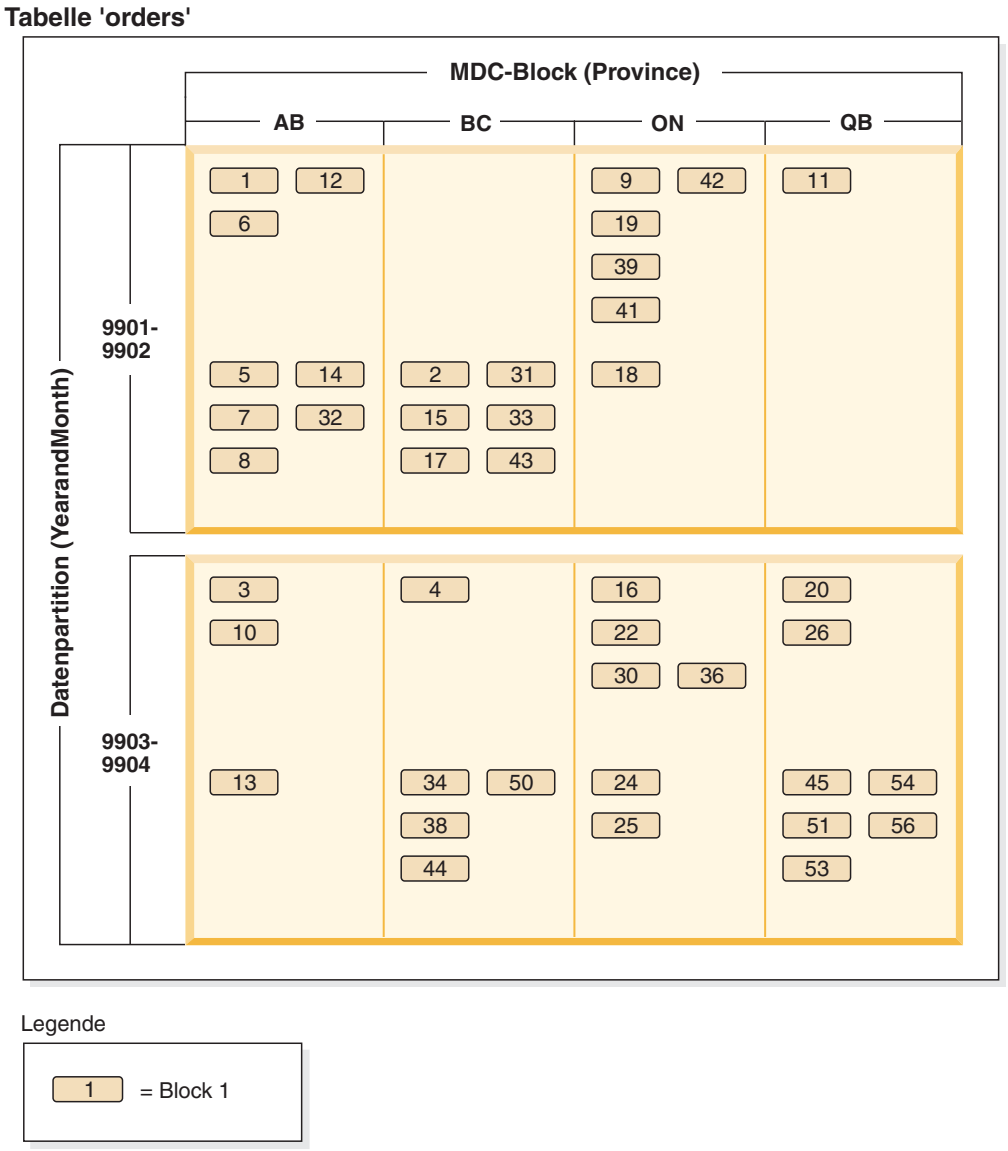


Abbildung 44. Eine nach 'YearAndMonth' partitionierte und nach 'Province' organisierte Tabelle

Beispiel 2:

Eine exaktere Differenzierung kann durch Hinzufügen von 'YearAndMonth' zur Klausel ORGANIZE BY DIMENSIONS (siehe hierzu Abb. 7 auf Seite 38) erzielt werden.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

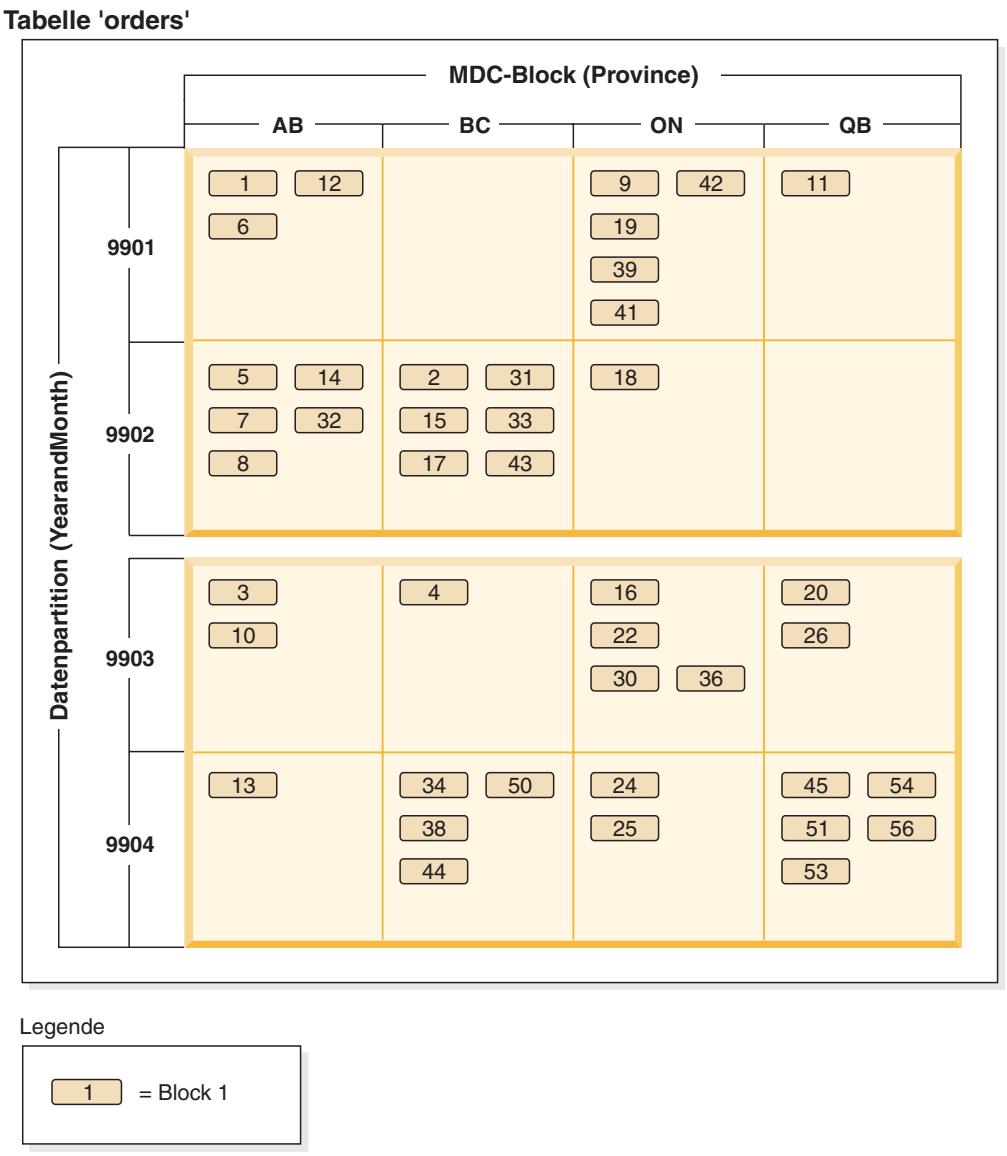


Abbildung 45. Eine nach 'YearAndMonth' partitionierte und nach 'Province' und 'YearAndMonth' organisierte Tabelle

Wenn die Partitionierung so festgelegt wurde, dass jeder Bereich nur einen einzigen Wert enthält, ergeben sich keine Vorteile, wenn die Tabellenpartitionierungsspalte in den MDC-Schlüssel aufgenommen wird.

Wichtige Hinweise

- Im Vergleich mit einer Basistabelle benötigen sowohl MDC-Tabellen als auch partitionierte Tabellen mehr Speicherplatz. Diese Speicherplatzanforderungen gelten zusätzlich zu den sonstigen Anforderungen, sind jedoch unter Berücksichtigung der sich daraus ergebenden Vorteile sinnvoll.
- Wenn Sie die Tabellenpartitionierung und die MDC-Funktionalität in Ihrer partitionierten Datenbankumgebung nicht zusammen einsetzen wollen, dann sollten Sie die Tabellenpartitionierung in den Fällen einsetzen, in denen die Datenverteilung verlässlich vorausgesagt werden kann. Dies ist normalerweise bei den hier erläuterten Systemtypen der Fall. Andernfalls sollten Sie die Verwendung von MDC in Betracht ziehen.

- Bei einer MDC-Datenpartitionstabelle, die mit DB2 Version 9.7 Fixpack 1 (oder einem neueren Release) erstellt wurde, sind die MDC-Blockindizes für die Tabelle partitioniert. Bei einer MDC-Datenpartitionstabelle, die mit DB2 V9.7 (oder einem früheren Release) erstellt wurde, sind die MDC-Blockindizes für die Tabelle nicht partitioniert.

Optimierungsstrategien für partitionierte Tabellen

Die Bezeichnung *Ausschluss von Datenpartitionen* bezieht sich auf die Fähigkeit des Datenbankservers, auf der Grundlage von Abfragevergleichselementen 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 Tabelle werden in mehrere Speicherobjekte auf der Basis von Spezifikationen partitioniert, die in der Klausel PARTITION BY der Anweisung CREATE TABLE angegeben werden. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, im selben Tabellenbereich oder in einer Kombination solcher Tabellenbereiche befinden.

Das folgende Beispiel veranschaulicht die Leistungsvorteile des Ausschlusses von Datenpartitionen.

```
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 nur für Kundeninformationen des Jahres 2000.

```
select * from custlist
  where subsdate between '1/1/2000' and '12/31/2000'
```

Wie Abb. 46 auf Seite 333 zeigt, stellt der Datenbankserver fest, dass nur auf eine Datenpartition in Tabellenbereich TS4 zugegriffen werden muss, um diese Abfrage zu erfüllen.

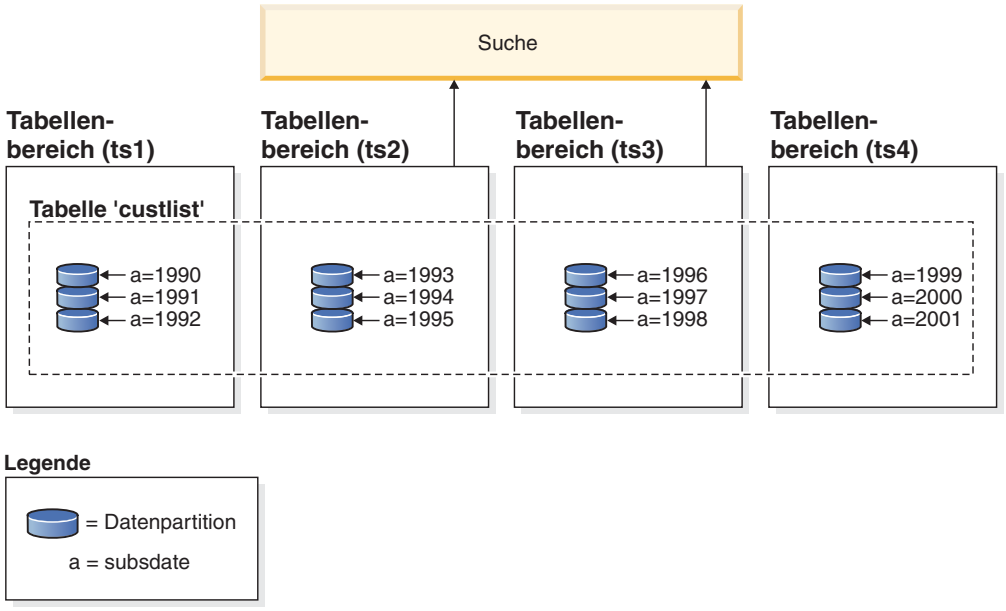


Abbildung 46. Die Leistungsvorteile des Datenpartitionsausschlusses

Ein weiteres Beispiel für den Ausschluss von Datenpartitionen basiert auf folgendem Schema:

```
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)
```

Nehmen Sie an, dass Sie die folgende Abfrage ausführen:

```
select * from multi
  where sale_date between '6/1/2005'
    and '7/31/2005' and region = 'NW'
```

Ohne Tabellenpartitionierung besteht ein wahrscheinlicher Plan in der logischen Verknüpfung der Indizes über AND. Beim logischen Verknüpfen von Indizes über AND (Index ANDing) werden die folgenden Aktionen ausgeführt:

- 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. 47 auf Seite 334 dargestellt, wird bei Tabellenpartitionierung der Index gelesen, um Übereinstimmungen für beide Spalten, d. h. REGION und SALE_DATE, zu ermitteln, sodass entsprechende Zeilen schnell abgerufen werden können.

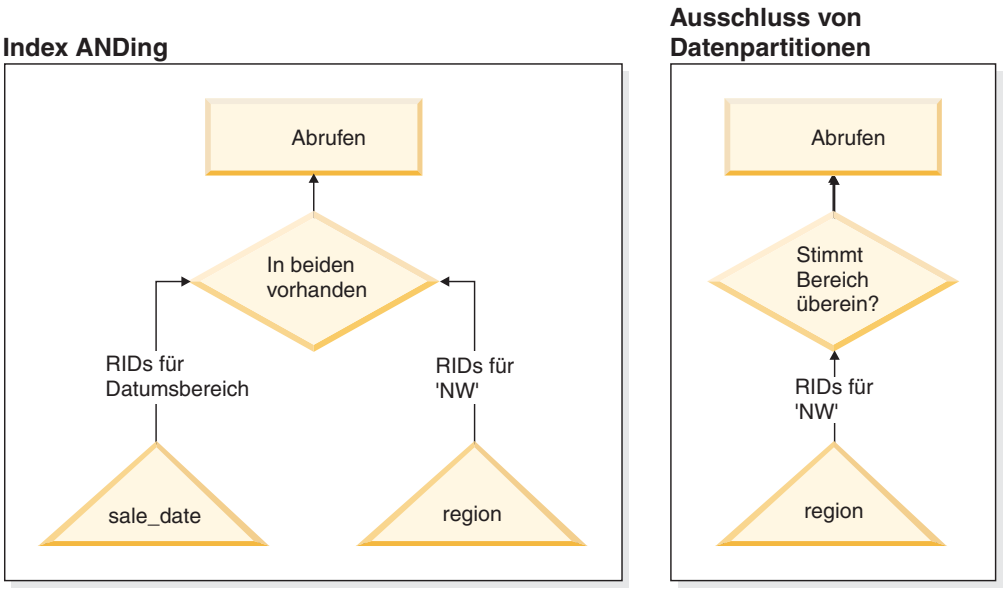


Abbildung 47. Der Entscheidungspfad des Optimierungsprogramms für die Tabellenpartitionierung und das logische Verknüpfen von Indizes über AND (Index ANDing)

DB2-Explain

Sie können auch die EXPLAIN-Funktion verwenden, um den Plan mit Datenpartitionsausschluss zu ermitteln, der vom Abfrageoptimierungsprogramm 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. Beispiel:

```
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)

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 bei Datenpartitionen, die mehrere Bereiche haben (d. h. Bereiche, die durch logisches OR verknüpft werden), zu erzielen. An der Tabelle SALES, die im vorigen Beispiel erstellt wurde, 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. Beispiel:

```
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, der zur Generierung der

Spalten verwendet wird, 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. Beispiel:

```
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)
```

Die folgenden beiden Vergleichselemente werden verwendet:

```
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 im Tabellenbereich TS6 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

Äußere Tabelle T2: Spalte A	Innere Tabelle T1: Spalte A	Innere Tabelle T1: Spalte B	Innere Tabelle T1: Position der Datenpartition
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 Tabellenbereich TS4.
4. Nach Anwendung von T1.A = 2 und T1.B > 15 durchsucht er die Datenpartitionen im Tabellenbereich TS4 von Tabelle T1, bis eine Zeile gefunden wird. Die erste qualifizierte Zeile, die gefunden wird, ist die dritte Zeile von T1.
5. Er verknüpft die übereinstimmenden Zeilen (Join).
6. Er durchsucht die Datenpartitionen in Tabellenbereich TS4 von Tabelle T1, 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 (wobei er den Wert 3 aus Spalte A nimmt). Dieses Verfahren wird fortgesetzt, bis alle Zeilen von T2 verarbeitet wurden.

Indizes zu XML-Daten

Ab DB2 Version 9.7 Fixpack 1 können Sie einen Index zu XML-Daten in einer partitionierten Tabelle als partitioniert oder nicht partitioniert erstellen. Standardmäßig wird ein partitionierter Index erstellt.

Partitionierte und nicht partitionierte XML-Indizes werden vom Datenbankmanager bei Einfüge-, Aktualisierungs- und Löschoperationen für Tabellen auf die gleiche Weise wie andere relationale Indizes für eine partitionierte Tabelle verwaltet. Nicht partitionierte Indizes zu XML-Daten für eine partitionierte Tabelle werden auf die gleiche Weise wie Indizes zu XML-Daten für eine nicht partitionierte Tabelle verwendet, um die Abfrageverarbeitung zu beschleunigen. Mithilfe des Abfragevergleichselements könnte zum Beispiel ermittelt werden, dass nur auf eine Untergruppe der Datenpartitionen in der partitionierten Tabelle zugegriffen werden muss, um die Abfrage zu erfüllen.

Der Ausschluss von Datenpartitionen und Indizes zu XML-Spalten können kombiniert werden, um die Abfrageleistung zu verbessern. Betrachten Sie die folgende partitionierte Tabelle:

```
create table employee (a int, b xml, c xml)
  index in tbspx
  partition by (a) (
    starting 0 ending 10,
    ending 20,
    ending 30,
    ending 40)
```

Betrachten Sie nun die folgende Abfrage:

```
select * from employee
  where a > 21
  and xmlexist('$doc/Person/Name/First[.="Eric"]'
    passing "EMPLOYEE"."B" as "doc")
```

Das Optimierungsprogramm kann die ersten beiden Partitionen aufgrund des Vergleichselements `a > 21` sofort ausschließen. Wenn der nicht partitionierte Index zu XML-Daten für Spalte B vom Optimierungsprogramm im Abfrageplan ausgewählt wird, kann eine Indexsuche im Index zu XML-Daten das Ergebnis des Datenpartitionsausschlusses des Optimierungsprogramms nutzen und nur Ergebnisse zurückgeben, die zu Partitionen gehören, die nicht durch die relationalen Vergleichselemente zum Datenpartitionsausschluss ausgeschlossen wurden.

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 beruhen in erster Linie auf der verbesserten Effizienz von Blockindizes. Jedoch bietet das Clustering in mehreren Dimensionen auch den Vorteil eines schnelleren Datenabrufs.

Optimierungsstrategien für MDC-Tabellen können auch die Leistungsvorteile der partitionsinternen und partitionsübergreifenden Parallelität ausnutzen. MDC-Tabellen bieten die folgenden besonderen Vorteile:

- Dimensionsblockindexsuchen können die erforderlichen Teile der Tabelle ermitteln und schnell nur die angeforderten Blöcke durchsuchen.
- Da Blockindizes kleiner als Satz-ID-Indizes (RID-Indizes) sind, arbeiten Blockindexsuchen schneller.
- Logische Verknüpfungen von Indizes über AND und OR (Index ANDing und Index ORing) können auf Blockebene durchgeführt und mit Satz-IDs kombiniert werden.
- Daten werden garantiert in EXTENTSIZE großen Speicherbereichen in Clustern gruppiert, was ein schnelleres Abrufen ermöglicht.
- Zeilen können schneller gelöscht werden, wenn ein Rollout (Datenauslagerung) 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 diese Blöcke durchsuchen, um die Ergebnismenge schnell abzurufen.

Rolloutlöschung

Wenn Bedingungen das Löschen durch einen Rollout zulassen, wird dieses effizientere Verfahren zum Löschen von Zeilen aus MDC-Tabellen verwendet. Die folgenden Bedingungen müssen erfüllt sein:

- Bei der DELETE-Anweisung handelt es sich um eine DELETE-Anweisung mit Suche, nicht um eine positionierte DELETE-Anweisung (die Anweisung verwendet keine Klausel WHERE CURRENT OF).

- Es gibt keine WHERE-Klausel (alle Zeilen sind zu löschen) oder die einzigen Bedingungen in der WHERE-Klausel gelten für 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 sind keine ON DELETE-Trigger definiert.
- Die Tabelle wird in keinen MQTs (Materialized Query Tables) verwendet, die sofort aktualisiert werden.
- Eine kaskadierende Löschoperation kommt für einen Rollout infrage, 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 Neuformatierung 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 angegeben 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 größer ist die Verbesserung (als Prozentsatz von der Gesamtzeit und vom gesamten Protokollspeicher).

Ein Schätzwert für den eingesparten Platz im Protokoll kann anhand der folgenden Formel ermittelt werden:

```
S + 38*N - 50*P
```

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. Dieser Wert stellt die Verkleinerung in den tatsächlichen Protokolldaten dar. Die Einsparungen an erforderlichem aktiven Protokollspeicher betragen das Doppelte dieses Werts, da auch der Speicherbereich, der für ein Rollback reserviert war, eingespart wird.

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_ROLLOUT** auf den Wert DEFER oder durch Angeben von DEFERRED in der Anweisung SET CURRENT MDC ROLLOUT MODE angegeben 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 bei einer Tabelle mit mehreren Satz-ID-Indizes führen. Die Geschwindigkeit der gesamten 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 der Datenbankmanager die erforderlichen Speicherstrukturen nicht zuordnen kann, schlägt der Rollout mit verzögerter Bereinigung fehl und eine Nachricht wird in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben.

Empfehlungen für die Verwendung eines Rollouts mit verzögerter Bereinigung

Wenn 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 folgende Aspekte bei der Entscheidung zwischen sofortiger und verzögerter Indexbereinigung berücksichtigt werden:

- Umfang der Löschoperation

  Wählen Sie einen Rollout mit verzögerter Bereinigung für sehr umfangreiche Löschoperationen aus. In Fällen, in denen DELETE-Anweisungen für Dimensionen häufig in vielen kleinen MDC-Tabellen ausgeführt werden, kann der Aufwand für die asynchrone Bereinigung von Indexobjekten den Vorteil der Zeiteinsparung während der Löschoperationen übersteigen.
- Anzahl und Typ von Indizes

  Wenn die Tabelle eine Reihe von Satz-ID-Indizes hat, die eine Verarbeitung auf Zeilenebene erfordern, sollte ein Rollout mit verzögerter Bereinigung verwendet werden.
- Blockverfügbarkeit

  Wenn Sie wünschen, dass der Blockspeicherplatz durch die Löschoperation freigegeben wird, sodass er sofort nach dem Festschreiben der DELETE-Anweisung verfügbar ist, verwenden Sie einen Rollout mit sofortiger Bereinigung.
- 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 bei Löschoperationen inaktivieren wollen, setzen Sie die Registrierdatenbankvariable **DB2_MDC_ROLLOUT** auf den Wert OFF oder geben NONE in der Anweisung SET CURRENT MDC ROLLOUT MODE an.

Anmerkung: In DB2 Version 9.7 und späteren Releases wird ein Rollout mit verzögerter Bereinigung für eine datenpartitionierte MDC-Tabelle mit partitionierten Satz-ID-Indizes nicht unterstützt. Es werden nur die Modi NONE und IMMEDIATE unterstützt. Der Modus des Rollouts mit Bereinigung ist IMMEDIATE, wenn die Registrierdatenbankvariable **DB2_MDC_ROLLOUT** auf den Wert DEFER gesetzt ist

oder wenn das Sonderregister CURRENT MDC ROLLOUT MODE auf den Wert DEFERRED gesetzt ist, um die Einstellung der Variablen **DB2_MDC_ROLLOUT** zu überschreiben.

Wenn nur nicht partitionierte Satz-ID-Indizes für die MDC-Tabelle vorhanden sind, wird ein Rollout mit verzögerter Indexbereinigung unterstützt.

Kapitel 14. Indizes

Indizes in partitionierten Tabellen

Indexverhalten bei partitionierten Tabellen

Indizes für partitionierte Tabellen verhalten sich ähnlich wie Indizes für nicht partitionierte Tabellen. Indizes für partitionierte Tabellen werden jedoch mit einem anderen Speichermodell gespeichert, je nachdem, ob es sich bei ihnen um partitionierte oder nicht partitionierte Indizes handelt.

Obwohl sich die Indizes für reguläre, nicht partitionierte Tabellen in einem gemeinsam genutzten Indexobjekt befinden, werden *nicht partitionierte Indizes* für partitionierte Tabellen in einem eigenen Indexobjekt und in einem einzelnen Tabellenbereich erstellt, auch wenn die Datenbankpartitionen mehrere Tabellenbereiche umfassen. Sowohl vom Datenbankmanager verwaltete Tabellenbereiche (DMS-Tabellenbereiche) als auch vom System verwaltete Tabellenbereiche (SMS-Tabellenbereiche) unterstützen die Verwendung von Indizes, die sich an einer anderen Position als die Tabellendaten befinden. Jeder nicht partitionierte Index kann in einem eigenen Tabellenbereich, auch in großen Tabellenbereichen (LARGE), gespeichert 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. Alle Tabellenbereiche müssen sich in derselben Datenbankpartitionsgruppe befinden.

Ein *partitionierter Index* verwendet ein Indexorganisationsschema, bei dem Indexdaten entsprechend dem Partitionierungsschema der Tabelle auf mehrere *Indexpartitionen* verteilt werden. Jede Indexpartition bezieht sich ausschließlich auf Tabellenzeilen in der entsprechenden Datenpartition. Alle Indexpartitionen für eine bestimmte Datenpartition befinden sich in demselben Indexobjekt.

Ab DB2 Version 9.7 Fixpack 1 können Indizes zu XML-Daten für XML-Spalten in partitionierten Tabellen partitioniert oder nicht partitioniert sein. Standardmäßig werden partitionierte Indizes erstellt. Vom System generierte Indizes zu XML-Regionen sind immer partitioniert, während vom System generierte Indizes zu Spaltenpfaden immer nicht partitioniert sind. In DB2 V9.7 sind Indizes zu XML-Daten nicht partitioniert.

Nicht partitionierte Indizes zeichnen sich durch folgende Vorteile aus:

- Es besteht die Möglichkeit, verschiedene Tabellenbereichsmerkmale für jeden Index zu definieren (z. B. können verschiedene Seitengrößen zu einer besseren Speicherplatznutzung beitragen).
- Indizes können unabhängig voneinander reorganisiert werden.
- Beim Löschen von Indizes wird eine bessere Leistung erzielt.
- Weniger E/A-Konkurrenzsituationen unterstützen einen effizienteren gemeinsamen Zugriff auf die Indexdaten.
- Wenn einzelne Indizes gelöscht werden, wird der Speicherplatz sofort für das System verfügbar, ohne dass eine Indexreorganisation erforderlich ist.

Partitionierte Indizes zeichnen sich durch folgende Vorteile aus:

- Bei der Ein- und Auslagerung von Daten (Rollin/Rollout) wird eine bessere Leistung erzielt.
- Durch die Partitionierung der Indizes entstehen weniger Konkurrenzsituationen bei Indexseiten.
- Es gibt eine B-Indexbaumstruktur für jede einzelne Indexpartition, die folgende Vorteile bieten kann:
  - Beim Einfügen, Aktualisieren, Löschen und Suchen wird eine bessere Leistung erzielt, da die B-Baumstruktur für eine Indexpartition im Allgemeinen weniger Ebenen enthält als ein Index, der auf alle Daten in der Tabelle verweist.
  - Verbesserung bei Suchleistung und gemeinsamem Zugriff, wenn der Ausschluss von Partitionen aktiviert ist. Obwohl der Partitionsausschluss sowohl beim Durchsuchen von partitionierten als auch von nicht partitionierten Indizes verwendet werden kann, ist er beim Suchen in partitionierten Indizes effektiver, da jede einzelne Indexpartition Schlüssel enthält, die sich ausschließlich auf die zugehörige Datenpartition beziehen. Diese Konfiguration kann dazu führen, dass weniger Schlüssel und weniger Indexseiten durchsucht werden müssen, als es bei einer entsprechenden Abfrage für einen nicht partitionierten Index erforderlich wäre.

Während ein nicht partitionierter Index die Reihenfolge der Indexspalten immer beibehält, kann eine über mehrere Partitionen zu wahrende Reihenfolge bei einem partitionierten Index in bestimmten Szenarien ein Stück weit verloren gehen. Dies ist zum Beispiel der Fall, wenn die Partitionierungsspalten nicht den Indexspalten entsprechen oder wenn ein Zugriff auf mehrere Partitionen erforderlich ist.

Bei der Onlineindexerstellung werden gleichzeitige Lese- und Schreibzugriffe auf die Tabelle zugelassen. Nach dem Erstellen eines Onlineindex werden Änderungen, die während der Indexerstellung an der jeweiligen Tabelle vorgenommen wurden, auf den neuen Index angewendet. Der Schreibzugriff auf die Tabelle wird blockiert, bis die Indexerstellung abgeschlossen ist und die Transaktion festgeschrieben wurde. Bei partitionierten Indizes wird jede Datenpartition *nur* in den Quiescemodus versetzt (sodass nur ein Lesezugriff möglich ist), während an der Datenpartition vorgenommene Änderungen (beim Erstellen der Indexpartition) angewendet werden.

Die Unterstützung für partitionierte Indizes ist insbesondere dann hilfreich, wenn Daten mit der Anweisung ALTER TABLE...ATTACH PARTITION eingelagert werden (Rollin). Führen Sie nach dem Zuordnen einer Partition die Anweisung SET INTEGRITY aus, wenn nicht partitionierte Indizes vorhanden sind (ausgenommen XML-Spaltenpfadindizes, wenn die Tabelle XML-Daten enthält). Diese Anweisung ist bei nicht partitionierten Indizes für die Verwaltung, die Bereichsprüfung, die Prüfung von Integritätsbedingungen und die MQT-Verwaltung erforderlich. Die Verwaltung nicht partitionierter Indizes kann sehr aufwendig sein und sehr viel Protokollspeicherplatz belegen. Umgehen Sie diesen Verwaltungsaufwand, indem Sie partitionierte Indizes verwenden.

Wenn nicht partitionierte Indizes (ausgenommen XML-Spaltenpfadindizes) für die Tabelle vorhanden sind, die nach einer ATTACH-Operation verwaltet werden müssen, verhält sich die Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED so, als wäre sie eine Anweisung SET INTEGRITY...IMMEDIATE CHECKED. Die gesamte Integritätsverarbeitung, die Verwaltung nicht partitionierter Indizes und die Tabellenstatusübergänge werden so ausgeführt, als sei eine Anweisung SET INTEGRITY...IMMEDIATE CHECKED ausgegeben worden.

In Abb. 48 sind zwei nicht partitionierte Indizes für eine partitionierte Tabelle zu sehen, wobei sich jeder Index in einem separaten Tabellenbereich befindet.

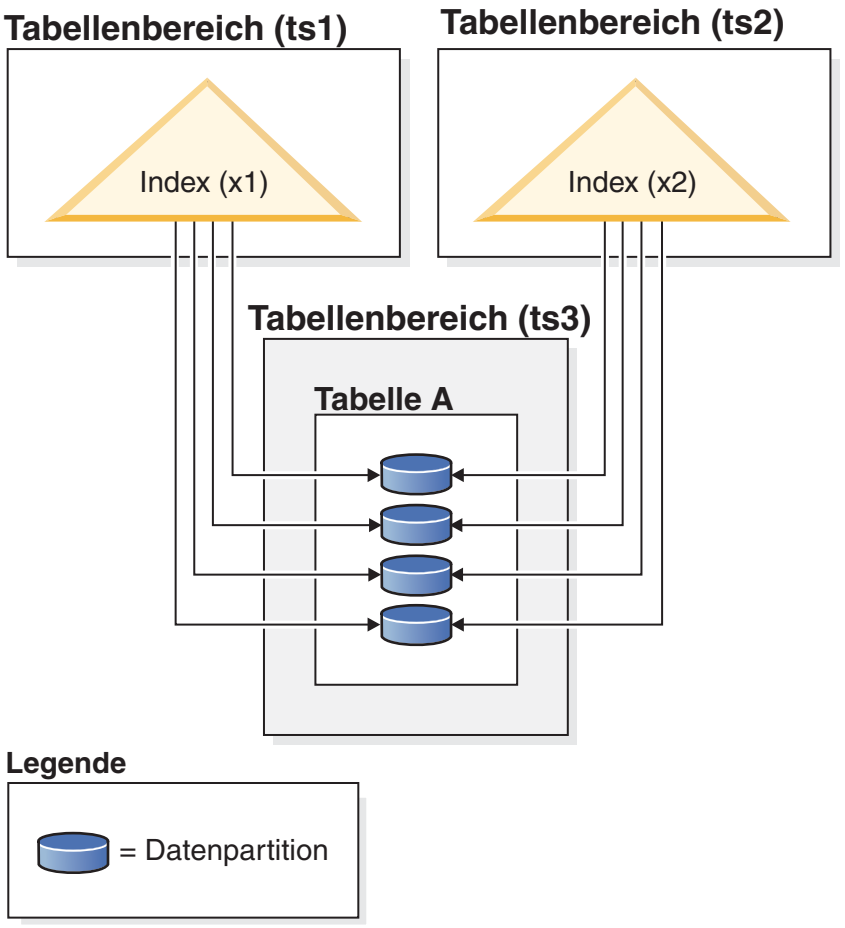


Abbildung 48. Nicht partitionierte Indizes für eine partitionierte Tabelle

Im Diagramm Abb. 49 auf Seite 346 ist ein nicht partitionierter Index für eine partitionierte Tabelle zu sehen, die zwei Datenbankpartitionen umfasst und sich in einem einzigen Tabellenbereich befindet.

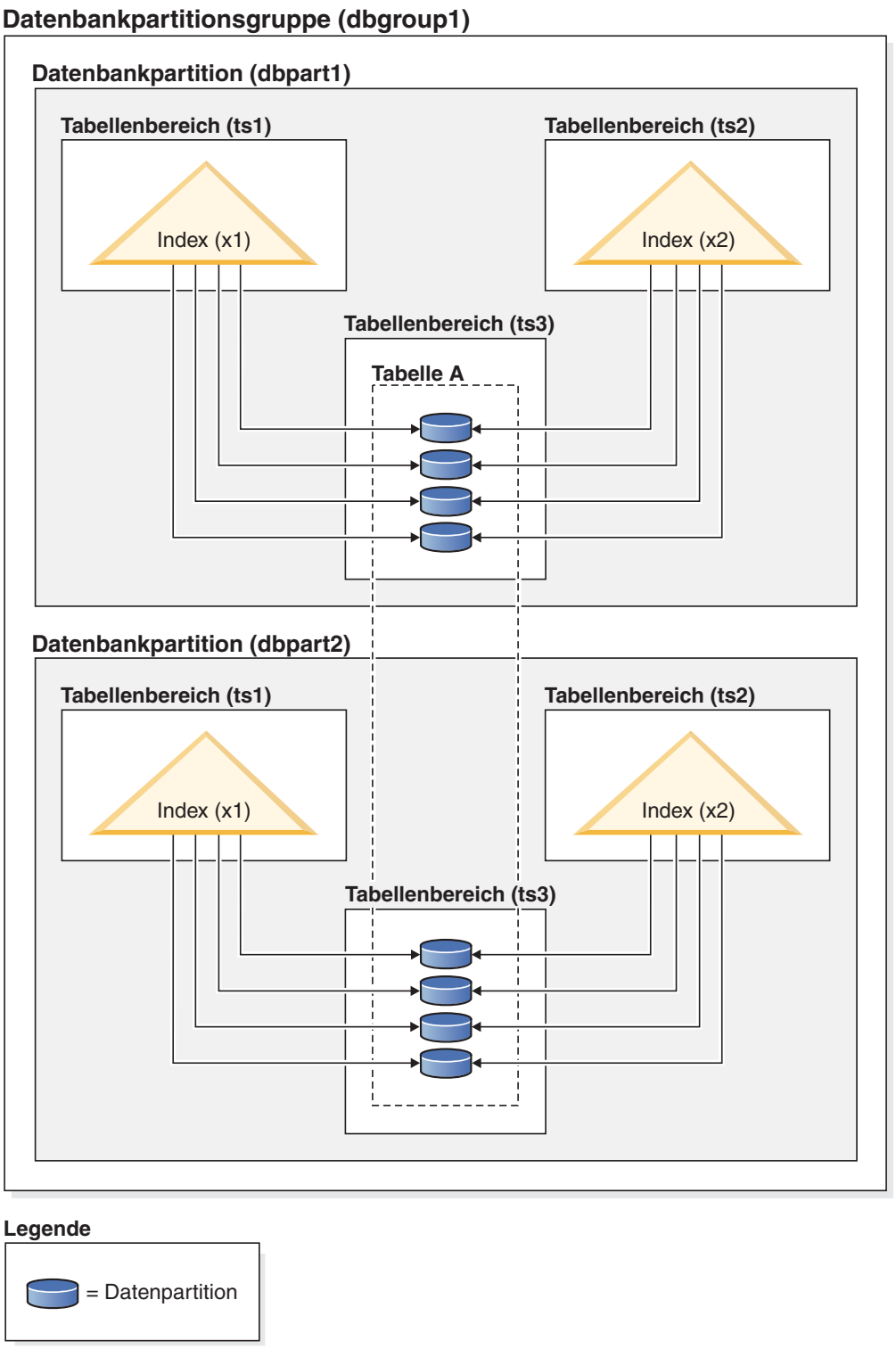


Abbildung 49. Nicht partitionierter Index für eine verteilte und partitionierte Tabelle

Das Diagramm Abb. 50 auf Seite 347 zeigt eine Kombination aus partitionierten und nicht partitionierten Indizes für eine partitionierte Tabelle.

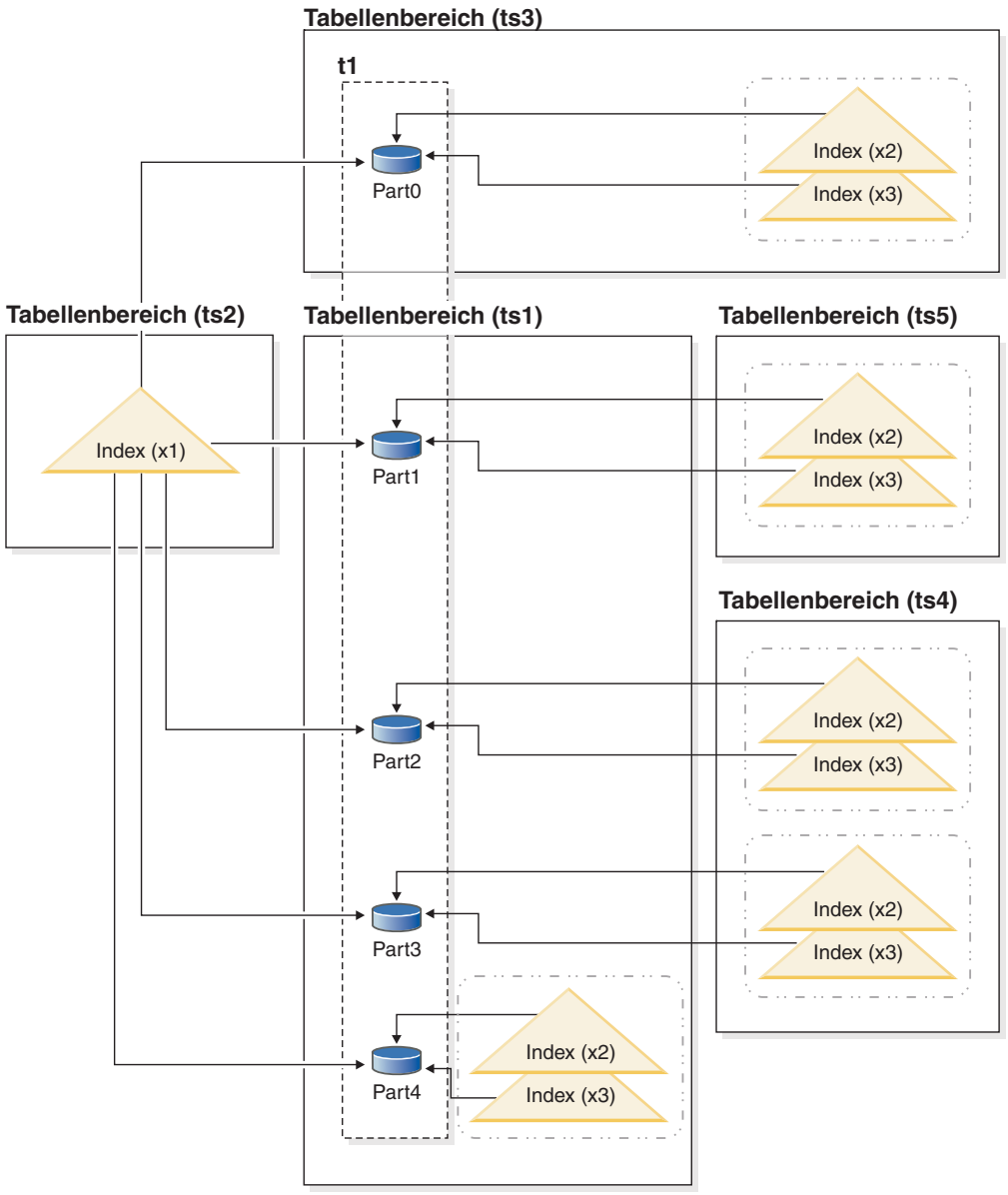


Abbildung 50. Partitionierter Index und nicht partitionierte Indizes für eine partitionierte Tabelle

Der nicht partitionierte Index X1 bezieht sich auf Zeilen in allen Datenpartitionen. Die partitionierten Indizes X2 und X3 beziehen sich dagegen nur auf Zeilen in der Datenpartition, der sie zugeordnet sind. Bei Tabellenbereich TS3 ist ebenfalls zu sehen, dass die Indexpartitionen den Tabellenbereich der Datenpartitionen, denen sie zugeordnet sind, gemeinsam nutzen. Diese Konfiguration ist bei partitionierten Indizes Standard.

Sie können die Standardposition bei nicht partitionierten und bei partitionierten Indizes überschreiben, müssen dabei jedoch unterschiedlich vorgehen. Bei nicht partitionierten Indizes können Sie beim Erstellen des Index einen Tabellenbereich angeben. Bei partitionierten Indizes müssen Sie beim Erstellen der Tabelle festlegen, in welchen Tabellenbereichen Indexpartitionen gespeichert werden.

Nicht partitionierte Indizes

Die Indexposition für nicht partitionierte Indizes können Sie überschreiben, indem Sie die Anweisung CREATE INDEX mit der Klausel IN verwenden und eine alternative Tabellenbereichsposition für den Index angeben. Sie können verschiedene Indizes nach Bedarf in unterschiedlichen Tabellenbereichen speichern. Wenn Sie eine partitionierte Tabelle erstellen, ohne die Speicherposition für die zugehörigen nicht partitionierten Indizes anzugeben, und anschließend einen Index mit einer Anweisung CREATE INDEX erstellen, ohne einen 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. Diese Auswertung zur Bestimmung der Tabellenbereichsposition für den Index wird beendet, wenn ein übereinstimmender Fall gefunden ist.

Fall 1:
Wenn ein Tabellenbereich für den Index in der Anweisung
CREATE INDEX...IN *tabellenbereich*
angegeben ist, wird der angegebene Tabellenbereich für diesen Index verwendet.

Fall 2:
Wenn ein Indextabellenbereich in der Anweisung
CREATE TABLE...INDEX IN *tabellenbereich*
angegeben ist, wird der angegebene Tabellenbereich für diesen Index verwendet.

Fall 3:
Wenn kein Tabellenbereich angegeben ist, wird der Tabellenbereich ausgewählt, der von der ersten zugeordneten oder sichtbaren Datenpartition verwendet wird.

Partitionierte Indizes

Standardmäßig werden Indexpartitionen in demselben Tabellenbereich platziert wie die Datenpartitionen, auf die sie verweisen. Dieses Standardverhalten können Sie überschreiben, indem Sie die Klausel INDEX IN für jede Datenpartition verwenden, die Sie mit der Anweisung CREATE TABLE definieren. Dies bedeutet, dass Sie beim Erstellen einer Tabelle bereits wissen müssen, an welcher Stelle die Indexpartitionen gespeichert werden sollen, wenn Sie partitionierte Indizes für eine partitionierte Tabelle verwenden möchten. Wenn Sie die Klausel INDEX IN beim Erstellen eines partitionierten Index verwenden, erhalten Sie eine Fehlernachricht.

Beispiel 1: Gegeben ist die partitionierte Tabelle SALES (a int, b int, c int). Es wird ein eindeutiger Index A_IDX erstellt.

```
create unique index a_idx on sales (a)
```

Da die Tabelle SALES partitioniert ist, wird der Index A_IDX ebenfalls als partitionierter Index erstellt.

Beispiel 2: Erstellen von Index B_IDX.

```
create index b_idx on sales (b)
```

Beispiel 3: Zum Überschreiben der Standardposition für die Indexpartitionen in einem partitionierten Index verwenden Sie die Klausel INDEX IN für jede Partition, die Sie beim Erstellen der partitionierten Tabelle definieren. Im folgenden Beispiel werden Indizes für die Tabelle Z im Tabellenbereich TS3 erstellt.

```
create table z (a int, b int)
      partition by range (a) (starting from (1)
      ending at (100)  index in ts3)

   create index c_idx on z (a) partitioned
```

Clustering bei nicht partitionierten Indizes für partitionierte 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 einen Clusterindex 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. Während einer Clustereinfügeoperation wird eine Indexsuche durchgeführt, um eine passende Satz-ID (RID) ausfindig zu machen. Diese Satz-ID wird als Ausgangspunkt in der Tabelle für die Suche nach einem Platz zum Einfügen des Datensatzes verwendet. Zur Erzielung einer optimalen 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. Nehmen Sie zum Beispiel an, dass eine Tabelle über die Spalte QUARTER (Quartal) partitioniert ist und ein Clusterindex für die Spalte DATE (Datum) definiert ist. Zwischen den Spalten QUARTER und DATE besteht eine Beziehung und es lässt sich ein optimales Clustering der Daten mit guter Leistung erzielen, weil alle Schlüssel einer Datenpartition im Index zusammen gruppiert werden. Abb. 51 auf Seite 350 veranschaulicht, dass sich eine optimale Suchleistung nur erzielen lässt, wenn das Clustering mit dem Tabellenpartitionierungsschlüssel korreliert.

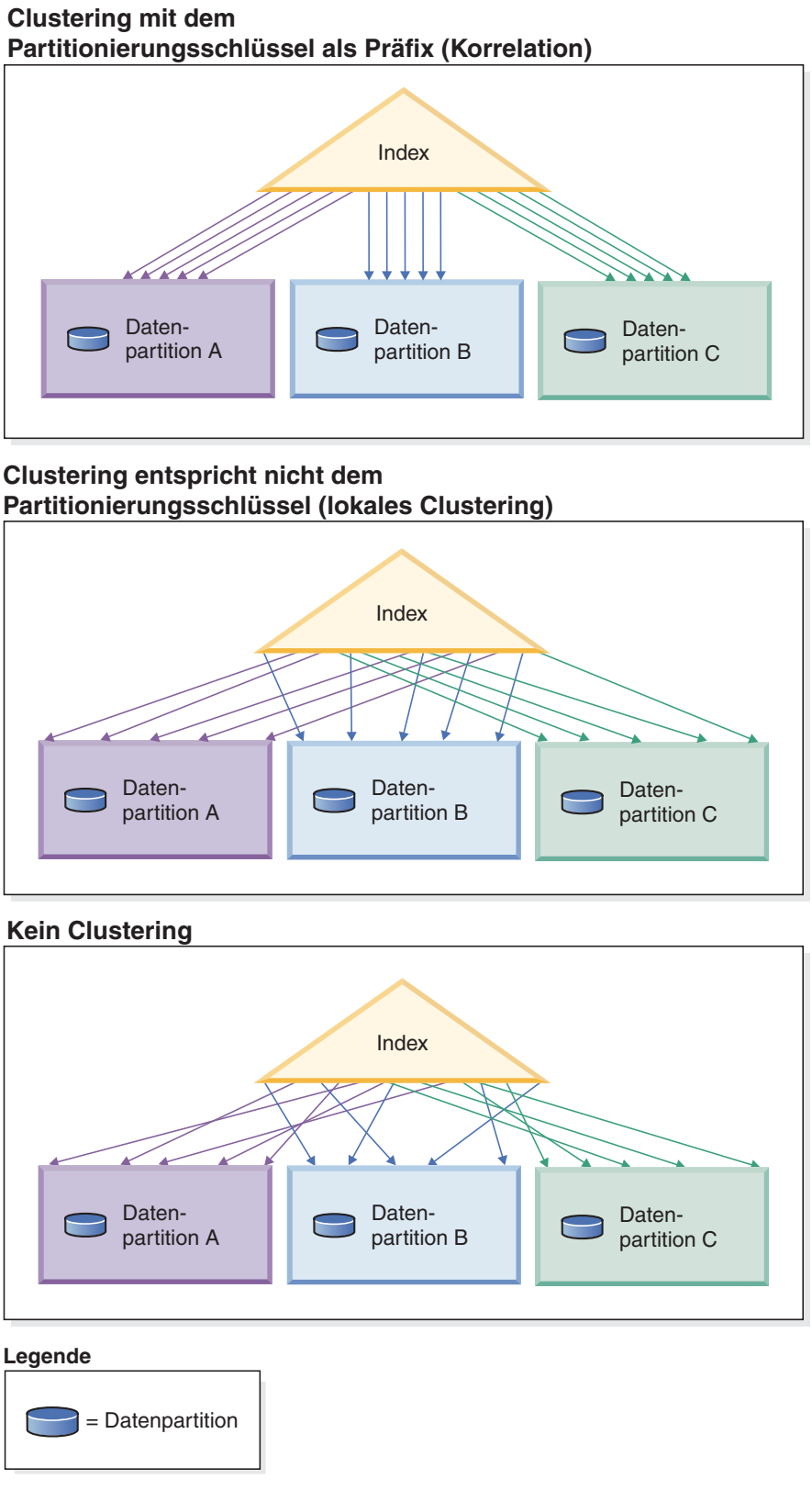


Abbildung 51. Die möglichen Effekte eines Clusterindex für eine partitionierte Tabelle.

Das Clustering bietet folgende Vorteile:

- Die Zeilen sind in jeder Datenpartition in der Reihenfolge des Schlüssels angeordnet.
- Clusterindizes verbessern die Leistung von Suchläufen, welche die Tabelle in der Reihenfolge der Schlüssel durchqueren, weil der Suchvorgang die erste Zeile der ersten Seite abruft, dann jede Zeile auf eben dieser Seite, bevor er mit der nächsten Seite fortfährt. Dies bedeutet, dass sich nur eine Seite der Tabelle zu einem gegebenen Zeitpunkt im Pufferpool befinden muss. Wenn die Tabelle keine Clusterbildung aufweist, werden Zeilen wahrscheinlich von verschiedenen Seiten abgerufen. Wenn der Pufferpool nicht die gesamte Tabelle aufnehmen kann, ist es wahrscheinlich, dass die meisten Seiten mehrmals abgerufen werden und sich die Suche erheblich verlangsamt.

Wenn der Clusterschlüssel jedoch nicht mit dem Tabellenpartitionierungsschlüssel korreliert ist, die Daten jedoch 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. Dies liegt daran, dass jede abgerufene Zeile aus einer bestimmten Datenpartition der Zeile nahe ist, die zuvor aus dieser selben Partition abgerufen wurde (siehe zweites Beispiel in Abb. 51 auf Seite 350).

Kapitel 15. Designadvisor

Verwenden des Designadvisors für die Konvertierung von einer Einzelpartitions- in eine Mehrpartitionsdatenbank

Sie können den Designadvisor zur Unterstützung der Konvertierung einer Datenbank mit einer Einzelpartition in eine Datenbank mit mehreren Partitionen verwenden.

Informationen zu diesem Vorgang

Neben der Ermittlung von Vorschlägen zu neuen Indizes, MQTs (Materialized Query Tables) und Tabellen mit mehrdimensionalem Clustering (MDC-Tabellen) kann der Designadvisor auch Vorschläge zur Verteilung von Daten ausgeben.

Vorgehensweise

1. Mit dem Befehl **db2licm** können Sie den Lizenzschlüssel der Umgebung mit partitionierten Datenbanken registrieren.
2. Erstellen Sie mindestens einen Tabellenbereich in einer Partitionsgruppe einer Datenbank mit mehreren Partitionen.

   Anmerkung: Der Designadvisor kann eine Datenumverteilung nur auf vorhandene Tabellenbereiche empfehlen.
3. Führen Sie den Designadvisor unter Angabe der Partitionierungsoption im Befehl **db2advis** aus.
4. Nehmen Sie an der Ausgabedatei des Befehls **db2advis** geringfügige Änderungen vor, bevor Sie die DDL-Anweisungen (DDL - Data Definition Language, Datendefinitionssprache) ausführen, die vom Designadvisor generiert wurden. Da die Datenbankpartitionierung konfiguriert werden muss, bevor Sie das DDL-Script ausführen können, das vom Designadvisor generiert wird, sind die Vorschläge in dem Script, das zurückgegeben wird, durch Kommentarzeichen inaktiviert. Es bleibt Ihnen überlassen, die Tabellen den Vorschlägen entsprechend umzuwandeln.

Kapitel 16. Verwalten des gemeinsamen Zugriffs

Sperrmodi für MDC/ITC-Tabellensuchen und Satz-ID-Indexsuchen

Der Typ von Sperre, den eine MDC-Tabelle (MDC = Multidimensional Clustering) oder eine ITC-Tabelle (ITC = Insert Time Clustering) bei einer Tabellensuche oder Satz-ID-Indexsuche aktiviert, hängt von der geltenden Isolationsstufe sowie vom verwendeten Datenzugriffsplan ab.

In den folgenden Tabellen sind die Typen von Sperren aufgeführt, die für MDC- und ITC-Tabellen unter der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag hat drei Teile: die Tabellensperre, die Blocksperre und die Zeilensperre. Ein Bindestrich gibt an, dass eine bestimmte Sperrgranularität nicht verfügbar ist.

Die Tabellen 9 - 14 zeigen die Typen von Sperren, die für Satz-ID-Indexsuchen aktiviert werden, wenn das Lesen von Datenseiten verzögert wird. Wenn unter der Isolationsstufe UR Vergleichselemente für INCLUDE-Spalten im Index angewendet werden, wird die Isolationsstufe auf CS erhöht und die Sperren werden auf eine IS-Tabellensperre, eine IS-Blocksperre oder auf NS-Zeilensperren hochgestuft.

- Tabelle 1. Sperrmodi für Tabellensuchen ohne Vergleichselemente
- Tabelle 2. Sperrmodi für Tabellensuchen mit Vergleichselementen nur für Dimensionsspalten
- Tabelle 3. Sperrmodi für Tabellensuchen mit anderen Vergleichselementen (sargs, resids)
- Tabelle 4. Sperrmodi für Satz-ID-Indexsuchen ohne Vergleichselemente
- Tabelle 5. Sperrmodi für Satz-ID-Indexsuchen mit einer einzigen qualifizierten Zeile
- Tabelle 6. Sperrmodi für Satz-ID-Indexsuchen nur mit Start- und Stoppvergleichselementen
- Tabelle 7. Sperrmodi für Satz-ID-Indexsuchen nur mit Indexvergleichselementen
- Tabelle 8. Sperrmodi für Satz-ID-Indexsuchen mit anderen Vergleichselementen (sargs, resids)
- Tabelle 9. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Satz-ID-Indexsuche ohne Vergleichselemente
- Tabelle 10. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Satz-ID-Indexsuche (RID) ohne Vergleichselemente
- Tabelle 11. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Satz-ID-Indexsuche (RID) mit Vergleichselementen (sargs, resids)
- Tabelle 12. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)
- Tabelle 13. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen
- Tabelle 14. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen

Anmerkung: Sperrmodi können explizit mit einer *Sperranforderungsklausel* (LOCK REQUEST) einer SELECT-Anweisung geändert werden.

Tabelle 16. 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/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 17. Sperrmodi für Tabellensuchen mit Vergleichselementen nur für Dimensionsspalten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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 18. 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	UPDATE oder DELETE
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

Tabelle 19. 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/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 20. Sperrmodi für Satz-ID-Indexsuchen (RID) mit einer einzigen qualifizierten Zeile

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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 21. 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/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 22. 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	UPDATE oder DELETE
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 23. 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	UPDATE oder DELETE
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 24. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Satz-ID-Indexsuche (RID) 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/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 25. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Satz-ID-Indexsuche (RID) ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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 26. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Satz-ID-Indexsuche (RID) 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/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 27. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Satz-ID-Indexsuche (RID) 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	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 28. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Satz-ID-Indexsuche (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/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 29. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Satz-ID-Indexsuche (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	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

Sperrmodi für MDC-Blockindexsuchen

Der Typ von Sperre, den eine Tabelle mit mehrdimensionalem Clustering (MDC-Tabelle) bei einer Blockindexsuche aktiviert, hängt von der geltenden Isolationsstufe sowie vom verwendeten Datenzugriffsplan ab.

In den folgenden Tabellen werden die Typen von Sperren aufführt, die für MDC-Tabellen unter der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag hat drei Teile: die Tabellensperre, die Blocksperre und die Zeilensperre. Ein Bindestrich gibt an, dass eine bestimmte Sperrgranularität nicht verfügbar ist.

Die Tabellen 5 - 12 zeigen die Typen von Sperren, die für Blockindexsuchen aktiviert werden, wenn das Lesen von Datenseiten verzögert wird.

- Tabelle 1. Sperrmodi für Indexsuchen ohne Vergleichselemente
- Tabelle 2. Sperrmodi für Indexsuchen mit Vergleichselementen nur für Dimensionsspalten
- Tabelle 3. Sperrmodi für Indexsuchen nur mit Start- und Stoppvergleichselementen
- Tabelle 4. Sperrmodi für Indexsuchen mit Vergleichselementen
- Tabelle 5. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche ohne Vergleichselemente
- Tabelle 6. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Blockindexsuche ohne Vergleichselemente

- Tabelle 7. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche mit Vergleichselementen nur für Dimensionsspalten
- Tabelle 8. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Blockindexsuche Vergleichselementen nur für Dimensionsspalten
- Tabelle 9. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche nur mit Start- und Stoppvergleichselementen
- Tabelle 10. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Blockindexsuche nur mit Start- und Stoppvergleichselementen
- Tabelle 11. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche mit anderen Vergleichselementen (sargs, resids)
- Tabelle 12. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Blockindexsuche mit anderen Vergleichselementen (sargs, resids)

Anmerkung: Sperrmodi können explizit mit einer *Sperranforderungsklausel* (LOCK REQUEST) einer SELECT-Anweisung geändert werden.

Tabelle 30. Sperrmodi für Indexsuchen 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/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 31. Sperrmodi für Indexsuchen mit Vergleichselementen nur für Dimensionsspalten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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 32. 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	UPDATE oder DELETE
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 33. Sperrmodi für Indexsuchen mit Vergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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

Tabelle 34. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche 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/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 Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Blockindexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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 36. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche mit Vergleichselementen nur für Dimensionsspalten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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 37. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Nach einer Blockindexsuche Vergleichselementen nur für Dimensionsspalten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
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 38. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche 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/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 39. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: 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	UPDATE oder DELETE
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 40. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: Blockindexsuche mit 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/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 41. Sperrmodi für Indexsuchen, die für verzögerten Datenseitenzugriff verwendet werden: 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	UPDATE oder DELETE
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 Datenpartitionssperre wird in der Ausgabe des Befehls **db2pd**, von Ereignismonitoren, von Verwaltungssichten und von Tabellenfunktionen ausgewiesen.

Wenn auf eine Tabelle zugegriffen wird, wird zunächst eine Tabellensperre aktiviert. Anschließend werden Datenpartitionssperren nach Bedarf aktiviert. Aufgrund der verwendeten Zugriffsmethoden und Isolationsstufen ist es möglich, dass Datenpartitionen gesperrt werden müssen, die nicht in der Ergebnismenge vertreten sind. Wenn diese Datenpartitionssperren aktiviert wurden, können sie möglicherweise ebenso lange wie die Tabellensperre beibehalten werden. Bei einer Suche in einem Index unter der Isolationsstufe 'Cursorstabilität' (CS) können beispielsweise die Sperren für Datenpartitionen, auf die zuvor zugegriffen wurde, beibehalten werden, um den späteren Aufwand für eine erneute Aktivierung von Datenpartitionssperren zu verringern.

Datenpartitionssperren beinhalten auch den Aufwand für die Sicherstellung des Zugriffs auf Tabellenbereiche. Bei nicht partitionierten Tabellen wird der Zugriff auf Tabellenbereiche durch Tabellensperren gesteuert. Datenpartitionssperren werden aktiviert, auch wenn eine exklusive oder den gemeinsamen Zugriff zulassende Sperre (Share) auf Tabellenebene aktiviert ist.

Durch die feinere Differenzierung (Granularität) kann eine Transaktion exklusiven Zugriff auf eine bestimmte Datenpartition haben und Zeilensperren vermeiden, während andere Transaktionen auf andere Datenpartitionen zugreifen. Dies kann das Ergebnis des für eine Massenaktualisierung ausgewählten Zugriffsplans oder der Eskalation von Sperren auf die Datenpartitionsebene sein. 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 Datenpartitionsdeadlocks zwischen gleichzeitig zugreifenden Transaktionen zu vermeiden.

Anweisungen LOCK TABLE

Bei partitionierten Tabellen wird durch die Anweisung LOCK TABLE nur eine Sperre auf Tabellenebene angefordert. Dadurch werden Zeilensperren durch nachfolgende DML-Anweisungen (DML, Data Manipulation Language - Datenbearbeitungssprache) verhindert und Deadlocks auf Zeilen-, Block- oder Datenpartitionsebene vermieden. Mithilfe der Option IN EXCLUSIVE MODE kann ein exklusiver Zugriff bei der Aktualisierung von Indizes sichergestellt werden. Dies ist sinnvoll, um das Anwachsen von Indizes während einer umfangreichen Aktualisierung zu begrenzen.

Auswirkung der Option LOCKSIZE TABLE der Anweisung ALTER TABLE

Die Option LOCKSIZE TABLE stellt sicher, dass eine Tabelle im Modus für gemeinsamen Zugriff (Share) oder im Exklusivmodus ohne Intent-Sperren gesperrt wird. Bei einer partitionierten Tabelle wird diese Sperrenstrategie sowohl auf die Tabellensperre als auch auf Datenpartitionssperren angewendet.

Sperreneskalation auf Zeilen- und Blockebene

Sperren auf Zeilen- und Blockebene in partitionierten Tabellen können auf die Partitionsebene eskaliert werden. Wenn dies geschieht, können andere Transaktionen besser auf die Tabelle zugreifen, selbst wenn die Sperre für eine Datenpartition auf den gemeinsamen, exklusiven oder superexklusiven Modus (Share, Exclusive oder Super Exclusive) eskaliert wird, weil andere Datenpartitionen davon unberührt bleiben. Der Benachrichtigungsprotokolleintrag für eine Eskalation enthält die betroffene Datenpartition und den Namen der Tabelle.

Ein exklusiver Zugriff auf einen nicht partitionierten Index kann durch eine Sperreneskalation nicht sichergestellt werden. Für einen exklusiven Zugriff müssen folgende Bedingungen erfüllt sein:

- Die Anweisung muss eine exklusive Sperre auf Tabellenebene verwenden.
- Es muss die explizite Anweisung LOCK TABLE IN EXCLUSIVE MODE abgesetzt werden.
- Die Tabelle muss über das Attribut LOCKSIZE TABLE verfügen.

Bei partitionierten Indizes wird der exklusive Zugriff auf eine Indexpartition durch eine Sperreneskalation zu einem exklusiven Zugriffsmodus bzw. einem Zugriff im Modus "Z" (Super Exclusive) sichergestellt.

Interpretieren von Informationen über Sperren

Die Verwaltungssicht SNAPLOCK kann Sie bei der Interpretation von Sperreninformationen unterstützen, die für eine partitionierte Tabelle zurückgegeben werden. 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 L_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 L_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 (Tabellensperre) und die Datenpartitions-ID (DATA_PARTITION_ID) mit dem Wert -1 verwendet, um den Zugriff auf die partitionierte Tabelle TP1 und die gleichzeitige Verwendbarkeit dieser Tabelle zu steuern. Die Sperrobjekte des Typs TABLE_PART_LOCK dienen zur Steuerung des größtmöglichen Zugriffs auf jede Datenpartition und ihrer größtmöglichen gemeinsamen Nutzbarkeit.

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.

Kapitel 17. Agentenverwaltung

Agenten in einer partitionierten Datenbank

In einer Umgebung mit partitionierten Datenbanken oder in einer Umgebung mit aktivierter partitionsinterner Parallelität verfügt jede Datenbankpartition 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 zugeordnete Agenten im Pool bleiben und vom Datenbankmanager für neue Anforderungen von der Anwendung, der sie zugeordnet sind, oder von neuen Anwendungen verwendet werden.

Der Einfluss auf die Leistung und die Speicherbelegung innerhalb des Systems ist eng mit der Optimierung des Agentenpools verbunden. Der Konfigurationsparameter des Datenbankmanagers für die Größe des Agentenpools (**num_poolagents**) betrifft die Gesamtzahl von Agenten und Subagenten, die Anwendungen in einer Datenbankpartition 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 erneut Anwendungen zugeordnet werden müssen, sinkt die Leistung.

Standardmäßig hat der Parameter **num_poolagents** die Einstellung AUTOMATIC mit einem Wert von 100 und der Datenbankmanager verwaltet die Anzahl der inaktiven Agenten im Pool automatisch.

Wenn der Parameter **num_poolagents** manuell auf einen zu niedrigen Wert gesetzt wird, könnte eine Anwendung allein den Pool mit zugeordneten Subagenten füllen. Wenn nun eine andere Anwendung einen neuen Subagenten benötigt und keine Agenten im zugehörigen Agentenpool hat, übernimmt und verwendet sie inaktive Subagenten aus den Agentenpools anderer Anwendungen. Dieses Verhalten stellt sicher, dass Ressourcen vollständig genutzt werden.

Wenn der Parameter **num_poolagents** manuell auf einen zu hohen Wert gesetzt wird, verbleiben zugeordnete Subagenten über einen langen Zeitraum ungenutzt im Pool und belegen Datenbankmanagerressourcen, die dadurch für andere Tasks nicht verfügbar sind.

Wenn der Verbindungskonzentrator aktiviert ist, gibt der Wert des Parameters **num_poolagents** nicht unbedingt die exakte Anzahl der Agenten wieder, die sich zu einem bestimmten Zeitpunkt inaktiv im Pool befinden können. Agenten werden möglicherweise vorübergehend zum Auffangen höherer Auslastungsaktivitäten benötigt.

Neben Datenbankagenten werden auch 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-Listenerfunktionen
- Funktionen zum Neuausgleich von Daten in Tabellenbereichscontainern

Kapitel 18. Optimieren von Zugriffsplänen

Indexzugriff und Clusterverhältnisse

Bei der Auswahl des Zugriffsplans schätzt das Optimierungsprogramm die Anzahl der E/A-Operationen, die zum Einlesen der Seiten von der Platte in den Pufferpool erforderlich sind. Diese Schätzung schließt eine Voraussage über die Nutzung des Pufferpools ein, da zum Lesen von Zeilen einer Seite, die sich bereits im Pufferpool befindet, keine zusätzlichen E/A-Operationen anfallen.

Für Indexsuchen wird das Optimierungsprogramm durch Informationen aus dem Systemkatalog bei der Abschätzung des Ein-/Ausgabeaufwands zum Lesen von Datenseiten in einen Pufferpool unterstützt. Dabei werden Informationen aus den folgenden Spalten der Sicht 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, falls 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 Einlesen der Datenseiten in Pufferpools verschiedener Größen angeben. Für diese Spalten werden Daten nur erfasst, wenn Sie den Befehl **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 der Clusterbildung der Daten kann bedeutende Auswirkungen auf die Leistung haben, sodass einer der Indizes, die für die Tabelle definiert sind, 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 für einen Index eine Obermenge der Schlüssel für den Clusterindex darstellen oder in denen es eine tatsächliche 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 diese Clusterbildung bei der Verarbeitung von Einfügungen beibehalten wird. Da Aktualisierungs- und Einfügeoperationen die Clusterbildung in Bezug auf den Index verringern können, müssen Sie die Tabelle eventuell in regelmäßigen Abständen reorganisieren. Zur Verringerung der Anzahl von Reorganisationen für eine Tabelle, an der häufig Einfüge-, Aktualisierungs- oder Löschoperationen ausgeführt werden, geben Sie die Klausel PCTFREE in der Anweisung ALTER TABLE an.

Tabellen- und Indexverwaltung für MDC- und ITC-Tabellen

Die Tabellen- und Indexorganisation für MDC-Tabellen (MDC = Multidimensional Clustering) und ITC-Tabellen (ITC = Insert Time Clustering) basiert auf denselben logischen Strukturen wie die Organisation von Standardtabellen.

Ebenso wie Standardtabellen werden MDC- und ITC-Tabellen in Seiten organisiert, die Datenzeilen enthalten, die wiederum in Spalten unterteilt sind. Die Zeilen jeder Seite werden durch Satz-IDs (RIDs) gekennzeichnet. Die Seiten von MDC- und ITC-Tabellen werden jedoch in EXTENTSIZE große Blöcke gruppiert. Zum Beispiel zeigt Abb. 52 eine Tabelle mit dem EXTENTSIZE-Wert 4. Die ersten vier Seiten mit den Nummern 0 bis 3 bilden den ersten Block in der Tabelle. Die nächsten vier Seiten mit den Nummern 4 bis 7 bildet den zweiten Block in der Tabelle.

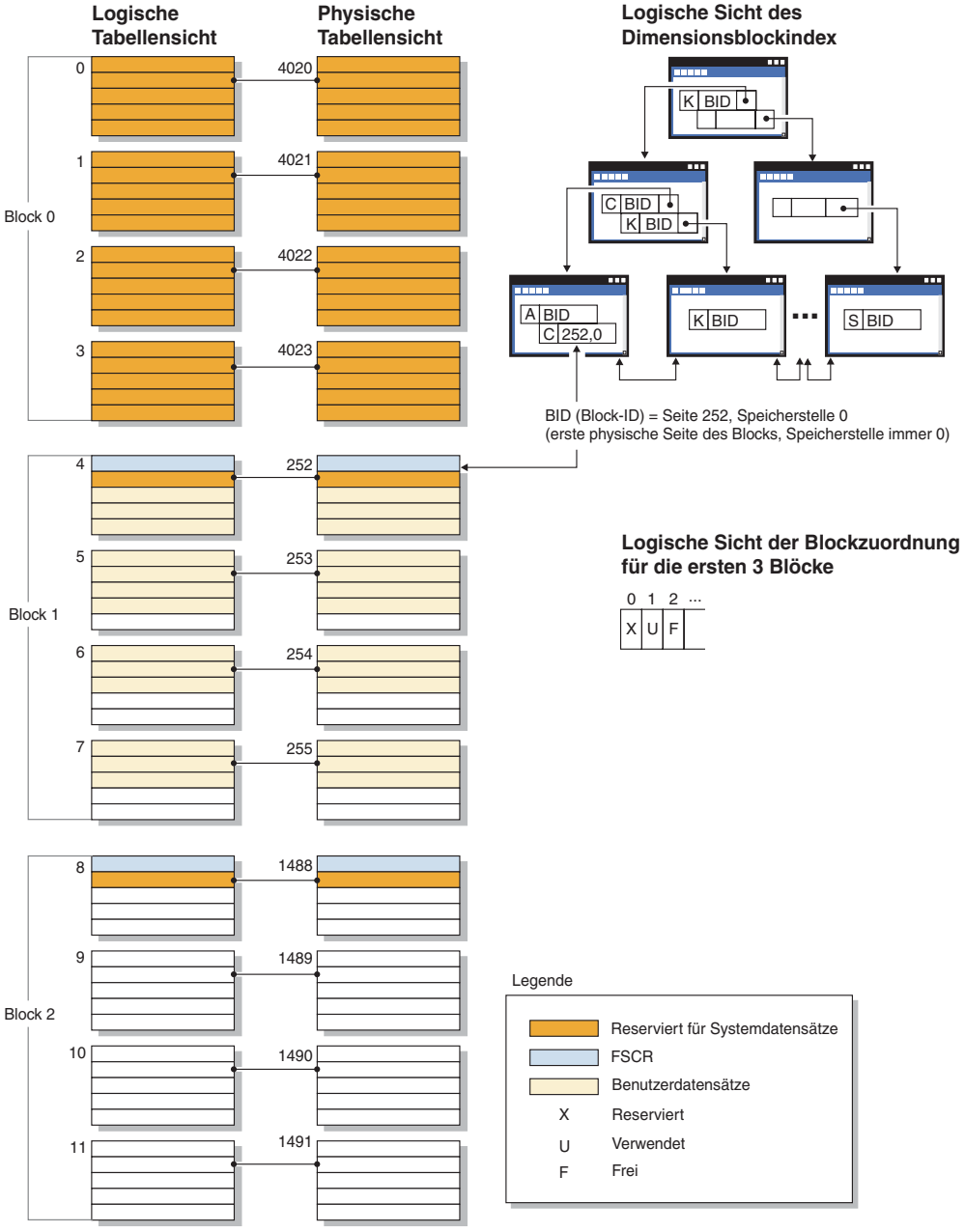


Abbildung 52. Logische Tabellen-, Datensatz- und Indexstruktur für MDC- und ITC-Tabellen

Der erste Block enthält besondere interne Datensätze, einschließlich des Steuersatzes für freien Speicherbereich (Free Space Control Record, FSCR), die vom DB2-Server zur Verwaltung der Tabelle verwendet werden. In den nachfolgenden Blöcken enthält jeweils die erste Seite den FSCR-Satz. Ein FSCR-Satz ordnet den freien

Speicher für neue Datensätze zu, der auf jeder Seite des Blocks 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 Indizes automatisch erstellt:

- Ein Dimensionsblockindex, der Zeiger auf jeden belegten Block für eine einzelne Dimension enthält
- Ein zusammengesetzter Blockindex, der alle Dimensionsschlüsselspalten enthält und zur Aufrechterhaltung des Clusterings bei Einfüge- und Aktualisierungsaktivitäten verwendet wird

Das Optimierungsprogramm zieht Zugriffspläne, die Dimensionsblockindizes verwenden, 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 Daten aus diesen abzurufen. Da sich diese Speicherbereiche in physisch aufeinander folgenden Seiten auf der Platte befinden, wird der E/A-Aufwand minimiert und eine bessere Leistung erzielt.

Sie können auch spezielle Satz-ID-Indizes (RID-Indizes) erstellen, wenn eine Analyse von Datenzugriffsplänen nahe legt, dass solche Indizes die Abfrageleistung verbessern würden.

Wie am Namen erkennbar, gruppieren ITC-Tabellen die Daten anhand der Zeileneinfügungszeit. Die MDC- und ITC-Tabellen unterscheiden sich wie folgt:

- Blockindizes werden nie für den Datenzugriff verwendet.
- Nur ein einzelner zusammengesetzter Blockindex wird für die Tabelle erstellt und dieser Index besteht aus einer virtuellen Dimension.
- Der Index wird keinesfalls vom Optimierungsprogramm für Pläne ausgewählt, da von einer SQL-Anweisung nicht auf die enthaltene Spalte verwiesen werden kann.

Aus MDC- und ITC-Tabellen können freie Blöcke an den Tabellenbereich freigegeben werden.

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 der Parallelität, der bei der Kompilierung der SQL-Anweisung angegeben wurde.

Zur Parallelisierung eines Zugriffsplans 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 Sub-

agenten. 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 im selben Index ausgeführt werden. Für parallele Tabellensuchen wird die Tabelle in Seiten- oder Zeilenbereiche unterteilt, die Subagenten zugewiesen werden. 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 eine parallele Tabellensuche 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.

Parallele Tabellensuchen können für bereichspartitionierte Tabellen ausgeführt werden. In ähnlicher Weise können parallele Indexsuchen für partitionierte Indizes ausgeführt werden. Für eine parallele Suche werden partitionierte Indizes auf der Basis von Indexschlüsselwerten und der Anzahl von Schlüsseleinträgen für einen Schlüsselwert in Bereiche von Datensätzen unterteilt. Wenn eine parallele Suche beginnt, wird den Subagenten ein Bereich von Datensätzen zugeordnet. Wenn ein Subagent einen Bereich durchsucht hat, wird ihm ein neuer Bereich zugeordnet. Die Indexpartitionen werden sequenziell durchsucht, wobei die Subagenten potenziell nicht reservierte Indexpartitionen zu einem beliebigen Zeitpunkt durchsuchen, ohne aufeinander zu warten. Nur der Teil der Indexpartitionen, der nach der Analyse zum Ausschluss von Datenpartitionen als für die Abfrage relevant eingestuft wird, wird durchsucht.

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 werden, werden die Maschinenressourcen 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, um die Daten effizient auf alle Subagenten möglichst gleichmäßig zu verteilen. Zur Realisierung der gleichmäßigen Vertei-

lung 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 der Reihumsortierung 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 zur Neuverteilung des Datenstroms verwendet werden.
- Gemeinsame Sortierung

  Diese Art der Sortierung entspricht der replizierten Sortierung, abgesehen davon, dass Subagenten eine parallele Suche über das Sortierergebnis öffnen, um die Daten unter den Subagenten ähnlich wie bei einer Reihumsortierung zu verteilen.

Temporäre Tabellen mit partitionsinterner Parallelität

Subagenten können kooperieren, um eine temporäre Tabelle durch Einfügen von Zeilen in dieselbe Tabelle zu erstellen. Eine solche Tabelle ist eine *gemeinsam genutzte temporäre Tabelle*. Die Subagenten können private oder parallele Suchoperationen über die gemeinsam genutzte temporäre Tabelle öffnen, je nachdem, ob der Datenstrom repliziert oder geteilt werden muss.

Strategien zur partitionsinternen parallelen Aggregation

Spaltenberechnungen (Aggregation) können von Subagenten parallel ausgeführt 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 ausführen. Dies ist möglich, wenn der Strom bereits aufgrund einer früheren partitionierten Sortierung über die Gruppierungsspalten geteilt ist.

Andernfalls kann der Subagent eine partielle Spaltenberechnung ausführen 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 für partitionsinterne parallele Joins

Joinoperationen können von Subagenten parallel ausgeführt 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 (Nested Loop Join) 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.

Durch Datenfilterung und ungleiche Datenverteilung können Auslastungen zwischen Subagenten während der Ausführung einer Abfrage unausgeglichen werden. Die Ineffizienz unausgeglichener Auslastungen wird durch Joins und andere rechenintensive Operationen noch verstärkt. Das Optimierungsprogramm sucht im Zugriffsplan einer Abfrage nach Quellen für Unausgeglichenheit und wendet eine Ausgleichsstrategie an, um sicherzustellen, dass die Arbeitslast gleichmäßig auf die Subagenten verteilt wird. Für einen nicht geordneten äußeren Datenstrom gleicht das Optimierungsprogramm den Join mithilfe des Operators REBAL für den äußeren Datenstrom aus. Für einen geordneten Datenstrom (bei dem die Daten durch einen Indexzugriff oder eine Sortierung geordnet werden) gleicht das Optimierungsprogramm die Daten durch eine gemeinsame Sortierung aus. Eine gemeinsame Sortierung wird nicht verwendet, wenn die Sortierung einen Datenüberlauf in die temporären Tabellen zur Folge hätte, da dies mit hohem Aufwand verbunden wäre.

Joins

Ein *Join* ist der Prozess der Kombination von Daten 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 verknüpft, wenn die Informationen in den entsprechenden Zeilen die Joinbedingung (das *Joinvergleichselement*) erfüllen.

Betrachten Sie zum Beispiel die beiden folgenden Tabellen:

TABLE1

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

TABLE2

PROJ_ID	NAME
1	Sam
3	Joe
4	Mary
1	Sue
2	Mike

Für den Join von TABLE1 und TABLE2, sodass die Spalten PROJ_ID 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
```

In diesem Fall sieht das entsprechende Joinvergleichselement wie folgt aus: `where x.proj_id = y.proj_id`.

Die Abfrage liefert die folgende Ergebnismenge:

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

Abhängig von der Spezifik der Joinvergleichselemente sowie von Aufwänden, die auf der Basis von 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 ausgewählt und die andere als innere Tabelle des Joins betrachtet. Auf die äußere Tabelle wird zuerst zugegriffen und sie wird nur einmal durchsucht. Ob die innere Tabelle mehrere Male durchsucht wird, hängt vom Typ des Joins und den verfügbaren Indizes ab. Auch wenn in einer Abfrage mehr als zwei Tabellen durch einen Join 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, und anschließend die Abfrageleistung analysieren, um zu entscheiden, ob Joinoperatoren hinzugefügt werden sollen.

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 so aufgeteilt werden, dass 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 durch Kollokation zusammengefasst werden.

Abhängig von der Größe der Tabelle reduziert das Verteilen der Daten über mehrere Datenbankpartitionen die geschätzte Dauer der Abfrageausführung. 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.

Joinstrategien für partitionierte Datenbanken

Joinstrategien für eine Umgebung mit partitionierten Datenbanken können sich von Strategien für eine nicht partitionierte Datenbankumgebung unterscheiden. Zur Verbesserung der Leistung können zusätzliche Techniken auf die Standardjoinmethoden angewendet werden.

Für Tabellen, die häufig durch einen Join verknüpft werden, sollte eine Tabellenkollokation in Betracht gezogen werden. In einer Umgebung mit partitionierten Datenbanken bezeichnet der Begriff *Tabellenkollokation* einen Zustand, der vorliegt, wenn zwei Tabellen, die dieselbe Anzahl kompatibler Patitionierungsschlüssel haben, in derselben Datenbankpartitionsgruppe gespeichert werden. Wenn dies der Fall ist, kann die Joinverarbeitung in der Datenbankpartition ausgeführt werden, in der die Daten gespeichert sind, und nur die Ergebnismenge muss an die Koordinatordatenbankpartition übertragen werden.

Tabellenwarteschlangen

In Beschreibungen von Joinmethoden in einer Umgebung mit partitionierten Datenbanken wird die folgende Terminologie verwendet:

- Eine *Tabellenwarteschlange* (auch als TQ (Table Queue) bezeichnet) ist ein Mechanismus zur Übertragung von Zeilen zwischen Datenbankpartitionen oder zwischen Prozessoren in einer Einzelpartitionsdatenbank.
- Eine *gezielt übertragene Tabellenwarteschlange* (auch als DTQ (Directed Table Queue) bezeichnet) ist eine Tabellenwarteschlange, in der Zeilen durch ein Hashverfahren an eine der empfangenden Datenbankpartitionen gesendet werden.
- Eine *Broadcast-Tabellenwarteschlange* (auch als BTQ (Broadcast Table Queue) bezeichnet) ist eine Tabellenwarteschlange, in der Zeilen an alle empfangenden Datenbankpartitionen, jedoch ohne Hashverfahren, gesendet werden.

Eine Tabellenwarteschlangen dient zur Übergabe von Tabellendaten zwischen folgenden Komponenten:

- Von einer Datenbankpartition zu einer anderen bei Verwendung partitionsübergreifender Parallelität
- Innerhalb einer Datenbankpartition bei Verwendung partitionsinterner Parallelität
- Innerhalb einer Datenbankpartition bei Verwendung einer Einzelpartitionsdatenbank

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 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 Abrufanweisung (FETCH) von einer Anwendung 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 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 Mischjoins

  Bei dieser Art von Tabellenwarteschlange bleibt die Reihenfolge gewahrt.
- Reguläre Tabellenwarteschlangen

  Reguläre Tabellenwarteschlangen wahren nicht die Reihenfolge.
- Empfangende Tabellenwarteschlange (manchmal auch als LTQ (Listener Table Queue) bezeichnet)

  Diese Tabellenwarteschlangen werden mit korrelierten Unterabfragen verwendet. Die Korrelationswerte werden an die Unterabfrage übergeben und die Ergebnisse mithilfe dieses Typs von Tabellenwarteschlange an den übergeordneten Abfrageblock zurückgeliefert.

Joinmethoden für partitionierte Datenbanken

Für Umgebungen mit partitionierten Datenbanken stehen verschiedene Joinmethoden zur Verfügung. Dazu gehören: zusammengefasste Joins, Broadcast-Outer-Table-Joins, Directed-Outer-Table-Joins, Directed-Inner-Table- und Directed-Outer-Table-Joins, Broadcast-Inner-Table-Joins und Directed-Inner-Table-Joins.

In den folgenden Diagrammen bezeichnen q1, q2 und q3 ('Queue') Tabellenwarteschlangen. Die gezeigten Tabellen sind auf zwei Datenbankpartitionen verteilt, und die Pfeile geben die Richtung an, in der die Tabellenwarteschlangen übertragen bzw. gesendet werden. Die Koordinatordatenbankpartition ist Datenbankpartition 0.

Wenn die vom Compiler gewählte Joinmethode ein Hash-Join ist, können die in allen fernen Datenbankpartitionen erstellten Filter zum Eliminieren von Tupeln verwendet werden, bevor sie an die Datenbankpartition gesendet werden, wo der Hash-Join ausgeführt wird. Dadurch wird die Leistung verbessert.

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 der entsprechenden Verteilungsschlüssel müssen in den Gleichheitsvergleichselementen enthalten sein. Abb. 53 zeigt ein Beispiel.

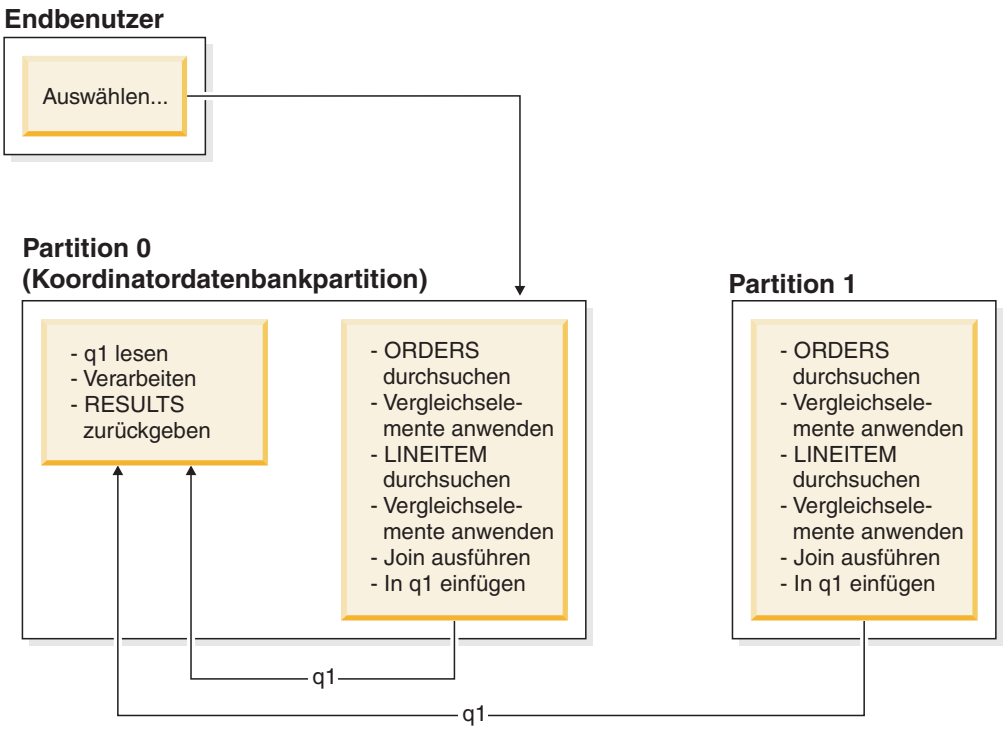


Abbildung 53. Beispiel für einen zusammengefassten Join

Die Tabellen LINEITEM und ORDERS werden beide über die Spalte ORDERKEY partitioniert. Der Join wird lokal in jeder Datenbankpartition ausgeführt. In diesem Beispiel wird folgendes Vergleichselement für den Join angenommen: `orders.orderkey = lineitem.orderkey`.

Replizierte MQTs (Materialized Query Tables) erhöhen die Wahrscheinlichkeit zusammengefasster Joins.

Broadcast-Outer-Table-Joins

Broadcast-Outer-Table-Joins (Joins mit rundgesendeter äußerer Tabelle) stellen eine parallele Joinstrategie dar, die angewendet werden kann, wenn zwischen den zu verknüpfenden Tabellen keine Equijoin-Vergleichselemente vorhanden sind. Sie kann außerdem in solchen Fällen verwendet werden, in denen sie sich als die Methode mit dem geringsten Aufwand herausstellt. 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 über die Spalten verteilt ist, die im Joinvergleichselement verwendet werden. Anstatt beide Tabellen zu teilen, kann es günstiger sein, die kleinere Tabelle an alle Datenbankpartitionen mit der größeren Tabelle per Broadcast rundzusenden. Abb. 54 zeigt ein Beispiel.

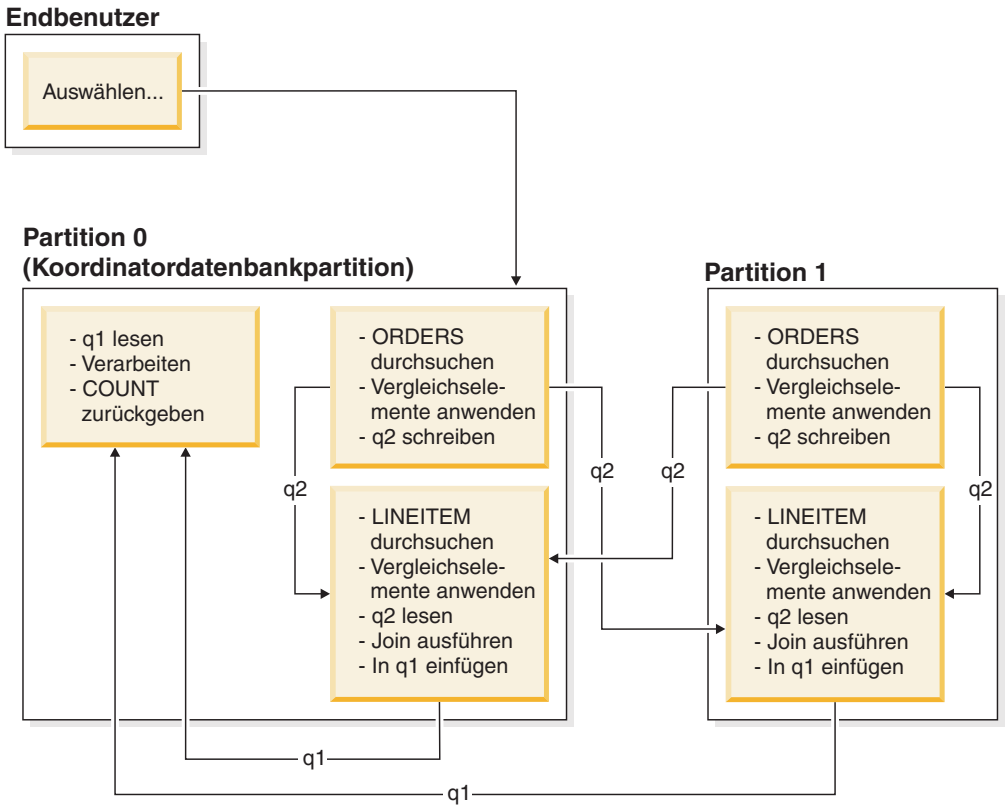
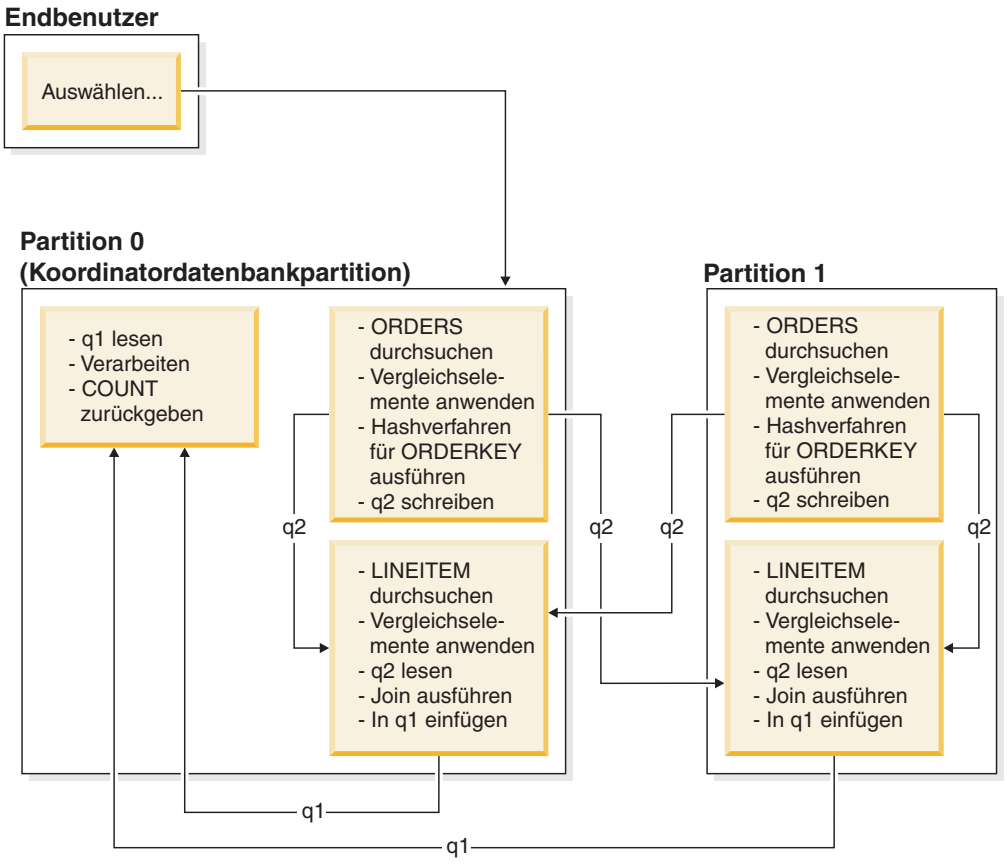


Abbildung 54. Beispiel für einen Broadcast-Outer-Table-Join

Die Tabelle ORDERS wird an alle Datenbankpartitionen mit der Tabelle LINEITEM gesendet. Tabellenwarteschlange q2 wird per Broadcast an alle Datenbankpartitionen der inneren Tabelle gesendet.

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. Abb. 55 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 55. 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. Abb. 56 zeigt ein Beispiel.

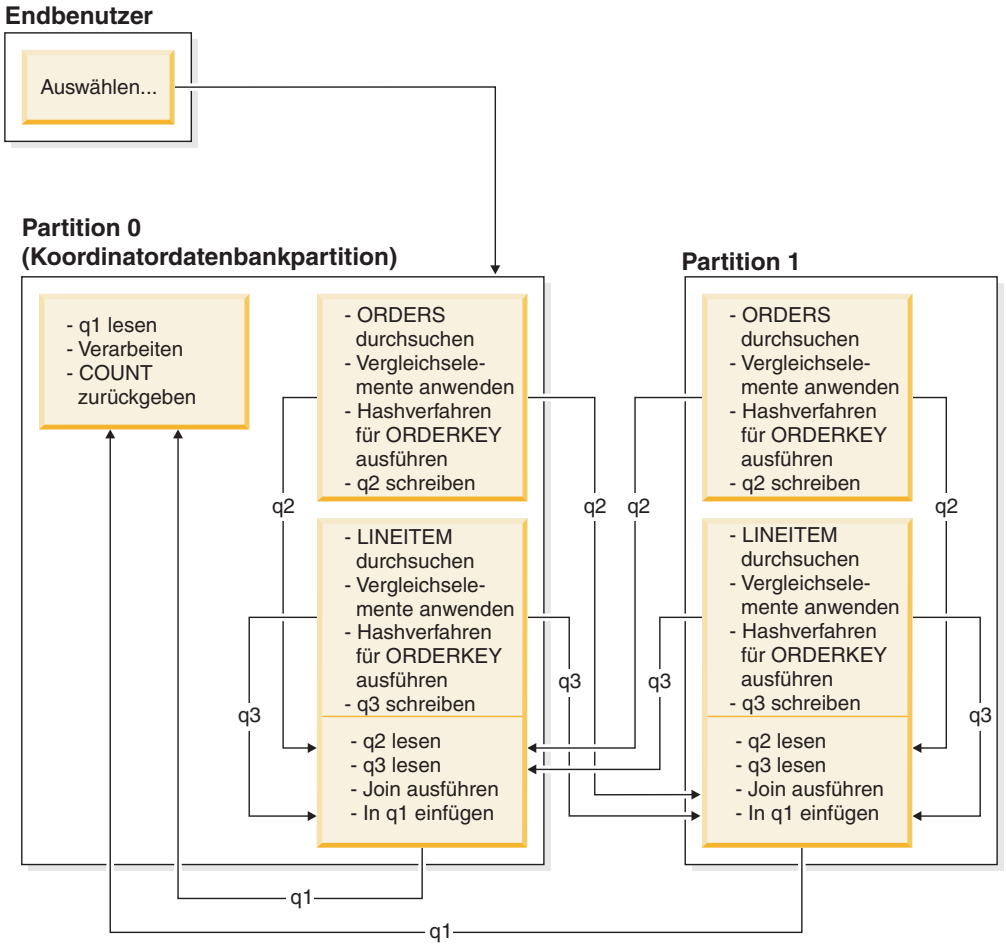
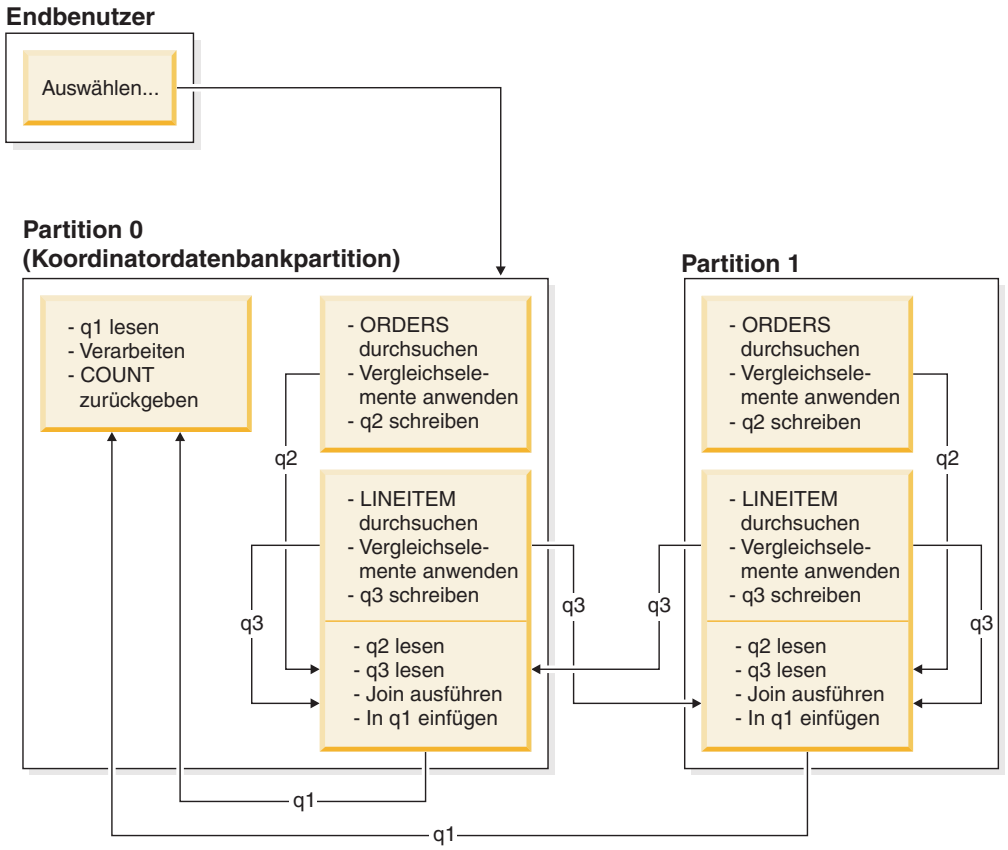


Abbildung 56. Beispiel für einen Directed-Inner-Table- und Directed-Outer-Table-Join

Es wird keine Tabelle über die Spalte ORDERKEY partitioniert. Für beide Tabellen wird das Hashverfahren ausgeführt und sie werden an neue Datenbankpartitionen gesendet, wo sie durch einen Join verknüpft werden. Beide Tabellenwarteschlangen q2 und q3 werden gezielt übertragen. In diesem Beispiel wird folgendes Vergleichselement für den Join angenommen: `orders.orderkey = lineitem.orderkey`.

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 Tabelle gesendet. Abb. 57 zeigt ein Beispiel.

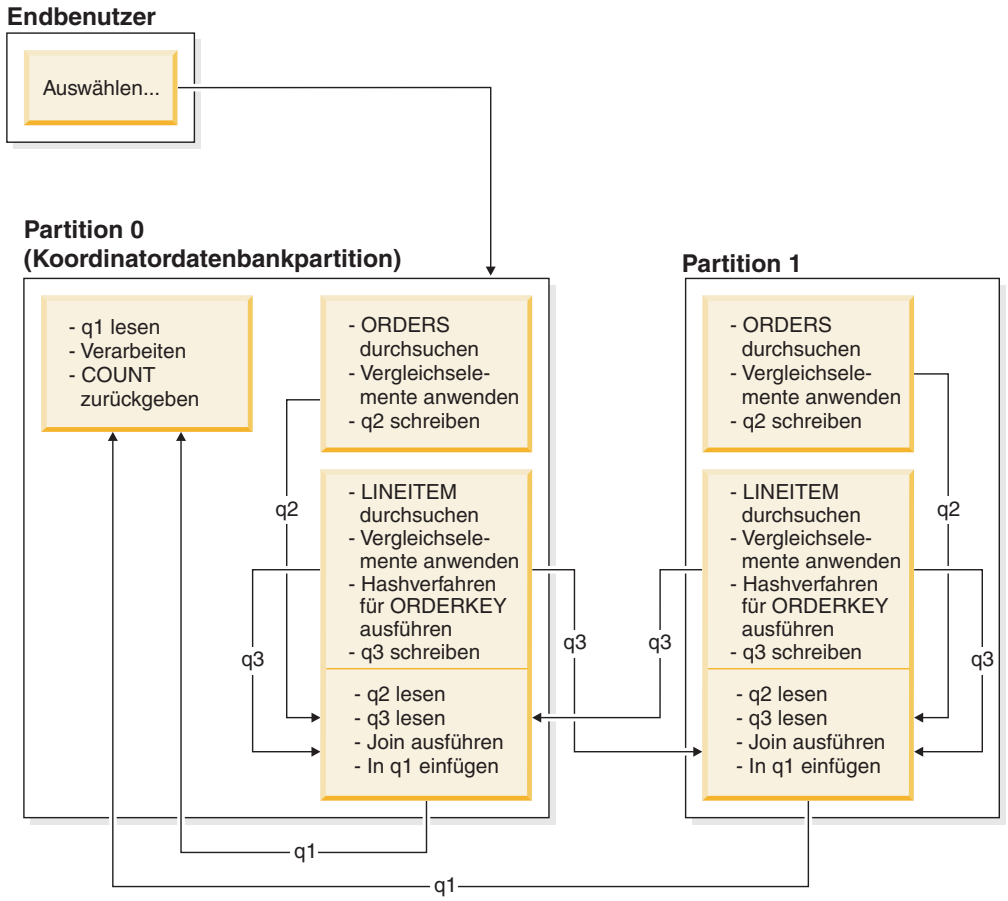


Die Tabelle LINEITEM wird an alle Datenbankpartitionen mit der Tabelle ORDERS gesendet. Tabellenwarteschlange q3 wird per Broadcast an alle Datenbankpartitionen der äußeren Tabelle gesendet.

Abbildung 57. 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. Abb. 58 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 58. Beispiel für einen Directed-Inner-Table-Join

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.

Beachten Sie, dass die Replikation einer replizierten MQT in diesem Kontext eine datenbankinterne Replikation ist. Bei der datenbankübergreifenden Replikation spielen Subskriptionen, Steuertabellen und Daten, die sich in verschiedenen Datenbanken und auf verschiedenen Betriebssystemen befinden, eine Rolle.

Betrachten Sie das folgende Beispiel:

- Die Tabelle SALES (Verkauf) befindet sich im Mehrpartitionstabellenbereich mit dem Namen REGIONTABLESPACE und wird über die Spalte REGION unterteilt.
- Die Tabellen EMPLOYEE (Mitarbeiter) und DEPARTMENT (Abteilung) sind in einer Datenbankpartitionsgruppe mit einer Einzelpartition.

Sie erstellen 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
```

Sie aktualisieren den Inhalt der replizierten MQT:

```
refresh table r_employee
```

Nach der Verwendung der Anweisung REFRESH sollten Sie das Dienstprogramm RUNSTATS für die replizierte Tabelle in gleicher Weise wie für andere Tabellen ausführen.

Die folgende Abfrage berechnet den Verkauf nach Mitarbeiter, die Summe für die Abteilung und die Gesamtsumme:

```
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 sich nur in einer Datenbankpartition befindet, verwendet der Datenbankmanager R_EMPLOYEE, d. h. die MQT, die in jede der Datenbankpartitionen repliziert wird, in der die Tabelle SALES gespeichert ist. Der Leistungsvorteil ergibt sich daraus, dass die Mitarbeiterinformationen zur Ausführung des Joins nicht an jede Datenbankpartition über das Netz gesendet werden müssen.

Replizierte MQTs in zusammengefassten Joins

Replizierte MQTs können auch bei der Zusammenfassung (Kollokation) von Joins helfen. Wenn beispielsweise ein Sternschema eine große Fakttabelle enthält, die sich über zwanzig Datenbankpartitionen 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, eine Tabelle mit dem Namen DIM1 (C1, dim1a, dim1b,...), die über C1 unterteilt ist, eine Tabelle mit dem Namen DIM2 (C2, dim2a, dim2b,...), die über C2 unterteilt ist, usw. In diesem Fall ist der Join zwischen FACT und DIM1 perfekt, da das Vergleichselement `dim1.c1 = fact.c1` von den durch Kollokation zusammengefasst Daten profitiert. Beide Tabellen sind über die Spalte C1 unterteilt.

Der Join mit der Tabelle DIM2 und dem Vergleichselement `dim2.c2 = fact.c2` kann jedoch nicht mit durch Kollokation zusammengefassten Daten realisiert werden, da FACT über die Spalte C1 unterteilt ist und nicht über die Spalte C2. In diesem Fall könnten Sie die Tabelle DIM2 in der Datenbankpartitionsgruppe der Fakttabelle replizieren, sodass der Join lokal in jeder Datenbankpartition erfolgen kann.

Wenn Sie eine replizierte MQT erstellen, kann die Quellentabelle eine Einzelpartitionstabelle oder eine Mehrpartitionstabelle in einer Datenbankpartitionsgruppe sein. In den meisten Fällen ist die replizierte Tabelle klein und kann in einer Datenbankpartitionsgruppe mit einer Einzelpartition untergebracht werden. Sie können die zu replizierende Datenmenge einschränken, indem Sie nur einen Teil der Spalten der Tabelle angeben oder indem Sie die Anzahl der qualifizierten Zeilen durch Vergleichselemente begrenzen.

Eine replizierte MQT kann auch in einer Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen 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 in der Quellentabelle nicht vorhanden sind, können Sie jedoch keine eindeutigen Indizes erstellen oder Integritätsbedingungen für replizierte Tabellen definieren, selbst wenn diese Integritätsbedingungen für die Quellentabelle gelten.

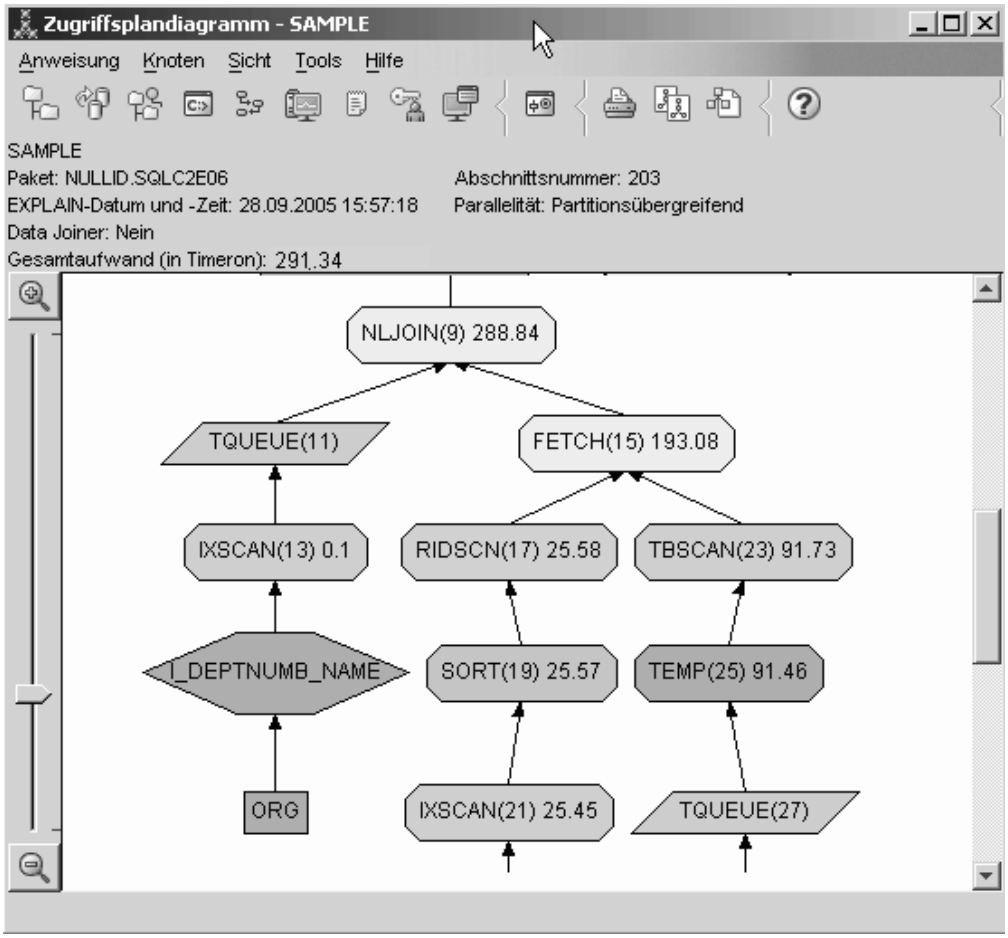
Auf replizierte Tabellen kann in einer Abfrage direkt verwiesen werden, jedoch können Sie nicht die Skalarfunktion DBPARTITIONNUM für eine replizierte Tabelle verwenden, um die Tabellendaten in einer bestimmten Datenbankpartition abzurufen.

Verwenden Sie die DB2-EXPLAIN-Funktion, um festzustellen, ob eine replizierte MQT vom Zugriffsplan für eine Abfrage genutzt wurde. Ob der Zugriffsplan, der vom Optimierungsprogramm ausgewählt wird, eine replizierte MQT verwendet, hängt von den Daten ab, die durch einen Join verknüpft werden müssen. Eine replizierte MQT könnte zum Beispiel dann nicht verwendet werden, wenn das Optimierungsprogramm feststellt, dass es günstiger wäre, die ursprüngliche Quellentabelle an die anderen Datenbankpartitionen der Datenbankpartitionsgruppe per Broadcast rundzusenden.

Erstellen zusätzlicher Indizes für Tabellenspalten in einer Umgebung mit partitionierten Datenbanken

Dieses Beispiel baut auf dem Zugriffsplan auf, der in Abfrage 3 beschrieben wurde, indem ein Index für die Spalte JOB in der Tabelle STAFF erstellt wird und indem DEPTNAME zum vorhandenen Index in der Tabelle ORG hinzugefügt wird. (Das Hinzufügen eines separaten Index könnte zu einem zusätzlichen Zugriff führen.)

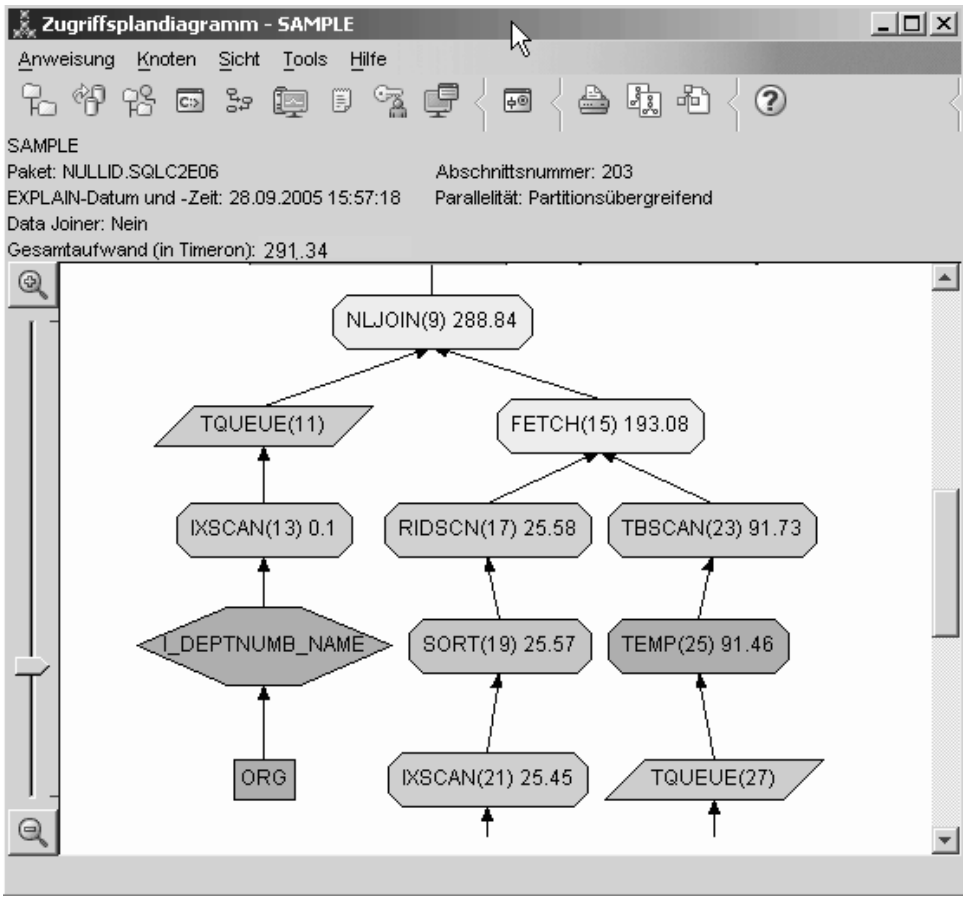
Wenn Sie das Zugriffsplandiagramm für diese Abfrage (Abfrage 4) anzeigen wollen, klicken Sie im Fenster mit dem Protokoll der mit EXPLAIN bearbeiteten Anweisungen den Eintrag doppelt an, der als Abfrage Nummer 4 angegeben ist. Das Fenster **Zugriffsplandiagramm** für diese Ausführung der Anweisung wird geöffnet.



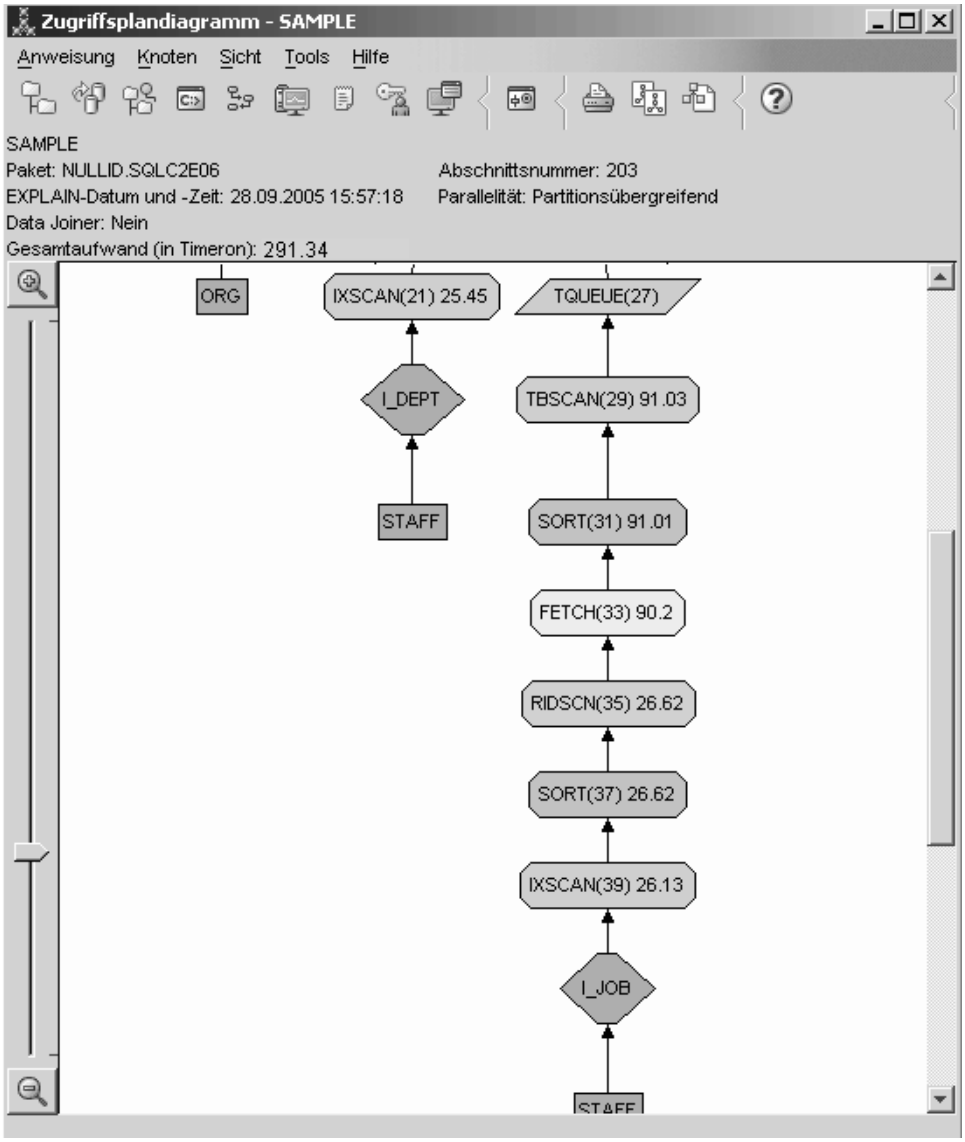
Durch Beantwortung der folgenden Fragen erhalten Sie Hinweise dazu, wie Sie die Abfrage verbessern können.

1. Was ändert sich in diesem Prozessplan aufgrund der Erstellung zusätzlicher Indizes?

   Beachten Sie im mittleren Teil des Zugriffsplandiagramms, dass für die Tabelle ORG die vorherige Tabellensuche in die Indexsuche IXSCAN(7) geändert wurde. Durch das Hinzufügen der Spalte DEPTNAME zum Index für die Tabelle ORG kann das Optimierungsprogramm den Zugriff eingrenzen, der die Tabellensuche beinhaltet.



   Beachten Sie im unteren Teil des Zugriffsplandiagramms, dass für die Tabelle STAFF die vorherige Indexsuche und die FETCH-Operation in die Indexsuche IXSCAN(39) geändert wurden. Durch das Erstellen des Index JOB für die Tabelle STAFF kann das Optimierungsprogramm den zusätzlichen Zugriff, der die FETCH-Operation beinhaltet, ausschließen.



2. Wie effektiv ist dieser Zugriffsplan?

   Dieser Zugriffsplan ist nicht so aufwendig wie der Zugriffsplan aus dem vorherigen Beispiel. Der kumulative Aufwand wurde von etwa 753 Timerons in Abfrage 3 auf etwa 289 Timerons in Abfrage 4 reduziert.

Weitere Schritte

Verbesserung der Leistung Ihrer eigenen SQL- oder XQuery-Anweisungen.

Detaillierte Informationen über zusätzliche Schritte zur Verbesserung der Leistung finden Sie im *DB2 Information Center*. Danach können Sie wieder zu Visual Explain zurückkehren, um die Auswirkungen Ihrer Maßnahmen zu verwerten.

Kapitel 19. Umverteilung von Daten

Die *Datenumverteilung* ist eine Datenbankverwaltungsoperation, die hauptsächlich zum Versetzen von Daten innerhalb einer Umgebung mit partitionierten Datenbanken ausgeführt werden kann, wenn Partitionen hinzugefügt oder entfernt werden. Ziel dieser Operation ist in der Regel eine gleichmäßige Belegung des Speicherbereichs, eine Verbesserung der Datenbanksystemleistung oder die Erfüllung anderer Systemanforderungen.

Die Datenumverteilung kann mithilfe einer der folgenden Schnittstellen ausgeführt werden:

- Befehl **REDISTRIBUTE DATABASE PARTITION GROUP**
- Integrierte Prozedur ADMIN_CMD
- Integrierte Prozedur STEPWISE_REDISTRIBUTE_DBPG
- API sqludrdt

Für die Datenumverteilung innerhalb einer partitionierten Datenbank gibt es die folgenden Gründe:

- Neuausgleich der Daten, wenn eine neue Datenbankpartition der Datenbankumgebung hinzugefügt bzw. eine vorhandene Datenbankpartition daraus entfernt wird.
- Einleitung einer partitionsübergreifenden Verteilung von benutzerspezifischen Daten.
- Schutz für sensible Daten, indem sie innerhalb einer bestimmten Partition isoliert werden.

Die Datenumverteilung wird ausgeführt, indem eine Verbindung zu einer Datenbank auf der Katalogdatenbankpartition hergestellt und eine Datenumverteilungsoperation für eine bestimmte Partitionsgruppe mithilfe einer der unterstützten Schnittstellen gestartet wird. Für die Umverteilung von Daten ist das Vorhandensein von Verteilungsschlüsseldefinitionen für die Tabellen innerhalb der Partitionsgruppe erforderlich. Der Verteilungsschlüsselwert für eine Datenzeile innerhalb der Tabelle wird verwendet, um festzustellen, auf welcher Partition die Datenzeile gespeichert wird. Beim Erstellen einer Tabelle in einer Partitionsgruppe einer Mehrpartitionsdatenbank wird automatisch ein Verteilungsschlüssel generiert. Außerdem kann ein solcher Schlüssel mithilfe der Anweisung CREATE TABLE oder ALTER TABLE explizit definiert werden. Standardmäßig werden die Tabellendaten während der Datenumverteilung für jede Tabelle innerhalb einer angegebenen Datenbankpartitionsgruppe aufgeteilt und gleichmäßig auf die Datenbankpartitionen umverteilt. Eine andere Verteilung, wie z. B. eine ungleichmäßige Verteilung, kann aber erzielt werden, indem eine Eingabeverteilungszuordnung angegeben wird, die die Verteilung der Daten definiert. Verteilungszuordnungen können während einer Datenumverteilungsoperation für den zukünftigen Gebrauch generiert bzw. manuell erstellt werden.

Vergleich von protokollierter, wiederherstellbarer Umverteilung und minimal protokollierter, nicht aktualisierend wiederhergestellbarer Umverteilung

Wenn Sie mithilfe des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** oder der integrierten Prozedur ADMIN_CMD eine Datenumverteilung durchführen, haben Sie die Wahl zwischen zwei Methoden für die Datenumverteilung: protokollierte, wiederherstellbare Umverteilung und minimal protokollierte, nicht aktualisierend wiederherstellbare Umverteilung. Die letztere Methode wird durch den Befehlsparameter **NOT ROLLFORWARD RECOVERABLE** angegeben.

Für die Datenumverteilung in Kapazitätswachstumsszenarios, beim Lastausgleich oder bei der Leistungsverbesserung kann kostbare Wartungszeit, eine beträchtliche Menge an Planungszeit sowie kostenintensiver Protokollspeicherplatz und zusätzlicher Containerspeicherplatz erforderlich sein. Welche Umverteilungsmethode Sie auswählen, ist davon abhängig, ob Sie der Wiederherstellbarkeit oder der Geschwindigkeit den Vorrang geben:

- Bei der protokollierten, wiederherstellbaren Umverteilung wird eine umfassende Protokollierung aller Zeilenbewegungen durchgeführt, sodass die Datenbank später bei Unterbrechungen, bei Fehlern oder in sonstigen geschäftsrelevanten Bedarfssituationen wiederhergestellt werden kann.
- Die nicht aktualisierend wiederhergestellbare Umverteilung bietet eine verbesserte Leistung, da die Daten in Masseladeoperationen versetzt werden und für Einfüge- und Löschoperationen keine Protokolleinträge mehr erforderlich sind.

Die zweite Methode ist besonders nützlich, wenn Sie durch umfangreiche Anforderungen in Bezug auf den Speicherbereich für aktive Protokolle und Speicherbedarf in der Vergangenheit gezwungen wurden, eine einzige Datenumverteilungsoperation in mehrere kleinere Umverteilungstasks aufzuteilen, was möglicherweise sogar zu einem größeren Zeitaufwand für die Ausführung der Umverteilungsoperation insgesamt geführt hat.

Die nicht aktualisierend wiederhergestellbare Umverteilung ist für die meisten Situationen die empfohlene Methode, da die Datenumverteilung weniger Zeit in Anspruch nimmt, weniger fehlerträchtig ist und weniger Systemressourcen verbraucht. Dies führt dazu, dass der Gesamtaufwand für die Durchführung einer Datenumverteilung reduziert wird, was wiederum Betriebszeit und Ressourcen für andere Geschäftsoperationen freisetzt.

Minimal protokollierte, nicht aktualisierend wiederhergestellbare Umverteilung

Wird der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** abgesetzt und ist der Parameter **NOT ROLLFORWARD RECOVERABLE** angegeben, wird eine Strategie zur minimalen Protokollierung angewendet, mit der der Aufwand für das Schreiben von Protokolleinträgen für die einzelnen verschobenen Zeilen auf ein Mindestmaß reduziert wird. Dieser Protokollierungstyp ist für die Nutzbarkeit der Verteilungsoperation wichtig, da für eine Lösung, bei der jeder Versetzungsvorgang von Daten vollständig protokolliert wird, für Großsysteme möglicherweise eine unmögliche Menge von Speicherbereich für aktive sowie permanente Protokolle erforderlich ist und eine solche Lösung im Allgemeinen schlechtere Leistungsmerkmale aufweisen würde.

Außerdem gibt es Features und optionale Parameter, die nur bei der Auswahl der nicht aktualisierend wiederhergestellbaren Umverteilung verfügbar sind. Beispiel:

Diese Umverteilungsmethode stellt die Datenbank standardmäßig in den Wartemodus und führt eine Vorabprüfung durch, um sicherzustellen, dass alle Voraussetzungen erfüllt sind. Sie können außerdem optional angeben, dass im Rahmen der Umverteilungsoperation Indizes erneut erstellt und Daten zur Tabellenstatistik erfasst werden. Die Kombination und Automatisierung dieser ansonsten manuellen Tasks führt zu einer geringeren Fehleranfälligkeit, einer schnelleren Verarbeitung sowie einer höheren Effizienz und gibt Ihnen mehr Kontrolle über die Operationen.

Bei der nicht aktualisierend wiederhergestellbaren Umverteilung werden die Tabellen automatisch reorganisiert, was zur Freigabe von Plattenspeicher führen kann. Diese Reorganisation der Tabelle erfordert keinen Leistungsaufwand zusätzlich zur Umverteilungsoperation. Bei Tabellen mit Clusterindizes versucht die Reorganisation nicht, das Clustering beizubehalten. Wenn ein perfektes Clustering gewünscht wird, muss für Tabellen mit einem Clusterindex nach Abschluss der Datenumverteilung der Befehl **REORG TABLE** ausgeführt werden. Bei MDC-Tabellen behält die Reorganisation das Clustering der Tabelle bei und gibt nicht genutzte Blöcke zur Wiederverwendung frei; die Gesamtgröße der Tabelle erscheint jedoch nach der Umverteilung unverändert.

Anmerkung: Das Sichern jedes betroffenen Tabellenbereichs oder der gesamten Datenbank nach Abschluss der Umverteilungsoperation ist kritisch, da eine aktualisierende Wiederherstellung dieses Typs von Umverteilungsoperation dazu führt, dass alle Tabellen, die umverteilt wurden, als ungültig markiert werden. Solche Tabellen können nur gelöscht werden, was bedeutet, dass es keine Möglichkeit gibt, die Daten in diesen Tabellen wiederherzustellen. Dies ist der Grund dafür, warum das Dienstprogramm **REDISTRIBUTE DATABASE PARTITION GROUP** für wiederherstellbare Datenbanken beim Absetzen mit der Option **NOT ROLLFORWARD RECOVERABLE** alle Tabellenbereiche, mit denen es in Berührung kommt, in den Status BACKUP PENDING versetzt. Dieser Status zwingt Sie am Ende einer erfolgreichen Umverteilungsoperation zur Durchführung eines Backups für alle umverteilten Tabellenbereiche. Mit der Erstellung eines Backups nach der Umverteilungsoperation sollte kein Bedarf einer aktualisierenden Recovery für die Umverteilungsoperation selbst bestehen.

Das Fehlen einer Möglichkeit zur aktualisierenden Recovery hat eine wichtige Konsequenz zur Folge: Wenn Sie sich für das Zulassen von Aktualisierungen für Tabellen in der Datenbank während der Umverteilungsoperation entscheiden (auch Tabellen außerhalb der Datenbankpartitionsgruppe, für die eine Umverteilung durchgeführt wurde), einschließlich des Zeitraums am Ende der Umverteilung, bei dem die von der Umverteilung betroffenen Tabellenbereiche gesichert werden, kann es zum Verlust solcher Aktualisierungen im Fall einer schwerwiegenden Störung kommen, wenn z. B. ein Datenbankcontainer zerstört wird. Der Grund dafür, warum solche Aktualisierungen verloren gehen können, ist, dass die Umverteilungsoperation nicht aktualisierend wiederhergestellbar ist. Wenn eine Wiederherstellung der Datenbank mit dem vor der Umverteilung durchgeführten Backup erforderlich ist, ist es nicht möglich, eine aktualisierende Recovery für die Protokolle durchzuführen, um die Aktualisierungen, die während der Umverteilung durchgeführt wurden, zu wiederholen, ohne auch eine aktualisierende Recovery für die Umverteilung durchzuführen, die, wie bereits zuvor beschrieben, die umverteilten Tabellen im Status UNAVAILABLE belässt. Daher ist die einzige Möglichkeit in dieser Situation die Wiederherstellung der Datenbank mit dem vor der Umverteilung ohne aktualisierende Recovery durchgeführten Backup. Die Umverteilungsoperation kann anschließend erneut durchgeführt werden. Leider gehen sämtliche Aktualisierungen verloren, zu denen es bei der ursprünglichen Umverteilungsoperation kam.

Dem Stellenwert dieses Punkts kann nicht genügend Nachdruck verliehen werden. Um sicherzustellen, dass bei einer Umverteilungsoperation keine Aktualisierungen verloren gehen, muss eine der folgenden Bedingungen erfüllt sein:

- Es muss vermieden werden, dass Aktualisierungen während der Operation des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** vorgenommen werden, einschließlich des Zeitraums nach Ausführung des Befehls, in dem für die betroffenen Tabellenbereiche ein Backup durchgeführt wird.
- Die Umverteilungsoperation wird mit dem auf YES gesetzten Befehlsparameter **QUIESCE DATABASE** durchgeführt. Sie müssen weiterhin sicherstellen, dass sämtliche Anwendungen oder Benutzer, die auf die Datenbank im Wartemodus zugreifen dürfen, keine Aktualisierungen vornehmen.
- Aktualisierungen, die während der Umverteilungsoperation angewendet werden, stammen von einer reproduzierbaren Quelle; dies bedeutet, dass eine erneute Anwendung jederzeit möglich ist. Beispiel: Wenn es sich bei der Quelle von Aktualisierungen um Daten handelt, die in einer Datei gespeichert werden, und die Aktualisierungen während einer Stapelverarbeitung angewendet werden, gehen die Aktualisierungen offenkundig selbst bei einem Fehler, für den ein Datenbankrestore erforderlich ist, nicht verloren, da sie jederzeit einfach wieder angewendet werden können.

Was das Zulassen von Aktualisierungen für die Datenbank bei der Umverteilung betrifft, müssen Sie darüber entscheiden, ob solche Aktualisierungen geeignet sind oder nicht; die Grundlage hierfür bildet die Tatsache, ob die Aktualisierungen nach einem Datenbankrestore wiederholt werden können, falls dies erforderlich sein sollte.

Anmerkung: Nicht jede Störung bei der Operation des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** führt zu diesem Problem. Im Grunde tun dies die wenigsten. Der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** ist vollständig wieder anlauffähig; dies bedeutet, dass wenn das Dienstprogramm bei seiner Ausführung ausfällt, es mithilfe der Option **CONTINUE** einfach fortgesetzt oder mit der Option **ABORT** einfach abgebrochen werden kann. Bei den zuvor genannten Störungen handelt es sich um Störungen, bei denen der Benutzer Daten mit dem vor der Umverteilungsoperation durchgeführten Backup wiederherstellen muss.

Protokollierte, wiederherstellbare Umverteilung

Als ursprüngliche und standardmäßige Version des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** werden mit dieser Methode Daten mithilfe von SQL-Standardeinfüge- und -Löschoperationen umverteilt. Eine umfangreiche Protokollierung aller Zeilenbewegungen wird durchgeführt, sodass die Datenbank wiederherstellbar ist; dabei wird für sie mithilfe des Befehls **RESTORE DATABASE** ein Restore durchgeführt, anschließend wird für alle Änderungen mithilfe des Befehls **ROLLFORWARD DATABASE** eine aktualisierende Recovery vorgenommen.

Nach der Datenumverteilung enthält die Quellentabelle leere Speicherbereiche, da Zeilen gelöscht und an neue Datenbankpartitionen gesendet wurden. Wenn Sie die leeren Speicherbereiche freigeben möchten, müssen Sie die Tabellen reorganisieren. Für die Reorganisation der Tabellen müssen Sie nach Abschluss der Umverteilung eine separate Operation durchführen. Um die Leistung dieser Methode zu verbessern, müssen Sie die Indizes löschen und sie nach Abschluss der Umverteilung erneut erstellen.

Voraussetzungen für die Datenumverteilung

Bevor die Datenumverteilung für eine Reihe von Tabellen innerhalb einer Datenbankpartitionsgruppe erfolgreich ausgeführt werden kann, müssen bestimmte Voraussetzungen erfüllt werden.

Nachfolgend ist eine Liste der obligatorischen Voraussetzungen aufgeführt:

- Die Berechtigung zum Ausführen der Datenumverteilung über die unterstützte Datenumverteilungsschnittstelle Ihrer Wahl.
- Eine beträchtliche Menge an Zeit während eines Zeitraums mit geringer Systemaktivität, in dem die Umverteilungsoperation ausgeführt werden soll.
- Alle Tabellen mit Daten, die als Teil einer Datenumverteilungsoperation umverteilt werden sollen, müssen den Status NORMAL aufweisen. Tabellen dürfen sich z. B. nicht im Status LOAD PENDING oder in einem anderen Ladestatus befinden, der einen Zugriff unmöglich macht. Sie können den Status von Tabellen überprüfen, indem Sie eine Verbindung zu jeder Partition in der Datenbankpartitionsgruppe herstellen und den Befehl **LOAD QUERY** absetzen. Die Ausgabe dieses Befehls enthält Informationen zum Status der Tabelle. In der Dokumentation des Befehls **LOAD QUERY** wird die Bedeutung jedes Tabellenstatus erläutert, außerdem erfahren Sie dort, wie Sie Tabellen von einem Status in einen anderen versetzen.
- Alle Tabellen innerhalb der Datenbankpartition, die umverteilt wird, müssen mit einem Verteilungsschlüssel definiert sein. Wird eine neue Datenbankpartition zu einem System mit einer einzelnen Partition hinzugefügt, kann die Datenumverteilung erst ausgeführt werden, wenn alle Tabellen innerhalb der Partitionen über einen Verteilungsschlüssel verfügen. Bei Tabellen, die mit der Anweisung CREATE TABLE erstellt wurden und über Definitionen ohne Verteilungsschlüssel verfügen, müssen Sie die Tabelle mit der Anweisung ALTER TABLE ändern, um einen Verteilungsschlüssel vor der Datenumverteilung hinzuzufügen.
- Replizierte MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle), die in einer Datenbankpartitionsgruppe enthalten sind, müssen gelöscht werden, bevor Sie die Daten umverteilen. Speichern Sie eine Kopie der MQT-Definitionen, sodass sie nach Beendigung der Datenumverteilung erneut erstellt werden können.
- Wird eine ungleichmäßige Umverteilung gewünscht, muss eine Verteilungszuordnung als Zielverteilungszuordnung erstellt werden, die als Parameter für die Umverteilungsschnittstelle verwendet werden soll.
- Ein Backup der Datenbank muss mithilfe des Befehls **BACKUP DATABASE** erstellt werden. Dieses Backup gehört nicht zu den obligatorischen Voraussetzungen, es wird jedoch dringend dazu geraten.
- Es muss eine Verbindung zur Datenbank von der Katalogdatenbankpartition hergestellt werden.
- Es muss ausreichend Speicherplatz verfügbar sein, um alle Indizes entweder während oder nach der Datenumverteilung erneut zu erstellen. Der Befehlsparameter **INDEXING MODE** hat Auswirkungen, wenn die Indizes erneut erstellt werden.
- Wenn der Befehlsparameter **NOT ROLLFORWARD RECOVERABLE** angegeben wird, sollte ausreichend Speicherplatz für Steuerdateien, die vom IBM Service zur Fehlerbestimmung verwendet werden, zum Schreiben der Statusinformationen verfügbar sein. Die Steuerdateien werden in den folgenden Pfaden generiert und sollten nach Abschluss der Datenumverteilungsoperation manuell gelöscht werden:

– Linux- und UNIX-Betriebssysteme: **diagpath**/redist/*db-name*/*db-partitionsgruppenname*/*zeitmarke*/
– Windows-Betriebssysteme: **diagpath**\redist*db-name**db-partitionsgruppenname**zeitmarke*\

Sie können den Platzbedarf in Byte für die Steuerdateien anhand der folgenden Formel berechnen:

```
(Anzahl Seiten für alle Tabellen in der Datenbankpartitionsgruppe) * 64 Byte
+ (Anzahl LOB-Werte in der Datenbankpartitionsgruppe) * 600 Byte
```

Zur Schätzung der *Anzahl LOB-Werte in der Datenbankpartitionsgruppe* addieren Sie die Anzahl der LOB-Spalten in Ihren Tabellen und multiplizieren diesen Wert mit der Anzahl der Zeilen in der größten Tabelle.

- Wenn der Befehlsparameter **NOT ROLLFORWARD RECOVERABLE nicht** angegeben ist, muss ausreichend Protokolldateispeicher verfügbar sein, um die Protokolleinträge aufzunehmen, die zu den INSERT- und DELETE-Operationen gehören, die während der Datenumverteilung ausgeführt werden. Andernfalls wird die Datenumverteilung unterbrochen oder sie schlägt fehl.

Der Wert des Datenbankkonfigurationsparameters **util_heap_sz** ist für die Verarbeitung der Datenversetzung zwischen Datenbankpartitionen von kritischer Bedeutung. Ordnen Sie durch den Parameter **util_heap_sz** eine möglichst hohe Speicherkapazität für die Dauer der Umverteilungsoperation zu. Außerdem muss der Wert für den Parameter **sortheap** (= Sortierspeichergröße) ausreichend bemessen sein, falls im Rahmen der Umverteilungsoperation Indizes neu erstellt werden. Setzen Sie den Wert der Datenbankkonfigurationsparameter **util_heap_sz** und **sortheap** wie benötigt herauf, um das Leistungsverhalten bei der Umverteilung zu verbessern.

Einschränkungen bei der Datenumverteilung

Die Einschränkungen bei der Datenumverteilung sollten Sie unbedingt beachten, bevor Sie mit der Datenumverteilung fortfahren bzw. wenn Sie Fehler hinsichtlich der Datenumverteilung beheben.

Die folgenden Einschränkungen gelten für die Datenumverteilung:

- Die Datenumverteilung auf Partitionen, auf denen Tabellen keine Partitionierungsschlüsseldefinitionen haben, ist nicht zulässig.
- Bei laufender Datenumverteilung gilt Folgendes:
  – Das Starten einer weiteren Umverteilungsoperation für dieselbe Datenbankpartitionsgruppe ist nicht zulässig.
  – Das Löschen der Datenbankpartitionsgruppe ist nicht zulässig.
  – Das Ändern der Datenbankpartitionsgruppe ist nicht zulässig.
  – Das Ausführen einer Anweisung ALTER TABLE für eine Tabelle in der Datenbankpartitionsgruppe ist nicht zulässig.
  – Das Erstellen neuer Indizes in der Tabelle, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.
  – Das Löschen von Indizes, die in der Tabelle definiert sind, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.
  – Das Abfragen von Daten in der Tabelle, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.
  – Das Aktualisieren der Tabelle, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.

- Das Aktualisieren von Tabellen in einer Datenbank, für die eine Datenumverteilung durchgeführt wird, die mit dem Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** und unter Angabe des Befehlsparameters **NOT ROLLFORWARD RECOVERABLE** gestartet wurde, ist nicht zulässig. Obwohl die Aktualisierungen durchgeführt werden können, könnten die an den Daten vorgenommenen Änderungen verloren gehen, falls die Datenumverteilung unterbrochen wird. Aus diesem Grund wird hiervon abgeraten.
- Wird der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** abgesetzt und ist der Befehlsparameter **NOT ROLLFORWARD RECOVERABLE** angegeben, gilt Folgendes:
  - Änderungen bei der Datenverteilung, die während der Umverteilung auftreten, sind nicht aktualisierend wiederherstellbar.
  - Wenn die Datenbank wiederherstellbar ist, wird der Tabellenbereich in den Status BACKUP PENDING versetzt, nachdem auf die erste Tabelle innerhalb des Tabellenbereichs zugegriffen wurde. Wenn Sie diesen Status der Tabelle ändern wollen, müssen Sie ein Backup der Tabellenbereichsänderungen durchführen, nachdem die Umverteilungsoperation beendet wurde.
  - Während der Datenumverteilung können die Daten aus den Tabellen der Datenbankpartitionsgruppe, die gerade umverteilt wird, nicht aktualisiert werden; die Daten sind schreibgeschützt. Auf Tabellen, die gerade aktiv umverteilt werden, kann nicht zugegriffen werden.
- Wenn bei typisierten Tabellen (Hierarchietabellen) der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** verwendet wird und der Parameter **TABLE** mit dem Wert ONLY angegeben ist, dann ist der Tabellenname ausschließlich auf den Namen der Stammtabelle beschränkt. Namen untergeordneter Tabellen können nicht angegeben werden.
- Die Datenumverteilung wird zum Versetzen von Daten zwischen Datenbankpartitionen unterstützt. Bei partitionierten Tabellen ist das Versetzen von Daten zwischen Bereichen einer datenpartitionierten Tabelle jedoch nicht zulässig, es sei denn, die folgenden Bedingungen treffen zu:
  - Die partitionierte Tabelle hat den Zugriffsmodus FULL ACCESS (Vollzugriff) in der Katalogtabelle SYSTABLES.ACCESS_MODE.
  - Die partitionierte Tabelle hat keine Partitionen, die gerade zugeordnet werden bzw. deren Zuordnung gerade aufgehoben wird.
- Bei replizierten MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle) müssen Sie, wenn die Daten in einer Datenbankpartitionsgruppe replizierte MQTs enthalten, diese Tabellen löschen, bevor Sie die Daten umverteilen. Nach der Umverteilung der Daten können Sie die MQTs erneut erstellen.
- Bei Datenbankpartitionen, die MDC-Tabellen (MDC - mehrdimensionales Clustering) enthalten, ist die Verwendung des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** nicht zulässig und der Befehl wird nicht erfolgreich ausgeführt, wenn MDC-Tabellen in der Datenbankpartitionsgruppe vorhanden sind, die durch ein Rollout gelöschte Blöcke enthalten, für die eine Bereinigung ansteht. Diese MDC-Tabellen müssen bereinigt werden, bevor die Datenumverteilung wieder aufgenommen bzw. erneut gestartet werden kann.
- Das Löschen von Tabellen, die derzeit in den DB2-Katalogsichten mit dem Status "Umverteilung läuft" markiert sind, ist nicht zulässig. Zum Löschen einer Tabelle in diesem Status führen Sie zunächst den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** mit dem Parameter **ABORT** oder **CONTINUE** und einer entsprechenden Tabellenliste aus, sodass die Umverteilung der Tabelle entweder abgeschlossen oder abgebrochen wird.

Ermitteln, ob Datenumverteilung erforderlich ist

Das Feststellen der aktuellen Datenverteilung für eine Datenbankpartitionsgruppe oder Tabelle kann hilfreich sein, wenn ermittelt werden soll, ob eine Datenumverteilung erforderlich ist. Die Datenverteilung kann zum Erstellen einer angepassten Verteilungszuordnung verwendet werden, mit der angegeben werden kann, wie die Daten verteilt werden sollen.

Informationen zu diesem Vorgang

Wenn eine neue Datenbankpartition zur Datenbankpartitionsgruppe hinzugefügt wird oder eine vorhandene Datenbankpartition aus einer Datenbankpartitionsgruppe gelöscht wird, führen Sie eine Datenumverteilung aus, damit die Daten auf allen Datenbankpartitionen gleichmäßig verteilt sind.

Wenn keine Datenbankpartitionen hinzugefügt oder aus einer Datenbankpartitionsgruppe gelöscht wurden, empfiehlt sich eine Datenumverteilung üblicherweise nur dann, wenn die Daten auf den Datenbankpartitionen der Datenbankpartitionsgruppe ungleichmäßig verteilt sind. Beachten Sie, dass in einigen Fällen eine ungleichmäßige Verteilung von Daten durchaus erwünscht sein kann. Wenn sich z. B. einige Datenbankpartitionen auf einer leistungsfähigen Maschine befinden, kann es für diese Datenbankpartitionen vorteilhaft sein, im Vergleich zu anderen Datenbankpartitionen größere Datenvolumina zu enthalten.

Vorgehensweise

Um zu ermitteln, ob eine Datenumverteilung notwenig ist, gehen Sie folgendermaßen vor:

1. Rufen Sie Informationen zur aktuellen Verteilung der Daten auf die Datenbankpartitionen der Datenbankpartitionsgruppe ab.

   Führen Sie die folgende Abfrage für die größte Tabelle (oder alternativ eine repräsentative Tabelle) in der Datenbankpartitionsgruppe aus:

   ```
   SELECT DBPARTITIONNUM(tabellenname), COUNT(*) FROM tabellenname
         GROUP BY DBPARTITIONNUM(spaltenname)
         ORDER BY DBPARTITIONNUM(spaltenname) DESC
   ```

   Hier ist *spaltenname* der Name des Verteilungsschlüssels für die Tabelle *tabellenname*.

   Die Ausgabe dieser Abfrage zeigt, wie viele Datensätze aus der Tabelle *tabellenname* sich auf jeder Datenbankpartition befinden. Wenn die Verteilung der Daten auf den Datenbankpartitionen nicht Ihren Wünschen entspricht, fahren Sie mit dem nächsten Schritt fort.

2. Rufen Sie Informationen zur Verteilung der Daten auf Hashpartitionen ab.

   Führen Sie die folgende Abfrage mit denselben Werten für *spaltenname* und *tabellenname* aus, die im vorherigen Schritt verwendet wurden:

   ```
   SELECT HASHEDVALUE(spaltenname), COUNT(*) FROM tabellenname
         GROUP BY HASHEDVALUE(spaltenname)
         ORDER BY HASHEDVALUE(spaltenname) DESC
   ```

   Die Ausgabe dieser Abfrage kann problemlos verwendet werden, um die benötigte Verteilungsdatei zu erstellen, wenn der Parameter **USING DISTFILE** im Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** angegeben ist. Eine Beschreibung des Formats der Verteilungsdatei finden Sie in der Befehlsreferenz zu **REDISTRIBUTE DATABASE PARTITION GROUP**.

3. Optional: Wenn die Daten eine Umverteilung erfordern, können Sie diese Operation während einer Systemwartung durchführen.

   Wenn der Parameter **USING DISTFILE** angegeben ist, verwendet der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** die Informationen in der Datei, um eine neue Partitionszuordnung für die Datenbankpartitionsgruppe zu generieren. Dies hat eine gleichmäßige Verteilung von Daten auf den Datenbankpartitionen zur Folge.

   Wenn eine gleichmäßige Verteilung nicht gewünscht wird, können Sie Ihre eigene Zielpartitionszuordnung für die Umverteilungsoperation erstellen. Die Zielpartitionszuordnung kann mithilfe des Parameters **USING TARGETMAP** im Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** angegeben werden.

Ergebnisse

Nach dieser Untersuchung sind Sie darüber im Bilde, ob Ihre Daten gleichmäßig oder ungleichmäßig verteilt sind und ob eine Datenumverteilung erforderlich ist.

Umverteilen von Daten auf Datenbankpartitionen unter Verwendung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP

Der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** ist die empfohlene Schnittstelle für das Ausführen einer Datenumverteilung.

Vorgehensweise

Gehen Sie wie folgt vor, um Daten auf Datenbankpartitionen in einer Datenbankpartitionsgruppe umzuverteilen:

1. Optional: Führen Sie ein Backup der Datenbank durch. Siehe den Befehl **BACKUP DATABASE**.

   Vor dem Durchführen einer Datenumverteilung, die nicht aktualisierend wiederherstellbar ist, sollten Sie unbedingt eine Backup-Kopie der Datenbank erstellen.
2. Stellen Sie eine Verbindung zu der Datenbankpartition her, in der die Systemkatalogtabellen enthalten sind. Siehe die Anweisung CONNECT.
3. Setzen Sie den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** ab.

   Anmerkung: In früheren Versionen des DB2-Datenbankprodukts arbeitete dieser Befehl mit dem Schlüsselwort **NODEGROUP** anstelle der Schlüsselwörter **DATABASE PARTITION GROUP**.

   Geben Sie die folgenden Argumente an:

   Name einer Datenbankpartitionsgruppe
   Sie müssen die Datenbankpartitionsgruppe angeben, in der die Daten umzuverteilen sind.

   UNIFORM
   OPTIONAL: Gibt an, dass Daten gleichmäßig verteilt werden sollen. **UNIFORM** ist die Standardeinstellung, wenn kein Verteilungstyp angegeben ist. Daher ist es zulässig, diese Option zu übergehen, wenn kein anderer Verteilungstyp angegeben wurde.

   USING DISTFILE *verteilungsdateiname*
   OPTIONAL: Gibt an, dass eine angepasste Verteilung erwünscht ist, und gibt ferner den Dateipfadnamen einer Verteilungsdatei an, in der

Daten enthalten sind, die die gewünschte ungleiche Datenverteilung definieren. Der Inhalt dieser Datei wird zum Generieren einer Zielverteilungszuordnung verwendet.

USING TARGETMAP *zielzuordnungsname*
OPTIONAL: Gibt an, dass eine Zieldatenumverteilungszuordnung verwendet werden soll, und gibt ferner den Namen der Datei an, in der die Zielumverteilungszuordnung enthalten ist.

Weitere Einzelheiten enthalten die Informationen zum Befehlszeilendienstprogramm **REDISTRIBUTE DATABASE PARTITION GROUP**.

4. Ermöglichen Sie dem Befehl eine ununterbrochene Ausführung. Wenn nach Beendigung des Befehls die Datenumverteilung erfolgreich fortgesetzt wurde, führen Sie die folgenden Aktionen aus:
   - Erstellen Sie ein Backup aller Tabellenbereiche in der Datenbankpartitionsgruppe, die sich im Status "Backup anstehend" befinden. Alternativ kann auch ein Datenbankgesamtbackup durchgeführt werden.

     Anmerkung: Tabellenbereiche werden nur dann in den Status BACKUP PENDING versetzt, wenn die Datenbank wiederherstellbar ist und der Befehlsparameter **NOT ROLLFORWARD RECOVERABLE** des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** verwendet wird.
   - Erstellen Sie alle replizierten MQTs erneut, die Sie vor der Umverteilung gelöscht haben.
   - Führen Sie den Befehl **RUNSTATS** aus, wenn die folgenden Bedingungen erfüllt sind:
     – Der Befehlsparameter **STATISTICS NONE** wurde im Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** angegeben, oder der Befehlsparameter **NOT ROLLFORWARD RECOVERABLE** wurde übergangen. Diese beiden Bedingungen zeigen an, dass die Statistikdaten während der Datenumverteilung nicht erfasst wurden.
     – In der Datenbankpartitionsgruppe gibt es Tabellen, die über ein Statistikprofil verfügen.

     Der Befehl **RUNSTATS** erfasst Statistikdaten für die Datenverteilung, die der SQL-Compiler bzw. das SQL-Optimierungsprogramm bei der Auswahl von Datenzugriffsplänen für Abfragen verwenden soll.
   - Wenn der Befehlsparameter **NOT ROLLFORWARD RECOVERABLE** angegeben wurde, löschen Sie die Steuerdateien, die sich in den folgenden Pfaden befinden:
     – Linux- und UNIX-Betriebssysteme: **diagpath**/redist/*db-name*/*db-partitionsgruppenname*/*zeitmarke*/
     – Windows-Betriebssysteme: **diagpath**\redist*db-name**db-partitionsgruppenname**zeitmarke*\

Ergebnisse

Die Datenumverteilung ist erfolgreich abgeschlossen, und die Informationen zum Datenumverteilungsprozess stehen in der Umverteilungsprotokolldatei zur Verfügung. Informationen zur verwendeten Verteilungszuordnung befinden sich in den DB2-EXPLAIN-Tabellen.

Umverteilen von Daten in einer Datenbankpartitionsgruppe

Zur Erstellung eines effektiven Umverteilungsplans für Ihre Datenbankpartitionsgruppe und zur Umverteilung von Daten müssen Sie den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** absetzen oder die API sqludrdt aufrufen.

Vorbereitende Schritte

Um mit Datenbankpartitionsgruppen zu arbeiten, müssen Sie über die Berechtigung SYSADM, SYSCTRL oder DBADM verfügen.

Vorgehensweise

Gehen Sie wie folgt vor, um Daten in einer Datenbankpartitionsgruppe umzuverteilen:

- Setzen Sie den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** im Befehlszeilenprozessor ab.
- Setzen Sie den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** mithilfe der Prozedur ADMIN_CMD ab.
- Rufen Sie die API sqludrdt auf.

Protokollspeicherbedarf für die Datenumverteilung

Für die erfolgreiche Ausführung einer Datenumverteilungsoperation muss ausreichend Protokolldateispeicher zugewiesen werden, um sicherzustellen, dass die Datenumverteilung nicht unterbrochen wird. Anforderungen an den Speicherbereich sind weniger wichtig, wenn Sie den Befehlsparameter **NOT ROLLFORWARD RECOVERABLE** angeben, da während dieser Art der Datenumverteilung kaum protokolliert wird.

Die Menge des erforderlichen Protokolldateispeichers hängt von mehreren Faktoren ab, beispielsweise welche Optionen des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** verwendet werden.

Wenn die Umverteilung von einer beliebigen unterstützten Schnittstelle ausgeführt wird, für die die Datenumverteilung aktualisierend wiederherstellbar ist, gilt Folgendes:

- Das Protokoll muss groß genug sein, um die INSERT- und DELETE-Operationen auf jeder Datenbankpartition unterzubringen, auf der Daten umverteilt werden. Die intensivsten Protokollierungsanforderungen werden an die Datenbankpartition gestellt, der die meisten Daten verloren gehen, bzw. an die Datenbankpartition, die die meisten Daten erhält.
- Wenn Sie die Anzahl der Datenbankpartitionen erhöhen, verwenden Sie das Verhältnis zwischen der Anzahl der aktuellen Datenbankpartitionen und der neuen Anzahl der Datenbankpartitionen, um die Anzahl der INSERT- und DELETE-Operationen abzuschätzen. Betrachten Sie z. B. das Umverteilen von Daten, die vor der Umverteilung gleichmäßig verteilt sind. Wenn Sie die Anzahl der Datenbankpartitionen von vier auf fünf erhöhen, werden ungefähr zwanzig Prozent der vier ursprünglichen Datenbankpartitionen in die neue Datenbankpartition versetzt. Dies bedeutet, dass zwanzig Prozent der DELETE-Operationen in jeder der vier ursprünglichen Datenbankpartitionen auftreten und alle INSERT-Operationen in der neuen Datenbankpartition auftreten.
- Betrachten Sie eine ungleichmäßige Verteilung von Daten, wie dies z. B. der Fall ist, wenn der Verteilungsschlüssel viele Nullwerte (NULL) enthält. In diesem

Fall werden alle Zeilen, die einen NULL-Wert im Verteilungsschlüssel enthalten von einer Datenbankpartition unter dem alten Verteilungsschema in eine andere Datenbankpartition unter dem neuen Verteilungsschema versetzt. Dadurch erhöht sich die Menge des Protokollspeicherbereichs, der in den beiden Datenbankpartitionen benötigt wird, unter Umständen deutlich gegenüber der Menge, die für eine gleichmäßige Verteilung berechnet wurde.

- Die Umverteilung von jeder Tabelle ist eine einzelne Transaktion. Aus diesem Grund multiplizieren Sie bei der Berechnung des Protokollspeicherbereichs den Prozentsatz der Änderung, z. B. zwanzig Prozent, mit der Größe der größten Tabelle. Bedenken Sie aber, dass die größte Tabelle unter Umständen gleichmäßig verteilt ist, die zweitgrößte Tabelle aber z. B. über mindestens eine überdurchschnittlich große Datenbankpartition verfügen könnte. In solch einem Fall sollten Sie anstelle der größten Tabelle die Verwendung der ungleichmäßig verteilten Tabelle in Betracht ziehen.

Anmerkung: Nachdem Sie das maximale Volumen der Daten geschätzt haben, die in einer Datenbankpartition eingefügt und gelöscht werden sollen, verdoppeln Sie diesen Wert, um den Spitzenwert für die Größe der aktiven Protokolldatei zu ermitteln. Ist dieser geschätzte Wert größer als der Grenzwert von 1024 GB für die aktive Protokolldatei, dann muss die Datenumverteilung schrittweise durchgeführt werden. Verwenden Sie beispielsweise die Prozedur STEPWISE_REDISTRIBUTE_DBPG so, dass die verwendete Anzahl der Schritte proportional ist zum Größenunterschied zwischen Schätzwert und dem Grenzwert für die aktive Protokolldatei. Sie können außerdem den Datenbankkonfigurationsparameter **logsecond** auf -1 setzen, um die meisten Probleme bezüglich des Protokollspeicherbereichs zu vermeiden.

Wenn die Umverteilung von einer beliebigen unterstützten Schnittstelle ausgeführt wird, für die die Datenumverteilung nicht aktualisierend wiederherstellbar ist, gilt Folgendes:

- Protokollsätze werden nicht erstellt, wenn Zeilen als Teil der Datenumverteilung versetzt werden. Durch dieses Verhalten wird der Protokolldateispeicherbedarf deutlich reduziert. Wird diese Option jedoch verwendet, wenn eine aktualisierende Recovery der Datenbank durchgeführt wird, kann der Protokollsatz der Umverteilungsoperation nicht aktualisierend wiederhergestellt werden und alle Tabellen, die als Teil der aktualisierenden Recoveryoperation verarbeitet wurden, verbleiben im Status UNAVAILABLE (nicht verfügbar).
- Wenn die Datenbankpartitionsgruppe, für die eine Datenumverteilung durchgeführt wird, Tabellen mit Langfelddaten (LF-Daten) oder Daten großer Objekte (LOB-Daten) in den Tabellen enthält, ist die Anzahl der Protokollsätze, die während der Datenumverteilung generiert werden, höher, weil ein Protokollsatz für jede Datenzeile erstellt wird. In diesem Fall ist ein Protokollspeicherbedarf pro Datenbankpartition zu erwarten, der ungefähr zwei Drittel des Volumens der Daten beträgt, die in dieser Partition bewegt (d. h. gesendet und/oder empfangen) werden.

Ereignisprotokolldateien für die Umverteilung

Während der Datenumverteilung wird eine Ereignisprotokollierung durchgeführt. Die Ereignisdaten werden in Ereignisprotokolldateien protokolliert, die später für die Durchführung einer Fehlerbehebung verwendet werden können.

Wenn die Datenumverteilung ausgeführt wird, werden Informationen zu den einzelnen verarbeiteten Tabellen in einem Ereignisprotokolldateienpaar protokolliert. Die Ereignisprotokolldateien erhalten die Namen

datenbankname.name_der_datenbankpartitionsgruppe.zeitmarke.log und *datenbankname.name_der_datenbankpartitionsgruppe.zeitmarke*.

Die Protokolldateien befinden sich an folgenden Positionen:

- Im Verzeichnis *Ausgangsinstallationsverzeichnis*/sqllib/redist unter Linux- und UNIX-Betriebssystemen
- Im Verzeichnis *db2instprof*\instance\redist unter Windows-Betriebssystemen; dabei ist *db2instprof* der Wert der Registrierdatenbankvariablen **DB2INSTPROF**

Nachfolgend finden Sie ein Beispiel für die Namen der Ereignisprotokolldateien:

```
SAMPLE.IBMDEFAULTGROUP.2012012620240204
SAMPLE.IBMDEFAULTGROUP.2012012620240204.log
```

Diese Dateien sind für eine Umverteilungsoperation auf einer Datenbank bestimmt, die den Namen SAMPLE aufweist und auf der sich eine Datenbankpartitionsgruppe mit dem Namen IBMDEFAULTGROUP befindet. Die Dateien wurden am 26. Januar 2012 um 8.24 Uhr Ortszeit erstellt.

Die Ereignisprotokolldateien dienen im Wesentlichen folgenden drei Zwecken:

- Zur Bereitstellung allgemeiner Informationen zur Umverteilungsoperation, wie zum Beispiel die alte und die neue Verteilungszuordnung.
- Zur Bereitstellung von Informationen, mit deren Hilfe Benutzer ermitteln können, welche Tabellen vom Dienstprogramm bereits umverteilt wurden.
- Zur Bereitstellung von Informationen zu den einzelnen Tabellen, die umverteilt wurden. Diese Informationen enthalten den für die Tabelle verwendeten Indexierungsmodus, eine Angabe zum Erfolg der Tabellenumverteilung sowie den Anfangs- und Endzeitpunkt für die Umverteilungsoperation an der Tabelle.

Umverteilen von Datenbankpartitionsgruppen mithilfe der Prozedur STEPWISE_REDISTRIBUTE_DBPG

Die Datenumverteilung kann mithilfe von integrierten Prozeduren ausgeführt werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Datenbankpartitionsgruppe mit der Prozedur STEPWISE_REDISTRIBUTE_DBPG umzuverteilen:

1. Analysieren Sie die Datenbankpartitionsgruppe im Hinblick auf Protokollspeicherverfügbarkeit und ungleiche Datenverteilung unter Verwendung der Prozedur ANALYZE_LOG_SPACE.

   Die Prozedur ANALYZE_LOG_SPACE gibt eine Ergebnismenge (einen geöffneten Cursor) von Ergebnissen einer Speicherbereichsanalyse zurück, die Felder für jede Datenbankpartition der angegebenen Datenbankpartitionsgruppe enthalten.
2. Erstellen Sie eine Datenverteilungsdatei für eine bestimmte Tabelle unter Verwendung der Prozedur GENERATE_DISTFILE.

   Die Prozedur GENERATE_DISTFILE generiert eine Datenverteilungsdatei für die angegebene Tabelle und speichert sie unter dem angegebenen Dateinamen.
3. Erstellen und dokumentieren Sie den Inhalt eines schrittweise durchgeführten Umverteilungsplans für die Datenbankpartitionsgruppe unter Verwendung der Prozedur STEPWISE_REDISTRIBUTE_DBPG.

4. Erstellen Sie eine Datenverteilungsdatei für eine bestimmte Tabelle unter Verwendung der Prozeduren GET_SWRD_SETTINGS und SET_SWRD_SETTINGS.

   Die Prozedur GET_SWRD_SETTINGS liest die vorhandenen Umverteilungsregistrierdatensätze für die angegebene Datenbankpartitionsgruppe.

   Die Prozedur SET_SWRD_SETTINGS erstellt oder ändert die Umverteilungsregistrierdatenbank. Wenn die Registrierdatenbank nicht vorhanden ist, erstellt die Funktion sie und fügt ihr Datensätze hinzu. Wenn die Registrierdatenbank bereits vorhanden ist, stellt die Funktion über das Feld *overwriteSpec* fest, welche Feldwerte überschrieben werden müssen. Das Feld *overwriteSpec* gibt dieser Funktion die Möglichkeit, NULL-Eingaben für Felder zu akzeptieren, die nicht aktualisiert werden müssen.
5. Verteilen Sie die Datenbankpartitionsgruppe dem Plan entsprechend unter Verwendung der Prozedur STEPWISE_REDISTRIBUTE_DBPG um.

   Die Prozedur STEPWISE_REDISTRIBUTE_DBPG führt die Umverteilung eines Teils der Datenbankpartitionsgruppe entsprechend der Eingabe und der Einstellungsdatei durch.

Beispiel

Das folgende Beispiel zeigt ein Script für den Befehlszeilenprozessor (CLP) unter AIX:

```
# ------------------------------------------------------------------------------------
# Angabe der Datenbank für die zu erstellende Verbindung
# ------------------------------------------------------------------------------------
dbName="SAMPLE"

# ------------------------------------------------------------------------------------
# Angabe des Namens für die Zieldatenbankpartitionsgruppe
# ------------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# ------------------------------------------------------------------------------------
# Angabe des Schemas und des Namens der Tabelle
# ------------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# ------------------------------------------------------------------------------------
# Angabe des Namens der Datenverteilungsdatei
# ------------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
export DB2COMM=TCPIP

# ------------------------------------------------------------------------------------
# Aufrufen der Aufrufanweisungen im Befehlszeilenprozessor (CLP)
# ------------------------------------------------------------------------------------
db2start
db2 -v "connect to $dbName"

# ------------------------------------------------------------------------------------
# Analyse des Effekts einer hinzugefügten Datenbankpartition ohne Anwendung der
# Änderungen - Analyse eines hypothetischen Falls
#
# - Im folgenden Fall werden hypothetisch die Datenbankpartitionen '40,50,60' der
#   Datenbankpartitionsgruppe hinzugefügt; für die Datenbankpartitionen '10,20,30,40,50,60'
#   werden die Zielverhältnisse 1:2:1:2:1:2 verwendet.
#
# Hinweis: In diesem Beispiel sind nur die Partitionen 10, 20 und 30 tatsächlich in der
#          Datenbankpartitionsgruppe vorhanden.
```

```
# ------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"

# ------------------------------------------------------------------------------
# Analyse des Effekts einer gelöschten Datenbankpartition ohne Anwendung der Änderungen
#
# - Im folgenden Fall wird hypothetisch die Datenbankpartition 30 aus der Datenbank-
#   partitionsgruppe gelöscht und die Daten werden auf die Datenbankpartitionen 10 und 20
#   unter Verwendung eines Zielverhältnisses von 1 : 1 umverteilt.
#
# Hinweis: In diesem Beispiel sollten alle Datenbankpartitionen, 10, 20 und 30, in der
#          Datenbankpartitionsgruppe vorhanden sein.
# ------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ------------------------------------------------------------------------------
# Generieren einer Datenverteilungsdatei zur Verwendung durch den Umverteilungsprozess
# ------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ------------------------------------------------------------------------------
# Schreiben eines schrittweisen Umverteilungsplans in eine Registrierdatenbank
#
# Der Wert 1 für den zehnten Parameter kann bewirken, dass die zurzeit ausgeführte
# gespeicherte Prozedur zur schrittweisen Umverteilung den aktuellen Schritt beendet und
# die Verarbeitung stoppt, bis dieser Parameter auf 0 zurückgesetzt wird und die
# gespeicherte Prozedur zur Umverteilung erneut aufgerufen wird.
# ------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ------------------------------------------------------------------------------
# Berichten des Inhalts des schrittweisen Umverteilungsplans für die angegebene
# Datenbankpartitionsgruppe
# ------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ------------------------------------------------------------------------------
# Umverteilen der Daten der Datenbankpartitionsgruppe "dbpgName" entsprechend dem in
# der Registrierdatenbank durch set_swrd_settings gespeicherten Umverteilungsplan.
# Mit Start bei Schritt 3 werden die Daten umverteilt, bis zwei Schritte im
# Umverteilungsplan ausgeführt sind.
# ------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

Kapitel 20. Konfigurieren des Speichers mit automatischer Leistungsoptimierung

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 für partitionierte 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 Optimierungsentscheidungen in dieser Partition getroffen werden, werden die Speicheranpassungen an die anderen Datenbankpartitionen verteilt, um sicherzustellen, dass alle Datenbankpartitionen ähnliche Konfigurationen behalten.

Das auf einer Optimierungspartition basierende Modell geht davon aus, dass die Funktion nur verwendet wird, wenn alle Datenbankpartitionen ähnliche Speicheranforderungen haben. Beachten Sie die folgenden Richtlinien bei der Entscheidung, ob die automatische Speicheroptimierung für eine partitionierte Datenbank aktiviert werden sollte.

Fälle, in denen die automatische Speicheroptimierung für partitionierte 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 Datenbankpartitionen sind gleichmäßig auf mehrere physische Datenbankpartitionen verteilt.
- Es ist eine perfekte oder nahezu perfekte Verteilung von Daten vorhanden.
- Auslastungen werden gleichmäßig über Datenbankpartitionen verteilt. Das heißt, keine Datenbankpartition hat höheren Speicherbedarf für einen oder mehrere Zwischenspeicherbereiche als irgendeine der anderen Datenbankpartitionen.

Wenn in einer solchen Umgebung alle Datenbankpartitionen gleich konfiguriert sind, sorgt die automatische Speicheroptimierung für eine ordnungsgemäße Konfiguration des Systems.

Fälle, in denen die automatische Speicheroptimierung für partitionierte Datenbanken unter Vorkehrungen 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 Sorgfalt erfolgt. Solche Systeme haben möglicherweise nur eine Gruppe von Datenbankpartitionen für Daten und 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.

Die automatische Speicheroptimierung sollte in allen Datenbankpartitionen aktiviert werden, die Daten enthalten, wobei eine dieser Datenbankpartitionen als Optimierungspartition vorgesehen werden sollte. Da die Koordinatorpartitionen und die Katalogpartitionen verschieden konfiguriert sein können, sollte außerdem die automatische Speicheroptimierung in diesen Partitionen inaktiviert werden. Zur Inaktivierung der automatischen Speicheroptimierung in den Koordinator- und Katalogpartitionen setzen Sie den Datenbankkonfigurationsparameter **self_tuning_mem** in diesen Partitionen auf den Wert OFF.

Fälle, in denen die automatische Speicheroptimierung für partitionierte Datenbanken nicht empfohlen wird

Wenn der Speicherbedarf für die einzelnen Datenbankpartitionen unterschiedlich ist oder verschiedene Datenbankpartitionen auf erheblich unterschiedlicher Hardware betrieben werden, ist es eine empfohlene Methode, die Funktion der automatischen Speicheroptimierung zu inaktivieren. Sie können die Funktion inaktivieren, indem Sie den Datenbankkonfigurationsparameter **self_tuning_mem** in allen Partitionen auf den Wert OFF setzen.

Vergleich des Speicherbedarfs 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 Datenbankpartitionen ähnliche Werte liefern (mit Abweichungen unter 20 %), können die Datenbankpartitionen als ausreichend ähnlich betrachtet werden.

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

```
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

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

Anzahl OLAP-Funktionen                                                  = 0
Anzahl Überläufe von OLAP-Funktionen                                    = 0
Aktive OLAP-Funktionen                                                  = 0
```

Erfassen Sie die folgenden Daten, indem Sie den Befehl `get snapshot for bufferpools on <datenbankname>` 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 Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken aktiviert wird, überwacht eine einzige Datenbankpartition (die *Optimierungspartition*) 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 Pufferpools.

- Zur Ermittlung, welche Datenbankpartition zurzeit als Optimierungspartition angegeben ist, rufen Sie die Prozedur **ADMIN_CMD** wie folgt auf:

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

- Zum Ändern der Optimierungspartition rufen Sie die Prozedur **ADMIN_CMD** wie folgt auf:

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

  Die Optimierungspartition wird asynchron oder beim nächsten Start der Datenbank aktualisiert. Wenn die Speicheroptimierungsfunktion die Optimierungspartition automatisch auswählen soll, geben Sie '-1' für *partitionsnummer* ein.

Starten der Speicheroptimierungsfunktion in Umgebungen mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken wird die Speicheroptimierungsfunktion nur gestartet, wenn die Datenbank explizit mit dem Befehl **ACTIVATE DATABASE** aktiviert wird, weil die automatische Speicheroptimierung voraussetzt, dass alle Partitionen aktiv sind.

Inaktivieren der automatischen Speicheroptimierungsfunktion für eine bestimmte Datenbankpartition

- Zur Inaktivierung der automatischen Speicheroptimierung für eine Teilgruppe von Datenbankpartitionen setzen Sie den Datenbankkonfigurationsparameter **self_tuning_mem** für diese Datenbankpartitionen auf den Wert OFF.
- Zur Inaktivierung der automatischen Speicheroptimierung 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 empfohlen, die Werte der Konfigurationsparameter für die automatische Speicheroptimierungsfunktion über alle aktiven Partitionen hinweg einheitlich zu definieren.
- Zur Inaktivierung der automatischen Speicheroptimierung für einen bestimmten Pufferpool in einer bestimmten Datenbankpartition führen Sie die Anweisung

ALTER BUFFERPOOL aus, indem Sie einen Größenwert sowie die Partition angeben, in der die automatische Speicheroptimierung inaktiviert werden soll.

Eine Anweisung ALTER BUFFERPOOL, die die Größe eines Pufferpools in einer bestimmten Datenbankpartition angibt, erstellt einen Ausnahmeeintrag (bzw. aktualisiert einen vorhandenen Eintrag) für diesen Pufferpool in der Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS. Wenn ein Ausnahmeeintrag für einen Pufferpool vorhanden ist, wird dieser Pufferpool an Operationen zur 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 aus, um die Größe des Pufferpools in dieser Datenbankpartition auf den Standardwert zu setzen.
3. Aktivieren Sie die automatische Optimierung für diesen Pufferpool, indem Sie eine weitere Anweisung ALTER BUFFERPOOL ausführen, die die Pufferpoolgröße auf den Wert AUTOMATIC setzt.

Aktivieren des Speichers mit automatischer Leistungsoptimierung in nicht einheitlichen Umgebungen

Im Idealfall sollten Daten gleichmäßig auf alle Datenbankpartitionen verteilt und die Auslastung, die in jeder Partition ausgeführt wird, durch ähnliche Speicheranforderungen gekennzeichnet sein. Wenn die Datenverteilung ungleichmäßig ist, sodass mindestens eine Datenbankpartition erheblich mehr oder weniger Daten als andere Datenbankpartitionen enthält, sollte für solche anomalen Datenbankpartitionen die automatische Speicheroptimierung nicht 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 Speicheroptimierung 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, ermitteln Sie eine Gruppe von Datenbankpartitionen, die ähnliche Daten- und Speicheranforderungen haben, und aktivieren für diese die automatische Speicheroptimierung. Der Speicher in den übrigen Partitionen muss manuell konfiguriert werden.

Kapitel 21. DB2-Konfigurationsparameter und -Variablen

Konfigurieren von Datenbanken über mehrere Partitionen

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

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

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

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

  ```
  DB2_UPDDBCFG_SINGLE_DBPARTITION=TRUE
  ```

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

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

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

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG DBPARTITIONNUM 10 USING sortheap 1000')
  ```
- Sie können den Parameter **DBPARTITIONNUM** mit der API db2CfgSet verwenden. Die Markierungen (Flags) in der Struktur **db2Cfg** geben an, ob der Wert für die Datenbankkonfiguration auf nur eine Datenbankpartition angewendet werden soll. Wenn Sie eine Markierung setzen, müssen Sie auch den Wert für **DBPARTITIONNUM** angeben. Beispiel:

  ```
  #define db2CfgSingleDbpartition          256
  ```

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

Bei einem Upgrade der Datenbanken auf Version 9.7 behalten vorhandene Datenbankkonfigurationsparameter im Allgemeinen ihre Werte nach dem Upgrade bei. Es werden jedoch neue Parameter mit Standardwerten hinzugefügt, und einige vorhandene Parameter werden auf ihre neuen Standardwerte der Version 9.7 gesetzt. Weitere detaillierte Informationen zu den Änderungen an vorhandenen Datenbankkonfigurationsparametern finden Sie im Abschnitt „Änderungen am Verhalten des DB2-Servers" im Handbuch *Upgrade auf DB2 Version 10.5*. Alle nachfolgenden Anforderungen zum Aktualisieren oder Zurücksetzen der Datenbankkonfiguration für die Datenbanken, für die das Upgrade erfolgt ist, werden standardmäßig auf alle Datenbankpartitionen angewendet.

Für vorhandene Aktualisierungs- oder Zurücksetzbefehlsscripts gelten dieselben zuvor genannten Regeln für alle Datenbankpartitionen. Sie können Ihre Scripts ändern, um die Option **DBPARTITIONNUM** des Befehls **UPDATE DATABASE CONFIGURATION** oder **RESET DATABASE CONFIGURATION** einzufügen, oder Sie können die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren.

Für vorhandene Anwendungen, die die API **db2CfgSet** aufrufen, müssen Sie nach den Anweisungen für Version 9.5 oder eine spätere Version verfahren. Wenn Sie das Verhalten vor Version 9.5 wünschen, können Sie die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren oder Ihre Anwendungen in der Weise ändern, dass sie die API mit der Versionsnummer von Version 9.5 (oder einer späteren Version), einschließlich der neuen Markierung (Flag) `db2CfgSingleDbpartition` und des neuen Felds **dbpartitionnum**, aufrufen, um Datenbankkonfigurationen für eine bestimmte Datenbankpartition zu aktualisieren bzw. zurückzusetzen.

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

Variablen für Umgebungen mit partitionierten Datenbanken

Variablen für Umgebungen mit partitionierten Datenbanken können dazu verwendet werden, das Standardverhalten einer Umgebung mit partitionierten Datenbanken zu steuern, zum Beispiel das Verhalten bei der Vergabe von Berechtigungen, Funktionsübernahmen und dem Netzbetrieb.

DB2CHGPWD_EEE

- Betriebssystem: DB2 ESE unter AIX, Linux und Windows
- Standardwert=NULL, Werte: YES oder NO
- Diese Variable gibt an, ob Sie zulassen, dass andere Benutzer Kennwörter auf ESE-Systemen unter AIX oder Windows ändern. Es muss sichergestellt werden, dass die Kennwörter für alle Datenbankpartitionen oder Knoten mithilfe eines Windows-Domänencontrollers unter Windows oder LDAP unter AIX zentral verwaltet werden. Wenn Kennwörter nicht zentral verwaltet werden, bleiben sie möglicherweise nicht für alle Datenbankpartitionen bzw. Knoten konsistent. Dies könnte dazu führen, dass ein Kennwort nur in der Datenbankpartition geändert wird, mit der der Benutzer zur Durchführung der Änderung verbunden ist.

DB2_FCM_SETTINGS

- Betriebssystem: Linux
- Standardwert=`YES`, Werte:
  - `FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE]`. Der Standardwert für `FCM_MAXIMIZE_SET_SIZE` ist `YES`.

– FCM_CFG_BASE_AS_FLOOR: [YES|TRUE|NO|FALSE]. Der Standardwert für FCM_CFG_BASE_AS_FLOOR ist NO.

- Sie können die Registrierdatenbankvariable **DB2_FCM_SETTINGS** mit dem Token FCM_MAXIMIZE_SET_SIZE definieren, um einen Standardspeicherbereich von 4 GB für den FCM-Puffer (FCM, Fast Communication Manager) vorab zuzuordnen. Das Token muss entweder den Wert YES oder den Wert TRUE haben, um diese Funktion zu aktivieren.

  Sie können die Registrierdatenbankvariable **DB2_FCM_SETTINGS** mit der Option FCM_CFG_BASE_AS_FLOOR verwenden, um den Basiswert als untere Grenze für die Konfigurationsparameter *fcm_num_buffers* und *fcm_num_channels* des Datenbankmanagers festzulegen. Wenn für die Option FCM_CFG_BASE_AS_FLOOR die Einstellung YES bzw. TRUE definiert ist und diese Parameter auf AUTOMATIC gesetzt sind und einen Anfangs- oder Startwert aufweisen, werden sie von DB2 nicht auf einen Wert unterhalb dieser Grenze optimiert.

DB2_FORCE_OFFLINE_ADD_PARTITION

- Betriebssystem: Alle
- Standardwert=FALSE, Werte: FALSE oder TRUE
- Mit dieser Variablen können Sie angeben, dass Operationen zum Hinzufügen von Datenbankpartitionsservern offline ausgeführt werden. Die Standardeinstellung FALSE gibt an, dass DB2-Datenbankpartitionsserver hinzufügt werden können, ohne die Datenbank offline zu nehmen. Wenn die Operation jedoch offline ausgeführt werden soll oder wenn eine Einschränkung das Hinzufügen von Datenbankpartitionsservern im Onlinemodus der Datenbank unmöglich macht, setzen Sie **DB2_FORCE_OFFLINE_ADD_PARTITION** auf den Wert TRUE. Wenn diese Variable auf den Wert TRUE gesetzt ist, werden neue DB2-Datenbankpartitionsserver entsprechend dem Verhalten von Version 9.5 und früheren Versionen hinzugefügt. Das heißt, neue Datenbankpartitionsserver werden für die Instanz erst sichtbar, wenn sie heruntergefahren und neu gestartet wurde.

DB2_NUM_FAILOVER_NODES

- Betriebssystem: Alle
- Standardwert=2, Werte: 0 bis zur erforderlichen Anzahl der Datenbankpartitionen
- Definieren Sie **DB2_NUM_FAILOVER_NODES**, um die Anzahl der zusätzlichen Datenbankpartitionen anzugeben, die auf einer Maschine im Falle einer Funktionsübernahme gestartet werden müssen.

  In einer DB2-Datenbanklösung mit hoher Verfügbarkeit können beim Ausfall eines Datenbankservers die Datenbankpartitionen auf der Maschine, auf der der Fehler aufgetreten ist, auf einer anderen Maschine erneut gestartet werden. FCM (Fast Communication Manager) verwendet **DB2_NUM_FAILOVER_NODES**, um zu berechnen, wie viel Speicherplatz auf den verschiedenen Maschinen reserviert werden muss, um diese Funktionsübernahme zu ermöglichen.

  Betrachten Sie zum Beispiel die folgende Konfiguration:

  – Maschine A verfügt über zwei Datenbankpartitionen: 1 und 2.
  – Maschine B verfügt über zwei Datenbankpartitionen: 3 und 4.
  – Für **DB2_NUM_FAILOVER_NODES** wird sowohl auf Maschine A als auch auf Maschine B der Wert 2 angegeben.

Bei Ausgabe des Befehls START DATABASE MANAGER reserviert FCM genügend Speicherplatz auf A und B, um bis zu vier Datenbankpartitionen zu verwalten, sodass bei Ausfall einer Maschine die beiden Datenbankpartitionen der fehlerhaften Einheit auf der anderen Maschine erneut gestartet werden können. Wenn Maschine A ausfällt, dann können die Datenbankpartitionen 1 und 2 auf Maschine B erneut gestartet werden. Wenn Maschine B ausfällt, dann können die Datenbankpartitionen 3 und 4 auf Maschine A erneut gestartet werden.

DB2_PARTITIONEDLOAD_DEFAULT

- Betriebssystem: Alle unterstützten ESE-Plattformen
- Standardwert=YES, Werte: YES oder NO
- Die Registrierdatenbankvariable **DB2_PARTITIONEDLOAD_DEFAULT** ermöglicht Benutzern, die Standardfunktionsweise des Dienstprogramms LOAD in einer ESE-Umgebung zu ändern, wenn keine ESE-spezifischen LOAD-Optionen angegeben werden. Der Standardwert `YES` gibt an, dass in einer ESE-Umgebung, wenn Sie keine ESE-spezifischen LOAD-Optionen angeben, das Laden in allen Datenbankpartitionen versucht wird, in denen die Zieltabelle definiert ist. Wenn der Wert `NO` ist, wird das Laden nur in der Datenbankpartition versucht, mit der das Dienstprogramm LOAD gegenwärtig verbunden ist.

  Anmerkung: Diese Variable ist veraltet und wird in einem späteren Release möglicherweise entfernt. Der Befehl LOAD bietet verschiedene Optionen, die zur Erzielung der gleichen Funktionsweise verwendet werden können. Sie können die gleichen Ergebnisse wie mit der Einstellung `NO` für diese Variable erzielen, indem Sie die folgenden Optionen im Befehl **LOAD** angeben: `PARTITIONED DB CONFIG MODE LOAD_ONLY OUTPUT_DBPARTNUMS x`. Dabei steht x für die Partitionsnummer der Partition, in die Sie Daten laden wollen.

DB2PORTRANGE

- Betriebssystem: Windows
- Werte: nnnn:nnnn
- Dieser Wert wird auf den TCP/IP-Portbereich gesetzt, der vom FCM verwendet wird, sodass alle zusätzlichen auf einer anderen Maschine erstellten Datenbankpartitionen den gleichen Portbereich haben.

Konfigurationsparameter für Umgebungen mit partitionierten Datenbanken

Kommunikation

conn_elapse - Antwortzeit für Verbindung

Dieser Parameter gibt an, innerhalb welcher Anzahl Sekunden eine Netzverbindung zwischen DB2-Membern aufgebaut werden soll.

Konfigurationstyp
Datenbankmanager

Gilt für
DB2 pureScale-Server (mit mehreren DB2-Membern)

Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`10 [0-100]`

Maßeinheit
Sekunden

Wird der Verbindungsaufbau innerhalb der durch den Parameter angegebenen Zeit erfolgreich durchgeführt, kommt es zum Datenaustausch. Wird die Verbindung nicht rechtzeitig aufgebaut, wird ein weiterer Versuch zum Verbindungsaufbau durchgeführt. Kommt innerhalb der im Parameter **max_connretries** angegebenen Anzahl von Neuversuchen keine Verbindung zustande, wird eine Fehlernachricht ausgegeben.

fcm_num_buffers - Anzahl FCM-Puffer

Dieser Parameter gibt die Anzahl von 4-KB-Puffern an, die sowohl zwischen den als auch innerhalb der Datenbankserver für interne Kommunikation (Nachrichten) verwendet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver oder DB2 pureScale-Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
`Automatic [895 - 65300]`

64-Bit-Plattformen
`Automatic [895 - 524288]`

- Datenbankserver mit lokalen und fernen Clients: 1024
- Datenbankserver mit lokalen Clients: 895
- Partitionierter Datenbankserver oder DB2 pureScale-Datenbankserver mit lokalen oder fernen Clients: 4096

Fast Communication Manager-Puffer (FCM) werden standardmäßig sowohl für die Kommunikation zwischen Membern als auch innerhalb eines Members verwendet.

Wichtig: Der Standardwert des Parameters **fcm_num_buffers** unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Sie können sowohl einen Anfangswert als auch den Wert AUTOMATIC für den Konfigurationsparameter **fcm_num_buffers** festlegen. Wenn Sie für den Parameter den Wert AUTOMATIC festlegen, überwacht FCM die Ressourcennutzung und kann Ressourcen erhöhen bzw. verringern, wenn sie innerhalb von 30 Minuten nicht genutzt werden. Der Wert, um den Ressourcen erhöht oder verringert werden, hängt vom Betriebssystem ab. Unter Linux-Betriebssystemen kann die Anzahl der Puffer gegenüber dem Anfangswert nur um 25 Prozent erhöht werden. Wenn der Datenbankmanager versucht, eine Instanz zu starten und die angegebene Anzahl Puffer nicht zuordnen kann, wird diese Anzahl verringert, bis die Instanz gestartet werden kann.

Falls für den Parameter **fcm_num_buffers** sowohl ein bestimmter Wert als auch AUTOMATIC festgelegt werden soll und Sie nicht möchten, dass der Systemcontroller-Thread die Ressourcen bis auf einen Wert unter dem angegebenen Wert anpasst, müssen Sie die Option FCM_CFG_BASE_AS_FLOOR der Registrierdatenbankvariable **DB2_FCM_SETTINGS** auf YES oder TRUE festlegen. Der Wert der Registrierdatenbankvariablen **DB2_FCM_SETTINGS** wird dynamisch angepasst.

Wenn Sie mehrere logische Knoten verwenden, wird ein Pool mit der im Parameter **fcm_num_buffers** angegebenen Anzahl von Puffern von allen logischen Knoten auf derselben Maschine gemeinsam genutzt. Sie können die Größe des Pools festlegen, indem Sie den Wert des Parameters **fcm_num_buffers** mit der Anzahl der logischen Knoten auf der physischen Maschine multiplizieren. Überprüfen Sie den Wert, den Sie verwenden. Berücksichtigen Sie, wie viele FCM-Puffer auf einer Maschine oder Maschinen mit mehreren logischen Knoten zugeordnet werden. Wenn Sie über mehrere logische Knoten auf derselben Maschine verfügen, müssen Sie den Wert des Parameters **fcm_num_buffers** möglicherweise erhöhen. Die Anzahl der Benutzer im System, die Anzahl der Datenbankpartitionsserver im System oder die Komplexität der Anwendungen kann dazu führen, dass die Anzahl der Nachrichtenpuffer in einem System nicht ausreicht.

fcm_num_channels - Anzahl FCM-Kanäle

Mit diesem Parameter wird die Anzahl der FCM-Kanäle für jede Datenbankpartition angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver oder DB2 pureScale-Datenbankserver mit lokalen und fernen Clients
- Satellitendatenbankserver mit lokalen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX-Plattformen (32-Bit)
Automatic mit dem Anfangswert 256, 512 oder 2048 [128 - 120000]

UNIX-Plattformen (64 Bit)
Automatic mit dem Anfangswert 256, 512 oder 2048 [128 - 524288]

Windows (32 Bit)
Automatic mit dem Anfangswert 10000 [128 - 120000]

Windows (64 Bit)
Automatic mit dem Anfangswert 256, 512 oder 2048 [128 - 524288]

Im Folgenden finden Sie die Standardanfangswerte für die verschiedenen Servertypen:

- Für einen Datenbankserver mit lokalen und fernen Clients ist der Anfangswert 512.
- Für einen Datenbankserver mit lokalen Clients ist der Anfangswert 256.
- Für Server in Umgebungen mit partitionierten Datenbanken mit lokalen und fernen Clients ist der Anfangswert 2048.

Fast Communication Manager-Puffer (FCM) werden standardmäßig sowohl für die Kommunikation zwischen Membern als auch innerhalb eines Members verwendet. Damit Datenbanksysteme ohne Cluster das FCM-Subsystem und den Parameter **fcm_num_channels** verwenden können, musste der Parameter **intra_parallel** auf die Einstellung YES gesetzt werden.

Ein FCM-Kanal stellt einen logischen Kommunikationsendpunkt zwischen EDUs (Engine Dispatchable Units) dar, die in der DB2-Steuerkomponente ausgeführt werden. Sowohl Steuerungsflüsse (Anforderung und Antwort) als auch Datenflüsse (Daten aus Tabellenwarteschlangen) nutzen Kanäle, um Daten zwischen Membern zu übertragen.

Wenn dieser Parameter auf AUTOMATIC gesetzt ist, überwacht FCM die Kanalnutzung und ordnet Ressourcen je nach Bedarfsänderung inkrementell zu bzw. gibt sie inkrementell frei.

max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten

Dieser Parameter gibt an, wie häufig höchstens versucht wird, eine Netzverbindung zwischen zwei DB2-Membern aufzubauen.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

DB2 pureScale-Server

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
5 [0-100]

Wenn der Verbindungsaufbau zwischen zwei DB2-Membern fehlschlägt (z. B. bei Erreichen des im Parameter **conn_elapse** angegebenen Werts), gibt **max_connretries** an, wie viele Wiederholungen durchgeführt werden dürfen, um die Verbindung zu einem DB2-Member herzustellen. Bei Überschreitung des für diesen Parameter angegebenen Werts wird eine Fehlernachricht zurückgegeben.

max_time_diff - Maximale Zeitdifferenz zwischen Membern ()

Dieser Parameter gibt die maximale Zeitdifferenz an, die in einer DB2 pureScale-Umgebung zwischen den in der Knotenkonfigurationsdatei aufgelisteten Membern zulässig ist.

Konfigurationstyp
Datenbankmanager

Gilt für
Member mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

In DB2 pureScale-Umgebungen
`1 [1 - 1 440]`

Außerhalb von DB2 pureScale-Umgebungen
`60 [1 - 1 440]`

Maßeinheit
Minuten

Jedes Member verfügt über eine eigene Systemuhr. Der Zeitunterschied zwischen den Systemuhren von zwei oder mehr Membern wird regelmäßig überprüft. Wenn die Zeitdifferenz zwischen den Systemuhren den Wert überschreitet, der im Parameter **max_time_diff** angegeben ist, werden in den db2diag-Protokolldateien Warnungen protokolliert.

Um in einer DB2 pureScale-Umgebung sicherzustellen, dass die Member synchron zueinander bleiben, ist eine NTP-Konfiguration (NTP - Network Time Protocol) erforderlich, die regelmäßig auf den einzelnen Membern überprüft wird. Wenn der NTP-Dämon nicht gefunden werden kann, werden entsprechende Warnungen in den db2diag-Protokolldateien protokolliert.

Der Fehler SQL1473N wird in partitionierten Datenbankumgebungen gemeldet, in denen die Systemuhr mit der virtuellen Zeitmarke (Virtual Time Stamp, VTS) verglichen wird, die in der Protokollsteuerdatei SQLOGCTL.LFH gespeichert ist. Wenn die Zeitmarke in der Protokollsteuerdatei .LFH kleiner als die Systemzeit ist, wird die Zeit im Datenbankprotokoll auf die virtuelle Zeitmarke gesetzt, bis die Systemuhr mit dieser Zeitmarke übereinstimmt.

Der DB2-Datenbankmanager verwendet die Weltzeit (Coordinated Universal Time, UTC), damit unterschiedliche Zeitzonen beim Festlegen des Parameters **max_time_diff** keine Rolle spielen. Die Weltzeit (UTC) ist identisch mit der Westeuropäischen Zeit (Greenwich Mean Time).

start_stop_time - Zeitlimit für DB2START und DB2STOP

Dieser Parameter gibt die Zeit (in Minuten) an, innerhalb der alle Datenbankpartitionsserver auf einen Befehl zum Starten oder Stoppen des Datenbankmanagers (**START DBM** bzw. **STOP DBM**) reagieren müssen. Außerdem wird er als Zeitlimitwert für **ADD DBPARTITIONNUM**- und **DROP DBPARTITIONNUM**-Operationen verwendet.

Konfigurationstyp
Datenbankmanager

Gilt für
Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`10 [1 - 1 440]`

Maßeinheit
Minuten

Member oder Knoten, die nicht innerhalb der angegebenen Zeit auf den Befehl **db2start** oder **db2stop** reagieren, werden in einer Instanz mit mehreren Membern/Knoten automatisch von **db2start** oder **db2stop** abgebrochen und bereinigt. Die Diagnosenachrichten werden in dem Pfad **diagpath** protokolliert, der in der Datenbankmanagerkonfiguration definiert ist oder mit dem zugehörigen Standardwert (z. B. `sqllib/db2dump/ $m` unter UNIX-Betriebssystemen).

Wenn eine **db2start**- oder **db2stop**-Operation in einer Mehrpartitionsdatenbank nicht innerhalb des Zeitraums ausgeführt wird, der durch den Konfigurationsparameter **start_stop_time** des Datenbankmanagers angegeben wurde, werden die Datenbankpartitionen automatisch abgebrochen und bereinigt, für die das zulässige Zeitlimit überschritten wurde. Bei Umgebungen mit vielen Datenbankpartitionen und einem geringen Wert für **start_stop_time** kann es zu diesem Verhalten kommen. Erhöhen Sie den Wert für **start_stop_time**, um diesem Verhalten entgegenzuwirken.

Beim Hinzufügen einer neuen Datenbankpartition mithilfe des Befehls **db2start**, **START DATABASE MANAGER** oder **ADD DBPARTITIONNUM** muss die Operation zum Hinzufügen der Datenbankpartition bestimmen, ob die einzelnen Datenbanken in der Instanz für dynamischen Speicher eingerichtet sind. Dies geschieht durch Kommunikation mit der Katalogpartition für jede einzelne Datenbank. Wenn der dynamische Speicher aktiviert ist, werden die Speicherpfaddefinitionen im Rahmen dieser Kommunikation abgerufen. Ähnlich gilt für den Fall, dass Tabellenbereiche für temporäre Tabellen zusammen mit den Datenbankpartitionen zu erstellen sind, dass die Operation möglicherweise mit einem anderen Datenbankpartitionsserver kommunizieren muss, um die Tabellenbereichsdefinitionen für die Datenbankpartitionen abzurufen, die sich auf diesem Server befinden. Diese Faktoren müssen bei der Festlegung des Werts für den Parameter **start_stop_time** berücksichtigt werden.

Parallelverarbeitung

intra_parallel - Partitionsinterne Parallelität aktivieren

Dieser Parameter gibt an, ob Datenbankverbindungen standardmäßig die partitionsinterne Abfrageparallelität verwenden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NO (0) [SYSTEM (-1), NO (0), YES (1)]

Der Wert -1 bewirkt, dass der Parameter auf YES oder NO gesetzt wird; abhängig von der Hardware, auf welcher der Datenbankmanager ausgeführt wird.

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Anmerkung:

- Bei der parallelen Indexerstellung wird dieser Konfigurationsparameter nicht verwendet.
- Wenn Sie diesen Parameterwert ändern, werden Pakete möglicherweise erneut an die Datenbank gebunden und es kann zu Leistungseinbußen kommen.
- Die Einstellung für **intra_parallel** kann in einer Anwendung durch einen Aufruf der Prozedur ADMIN_SET_INTRA_PARALLEL überschrieben werden. Sowohl die Einstellung für **intra_parallel** als auch der in einer Anwendung durch die Prozedur ADMIN_SET_INTRA_PARALLEL festgelegte Wert kann in einer Workload durch Festlegen des Attributs MAXIMUM DEGREE in einer Workloaddefinition überschrieben werden.

max_querydegree - max_querydegree - Maximaler Grad der Parallelität bei Abfragen

Dieser Parameter gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf dieser Instanz des Datenbankmanagers ausgeführt werden. Eine SQL-Anweisung verwendet bei ihrer Ausführung innerhalb einer Datenbankpartition nicht mehr als diese Anzahl paralleler Operationen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
-1 (ANY) [ANY, 1 - 32 767] (ANY bedeutet 'vom System festgelegt')

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Der Konfigurationsparameter **intra_parallel** muss aktiviert (YES) sein, damit die Datenbankpartition die partitionsinterne Parallelität für SQL-Anweisungen verwenden kann. Der Parameter **intra_parallel** ist für die parallele Indexerstellung nicht mehr erforderlich.

Der Standardwert für diesen Konfigurationsparameter lautet -1. Dieser Wert bedeutet, dass das System den vom Optimierungsprogramm festgelegten Parallelitätsgrad verwendet. Andernfalls wird der vom Benutzer angegebene Wert verwendet.

Anmerkung: Der Grad der Parallelität für eine SQL-Anweisung kann bei der Kompilierung der Anweisung mithilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption **DEGREE** angegeben werden.

Der maximale Grad der Parallelität für eine aktive Anwendung bei Abfragen kann mit dem Befehl **SET RUNTIME DEGREE** geändert werden. Der zur Laufzeit tatsächlich verwendete Parallelitätsgrad ist der niedrigste der folgenden Werte:

- Konfigurationsparameter **max_querydegree**
- Parallelitätsgrad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

Dieser Konfigurationsparameter gilt nur für Abfragen.

Teil 5. APIs, Befehle und SQL-Anweisungen zur Verwaltung

Kapitel 22. APIs zur Verwaltung

sqleaddn - Datenbankpartition der Umgebung mit partitionierten Datenbanken hinzufügen

Fügt eine Datenbankpartition auf einem Datenbankpartitionsserver hinzu.

Geltungsbereich

Diese API betrifft nur den Datenbankpartitionsserver, auf dem sie ausgeführt wird.

Berechtigung

Eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL

Erforderliche Verbindung

Keine

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqleaddn (
        void * pAddNodeOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgaddn (
        unsigned short addnOptionsLen,
        struct sqlca * pSqlca,
        void * pAddNodeOptions);
```

Parameter der API 'sqleaddn'

pAddNodeOptions
Eingabe. Ein Zeiger auf die optionale Struktur sqle_addn_options. Diese Struktur dient zur Angabe des Quellendatenbankpartitionsservers, sofern zutreffend, der Definitionen für Tabellenbereiche für temporäre Systemtabellen für alle zu erstellenden Datenbankpartitionen. Wenn nicht angegeben (d. h., wenn ein Nullzeiger angegeben wird), stimmen die Definitionen der Tabellenbereiche für temporäre Systemtabellen mit denen für die Katalogpartition überein.

pSqlca
Ausgabe. Ein Verweis auf die sqlca-Struktur.

Spezielle Parameter für die API 'sqlgaddn'

addnOptionsLen
Eingabe. Ein 2 Byte langer ganzzahliger Wert ohne Vorzeichen, der die Länge der optionalen Struktur sqle_addn_options in Byte angibt.

Hinweise

Diese API sollte nur verwendet werden, wenn ein Datenbankpartitionsserver zu einer Umgebung hinzugefügt wird, die nur über eine einzige Datenbank verfügt und deren Datenbank zum Zeitpunkt der Operation zum Hinzufügen der Partition nicht katalogisiert ist. In dieser Situation erkennt die Operation zum Hinzufügen der Partition die Datenbank nicht, da sie nicht katalogisiert ist, und erstellt keine Datenbankpartition für die Datenbank auf dem neuen Datenbankpartitionsserver. Alle Versuche, eine Verbindung zu der Datenbankpartition auf dem Datenbankpartitionsserver herzustellen, führen zu einem Fehler. Die API sqleaddn kann nur zum Erstellen der Datenbankpartition für die Datenbank auf dem neuen Datenbankpartitionsserver verwendet werden, wenn die Datenbank zuvor katalogisiert wurde.

Diese API sollte nicht verwendet werden, wenn die Umgebung über mehrere Datenbanken verfügt und mindestens eine der Datenbank katalogisiert ist, wenn die Operation zum Hinzufügen der Partition ausgeführt wird. Verwenden Sie in diesem Fall die API sqlecran zum Erstellen einer Datenbankpartition für die einzelnen Datenbanken, die bei Ausführung der Operation zum Hinzufügen der Partition nicht katalogisiert waren. Die API sqlecran kann erst zum Erstellen der Datenbankpartition für die Datenbank auf dem Datenbankpartitionsserver verwendet werden, wenn jede einzelne nicht katalogisierte Datenbank zuvor katalogisiert wurde.

Stellen Sie vor dem Hinzufügen einer neuen Datenbankpartition sicher, dass genügend Speicherplatz für die Container verfügbar ist, die erstellt werden müssen.

Die Operationen zum Hinzufügen eines Knotens erstellt eine leere Datenbankpartition auf dem neuen Datenbankpartitionsserver für jede Datenbank, die in der Instanz vorhanden ist. Die Konfigurationsparameter für die neuen Datenbankpartitionen werden auf ihre Standardwerte gesetzt.

Anmerkung: Nicht katalogisierte Datenbanken werden beim Hinzufügen einer neuen Datenbankpartition nicht erkannt. Die nicht katalogisierte Datenbank ist dann in der neuen Datenbankpartition nicht enthalten. Beim Versuch, eine Verbindung zu der Datenbank in der neuen Datenbankpartition herzustellen, wird die Fehlernachricht SQL1013N zurückgegeben.

Wenn eine Operationen zum Hinzufügen eines Knotens während der lokalen Erstellung einer Datenbankpartition fehlschlägt, tritt die Operation in eine Bereinigungsphase ein, in der sie alle erstellten Datenbanken lokal löscht. Dies bedeutet, dass die Datenbankpartitionen nur von dem Datenbankpartitionsserver entfernt werden, der hinzugefügt wird (d. h. vom lokalen Datenbankpartitionsserver). Vorhandene Datenbankpartitionen auf allen anderen Datenbankpartitionsservern bleiben unberührt. Wenn die Bereinigung fehlschlägt, wird keine weitere Bereinigung ausgeführt und ein Fehler zurückgegeben.

Die Datenbankpartitionen auf dem neuen Datenbankpartitionsserver können erst zum Speichern von Benutzerdaten verwendet werden, wenn der Datenbankpartitionsserver mit der Anweisung ALTER DATABASE PARTITION GROUP einer Datenbankpartitionsgruppe hinzugefügt wurde.

Diese API schlägt fehl, wenn gleichzeitig eine Operation zum Erstellen oder Löschen einer Datenbank ausgeführt wird. Die API kann erneut aufgerufen werden, wenn die Operation abgeschlossen ist.

Die Speicherpfaddefinitionen der Speichergruppen werden abgerufen, wenn die API sqleaddn für jede der Datenbanken in der Instanz mit der Katalogpartition

kommunizieren muss. Ähnlich gilt für den Fall, dass Tabellenbereiche für temporäre Systemtabellen zusammen mit den Datenbankpartitionen zu erstellen sind, dass die API sqleaddn möglicherweise mit einem anderen Datenbankpartitionsserver in der Umgebung mit partitionierten Datenbanken kommunizieren muss, um die Tabellenbereichsdefinitionen abzurufen. Der Konfigurationsparameter **start_stop_time** des Datenbankmanagers wird zur Angabe der Zeit (in Minuten) verwendet, in der der andere Datenbankpartitionsserver mit den Definitionen zum dynamischen Speicher und zu den Tabellenbereichen antworten muss. Wenn diese Zeit überschritten wird, schlägt die API fehl. Erhöhen Sie in diesem Fall den Wert für den Parameter **start_stop_time** und rufen Sie die API erneut auf.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqlecran - Datenbank auf einem Datenbankpartitionsserver erstellen

Erstellt eine Datenbank nur auf dem Datenbankpartitionsserver, der die Anwendungsprogrammierschnittstelle (API) aufruft.

Diese API ist nicht für den allgemeinen Gebrauch vorgesehen. Sie kann zum Beispiel mit db2Restore verwendet werden, wenn die Datenbankpartition auf einem Datenbankpartitionsserver beschädigt wurde und erneut erstellt werden muss. Eine unsachgemäße Verwendung dieser API kann Inkonsistenzen im System zur Folge haben. Daher sollte sie nur mit Vorsicht verwendet werden.

Anmerkung: Wenn diese API zur erneuten Erstellung einer Datenbankpartition, die (wegen Beschädigung) gelöscht wurde, verwendet wird, befindet sich die Datenbank auf diesem Datenbankpartitionsserver anschließend im Status 'Restore anstehend'. Nach der erneuten Erstellung der Datenbankpartition muss die Datenbank sofort auf diesem Datenbankpartitionsserver durch einen Restore wiederhergestellt werden.

Geltungsbereich

Diese API betrifft nur den Datenbankpartitionsserver, für den sie aufgerufen wird.

Berechtigung

Eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL

Erforderliche Verbindung

Instanzverbindung. Zur Erstellung einer Datenbank auf einem anderen Datenbankpartitionsserver ist es erforderlich, zunächst eine Verbindung (ATTACH) zu diesem Datenbankpartitionsserver herzustellen. Eine Datenbankverbindung wird während der Verarbeitung durch diese API temporär hergestellt.

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqlecran (
        char * pDbName,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcran (
        unsigned short reservedLen,
        unsigned short dbNameLen,
        struct sqlca * pSqlca,
        void * pReserved,
        char * pDbName);
```

Parameter der API 'sqlecran'

pDbName
Eingabe. Eine Zeichenfolge, die den Namen der zu erstellenden Datenbank enthält. Darf nicht den Wert NULL haben.

pReserved
Eingabe. Ein reservierter Zeiger, der auf NULL gesetzt ist oder auf null zeigt. Reserviert für zukünftigen Gebrauch.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur 'sqlca'.

Spezielle Parameter für die API 'sqlgcran'

reservedLen
Eingabe. Reserviert für die Länge des Parameters **pReserved**.

dbNameLen
Eingabe. Ein 2 Byte langer ganzzahliger Wert ohne Vorzeichen, der die Länge des Datenbanknamens in Byte angibt.

Hinweise

Wenn die Datenbank erfolgreich erstellt wird, wird sie in den Status 'Restore anstehend' versetzt. Für die Datenbank muss ein Restore auf diesem Datenbankpartitionsserver ausgeführt werden, bevor sie verwendet werden kann.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqledpan - Datenbank auf einem Datenbankpartitionsserver löschen

Löscht eine Datenbank auf einem angegebenen Datenbankpartitionsserver. Kann nur in einer Umgebung mit partitionierten Datenbanken ausgeführt werden.

Geltungsbereich

Diese Anwendungsprogrammierschnittstelle (API) betrifft nur den Datenbankpartitionsserver, für den sie aufgerufen wird.

Berechtigung

Eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL

Erforderliche Verbindung

Keine. Eine Instanzverbindung (ATTACH) wird für die Dauer des Aufrufs hergestellt.

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqledpan (
        char * pDbAlias,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdpan (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        void * pReserved2,
        char * pDbAlias);
```

Parameter der API 'sqledpan'

pDbAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der zu löschenden Datenbank enthält. Dieser Name dient zum Verweis auf den tatsächlichen Datenbanknamen im Systemdatenbankverzeichnis.

pReserved
Reserviert. Sollte NULL sein.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur 'sqlca'.

Spezielle Parameter für die API 'sqlgdpan'

Reserved1
Reserviert für zukünftigen Gebrauch.

DbAliasLen
Eingabe. Ein 2 Byte langer ganzzahliger Wert ohne Vorzeichen, der die Länge des Aliasnamens der Datenbank in Byte angibt.

pReserved2
Ein reservierter Zeiger, der auf NULL gesetzt ist oder auf null zeigt. Reserviert für zukünftigen Gebrauch.

Hinweise

Eine unsachgemäße Verwendung dieser API kann Inkonsistenzen im System zur Folge haben. Daher sollte sie nur mit Vorsicht verwendet werden.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqledrpn - Überprüfen, ob ein Datenbankpartitionsserver gelöscht werden kann

Prüft, ob ein Datenbankpartitionsserver von einer Datenbank verwendet wird. Es wird eine Nachricht zurückgegeben, die angibt, ob der Datenbankpartitionsserver gelöscht werden kann.

Geltungsbereich

Diese Anwendungsprogrammierschnittstelle (API) betrifft nur den Datenbankpartitionsserver, für den sie ausgeführt wird.

Berechtigung

Eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqledrpn (
        unsigned short Action,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpn (
        unsigned short Reserved1,
        struct sqlca * pSqlca,
        void * pReserved2,
        unsigned short Action);
```

Parameter der API 'sqledrpn'

Action
Die angeforderte Aktion. Gültiger Wert: `SQL_DROPNODE_VERIFY`

pReserved
Reserviert. Sollte `NULL` sein.

pSqlca
Ausgabe. Ein Verweis auf die sqlca-Struktur.

Spezielle Parameter für die API 'sqlgdrpn'

Reserved1
Reserviert für die Länge des Parameters **pReserved2**.

pReserved2
Ein reservierter Zeiger, der auf `NULL` gesetzt ist oder auf 0 zeigt. Für den zukünftigen Gebrauch reserviert.

Hinweise

Wenn eine Nachricht zurückgegeben wird, die besagt, dass der Datenbankpartitionsserver nicht im Gebrauch ist, können Sie den Befehl **db2stop** mit der Option **DROP NODENUM** verwenden, um den Eintrag für den Datenbankpartitionsserver aus der Datei db2nodes.cfg zu entfernen. Dadurch wird der Datenbankpartitionsserver zugleich aus der Umgebung mit partitionierten Datenbanken entfernt.

Wenn eine Nachricht zurückgegeben wird, die besagt, dass der Datenbankpartitionsserver im Gebrauch ist, sollten die folgenden Aktionen ausgeführt werden:

1. Der zu löschende Datenbankpartitionsserver enthält wahrscheinlich eine Datenbankpartition für jede Datenbank in der Instanz. Wenn einige dieser Datenbankpartitionen Daten enthalten, verteilen Sie die Datenbankpartitionsgruppen um, die diese Datenbankpartitionen verwenden. Verteilen Sie die Datenbankpartitionsgruppen um, um die Daten in Datenbankpartitionen zu versetzen, die sich auf Datenbankpartitionsservern befinden, die nicht gelöscht werden.
2. Nach der Umverteilung der Datenbankpartitionsgruppen können Sie die Datenbankpartition aus jeder Datenbankpartitionsgruppe löschen, die sie verwendet. Zum Entfernen einer Datenbankpartition aus einer Datenbankpartitionsgruppe können Sie entweder die Option zum Löschen von Knoten ('drop node') der API sqludrdt oder die Anweisung ALTER DATABASE PARTITION GROUP verwenden.
3. Löschen Sie alle Ereignismonitore, die auf dem Datenbankpartitionsserver definiert sind.
4. Führen Sie sqledrpn erneut aus, um sicherzustellen, dass die Datenbankpartition auf dem Datenbankpartitionsserver nicht mehr im Gebrauch ist.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqlugrpn - Nummer des Datenbankpartitionsservers für eine Zeile abrufen

Diese API wird seit Version 9.7 nicht weiter unterstützt. Sie können die Nummer der Datenbankpartition und des Datenbankpartitionsservers mit der API db2GetRowPartNum (Nummer des Datenbankpartitionsservers für eine Zeile abrufen) abrufen.

Wenn Sie die API sqlugrpn aufrufen und die Registrierdatenbankvariable **DB2_PMAP_COMPATIBILITY** auf OFF gesetzt ist, wird die Fehlernachricht SQL2768N zurückgegeben.

Gibt die Nummer der Datenbankpartition und des Datenbankpartitionsservers auf der Basis der Verteilungsschlüsselwerte zurück. Eine Anwendung kann anhand dieser Information feststellen, in welchem Datenbankpartitionsserver eine bestimmte Zeile einer Tabelle gespeichert ist.

Die Partitionierungsdatenstruktur sqlupi fungiert als Eingabe für diese Anwendungsprogrammierschnittstelle (API). Die Struktur kann von der API sqlugtpi zurückgegeben werden. Eine andere Eingabe sind die Zeichendarstellungen der entsprechenden Verteilungsschlüsselwerte. Die Ausgabe ist eine von der Verteilungsstrategie generierte Datenbankpartitionsnummer und die entsprechende Nummer des Datenbankpartitionsservers aus der Verteilungszuordnung. Wenn die

Informationen der Verteilungszuordnung nicht angegeben werden, wird nur die Datenbankpartitionsnummer zurückgegeben. Dies kann bei der Analyse der Datenverteilung nützlich sein.

Der Datenbankmanager braucht nicht aktiv zu sein, wenn diese API aufgerufen wird.

Geltungsbereich

Diese API muss von einem in der Datei `db2nodes.cfg` definierten Datenbankpartitionsserver aus aufgerufen werden. Diese API darf nicht von einem Client aus aufgerufen werden, da dies dazu führen könnte, dass falsche Informationen zur Datenbankpartitionierung auf Grund unterschiedlicher Codepages und ENDIAN-Definitionen zwischen dem Client und dem Server zurückgegeben werden.

Berechtigung

Keine

Einzuschließende Datei der API

```
sqlutil.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqlugrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_ctrycode,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);

SQL_API_RC SQL_API_FN
  sqlggrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_code,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);
```

Parameter der API 'sqlugrpn'

num_ptrs
Die Anzahl der Zeiger in **ptr_array**. Der Wert muss mit einem der Werte übereinstimmen, die für den Parameter **part_info** angegeben werden (d. h., **part_info**->**sqld**).

ptr_array
Eine Feldgruppe (Array) von Zeigern, die auf die Zeichendarstellungen der entsprechenden Werte jedes Teils des Verteilungsschlüssels verweisen, der in **part_info** angegeben wird. Wenn ein Nullwert erforderlich ist, wird der entsprechende Zeiger auf null gesetzt. Für generierte Spalten generiert diese Funktion keine Werte für die Zeile. Der Benutzer ist dafür verantwortlich, dass ein Wert angegeben wird, der eine ordnungsgemäße Partitionierung der Zeile zur Folge hat.

ptr_lens
Eine Feldgruppe aus ganzzahligen Werten (Integer) ohne Vorzeichen, die die Längen der Zeichendarstellungen der entsprechenden Werte jedes Teils des Partitionierungsschlüssels enthält, der in **part_info** angegeben wird.

territory_ctrycode
Der Landes-/Regionscode der Zieldatenbank. Dieser Wert kann auch aus der Datenbankkonfiguration mithilfe des Befehls **GET DATABASE CONFIGURATION** abgerufen werden.

codepage
Die Codepage der Zieldatenbank. Dieser Wert kann auch aus der Datenbankkonfiguration mithilfe des Befehls **GET DATABASE CONFIGURATION** abgerufen werden.

part_info
Ein Zeiger auf die Struktur sqlupi.

part_num
Ein Zeiger auf einen 2 Byte langen ganzzahligen Wert mit Vorzeichen, der zum Speichern der Datenbankpartitionsnummer verwendet wird.

node_num
Ein Zeiger auf ein SQL_PDB_NODE_TYPE-Feld, das zum Speichern der Knotennummer dient. Wenn der Zeiger null ist, wird keine Knotennummer zurückgegeben.

chklvl Ein ganzzahliger Wert ohne Vorzeichen, der den Grad der Überprüfung angibt, der auf Eingabeparameter angewendet wird. Wenn der angegebene Wert null ist, erfolgt keine Überprüfung. Wenn ein Wert ungleich null angegeben wird, werden alle Eingabeparameter überprüft.

sqlca Ausgabe. Ein Verweis auf die sqlca-Struktur.

dataformat
Gibt die Darstellung der Verteilungsschlüsselwerte an. Gültige Werte:

SQL_CHARSTRING_FORMAT
Alle Verteilungsschlüsselwerte werden durch Zeichenfolgen dargestellt. Dies ist der Standardwert.

SQL_IMPLIEDDECIMAL_FORMAT
Die Position einer impliziten Dezimalstelle wird durch die Spaltendefinition festgelegt. Wenn die Spaltendefinition zum Beispiel DECIMAL(8,2) angibt, wird der Wert 12345 als 123,45 verarbeitet.

SQL_PACKEDDECIMAL_FORMAT
Alle Verteilungsschlüsselwerte von Dezimalspalten werden im gepackten Dezimalformat verarbeitet.

SQL_BINARYNUMERICS_FORMAT
Alle numerischen Verteilungsschlüsselwerte liegen im Big Endian-Format vor.

pReserved1
Reserviert für zukünftigen Gebrauch.

pReserved2
Reserviert für zukünftigen Gebrauch.

Hinweise

Datentypen, die vom Betriebssystem unterstützt werden, sind diejenigen, die als Verteilungsschlüssel definiert werden können.

Anmerkung: CHAR-, VARCHAR-, GRAPHIC- und VARGRAPHIC-Datentypen müssen in die Datenbankcodepage konvertiert werden, bevor diese API aufgerufen wird.

Für numerische Datentypen und Datentypen für Datum und Uhrzeit müssen die Zeichendarstellungen in der Codepage des jeweiligen Systems vorliegen, auf dem die API aufgerufen wird.

Wenn **node_num** nicht null ist, muss die Verteilungszuordnung angegeben werden. Das heißt, dass das Feld **pmaplen** im Parameter **part_info** (**part_info**->**pmaplen**) entweder den Wert 2 oder den Wert 8192 hat. Ansonsten wird der SQLCODE-Wert -6038 zurückgegeben. Der Verteilungsschlüssel muss definiert werden. Das heißt, das Feld **sqld** im Parameter **part_info** (**part_info**->**sqld**) muss einen Wert größer null enthalten. Ansonsten wird der SQLCODE-Wert -2032 zurückgegeben.

Wenn einer Partitionierungsspalte, die keine Nullwerte enthalten darf, ein Nullwert zugewiesen wird, wird der SQLCODE-Wert -6039 zurückgegeben.

Alle führenden und folgenden Leerzeichen der Eingabezeichenfolge werden entfernt. Dies gilt jedoch nicht für CHAR-, VARCHAR-, GRAPHIC- und VARGRAPHIC-Datentypen. Bei diesen werden nur die folgenden Leerzeichen entfernt.

Kapitel 23. Befehle

REDISTRIBUTE DATABASE PARTITION GROUP

Führt eine Umverteilung von Daten auf die Partitionen in einer Datenbankpartitionsgruppe aus. Dieser Befehl betrifft alle Objekte, die sich in der Datenbankpartitionsgruppe befinden, und lässt sich nicht nur auf ein Objekt einschränken.

Dieser Befehl kann nur von der Katalogdatenbankpartition aus ausgeführt werden. Verwenden Sie den Befehl **LIST DATABASE DIRECTORY**, um festzustellen, welche Datenbankpartition die Katalogdatenbankpartition für die jeweilige Datenbank ist.

Geltungsbereich

Dieser Befehl betrifft alle Datenbankpartitionen in der jeweiligen Datenbankpartitionsgruppe.

Berechtigung

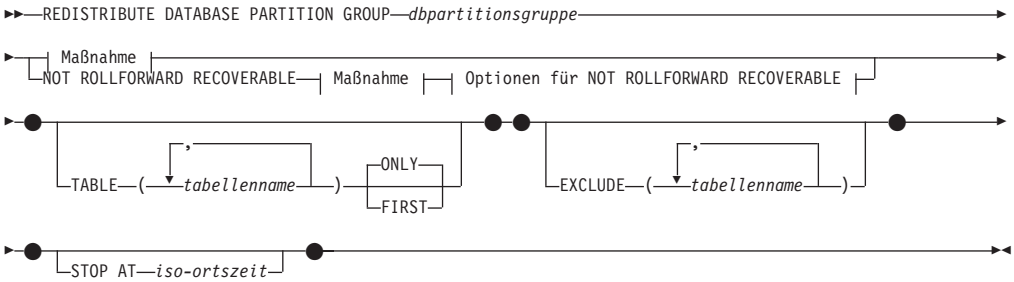
Eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL
- DBADM

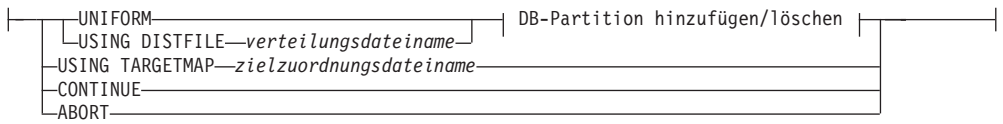
Darüber hinaus ist eine der folgenden Berechtigungsgruppen erforderlich:

- Zugriffsrechte DELETE, INSERT und SELECT für alle Tabellen der Datenbankpartitionsgruppe, die verteilt wird
- Berechtigung DATAACCESS

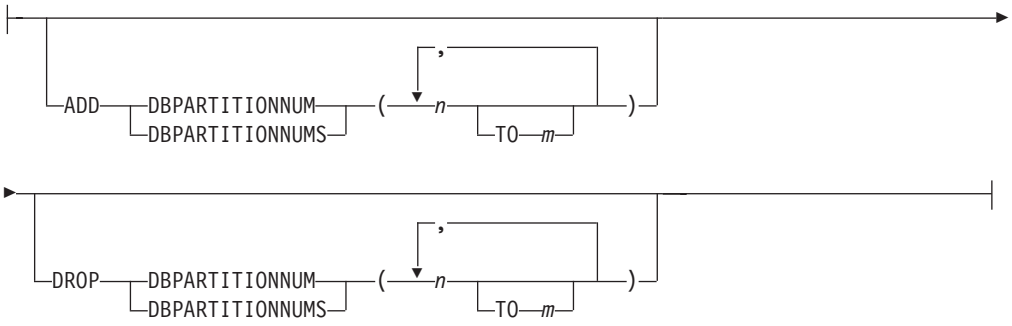
Befehlssyntax



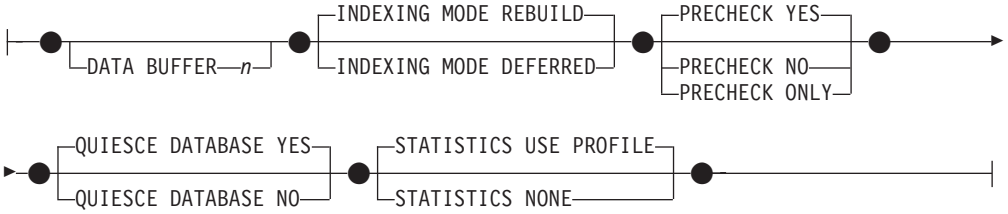
Aktion:



DB-Partition hinzufügen/löschen:



Optionen für NOT ROLLFORWARD RECOVERABLE:



Befehlsparameter

DATABASE PARTITION GROUP *dbpartitionsgruppe*
Der Name der Datenbankpartitionsgruppe. Dieser einteilige Name gibt eine Datenbankpartitionsgruppe an, die in der Katalogtabelle SYSCAT.DBPARTITIONGROUPS beschrieben ist. In der Datenbankpartitionsgruppe darf zu diesem Zeitpunkt keine Umverteilung stattfinden.

Anmerkung: Tabellen in den Datenbankpartitionsgruppen IBMCATGROUP und IBMTEMPGROUP können nicht umverteilt werden.

NOT ROLLFORWARD RECOVERABLE
Wenn diese Option verwendet wird, ist der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** nicht durch eine aktualisierende Recovery wiederherstellbar.

- Die Daten werden in einer Massendatenoperation und nicht durch interne Einfüge- und Löschoperationen versetzt. Dies verringert die Häufigkeit, mit der eine Tabelle durchsucht und ein Zugriff auf sie erfolgen muss, sodass sich eine bessere Leistung ergibt.
- Protokollsätze sind nicht mehr für jede einzelne der Einfüge- und Löschoperationen erforderlich. Dies bedeutet, dass Sie keine großen Speicherbereiche für aktive Protokolle und Protokollarchive mehr in Ihrem System verwalten müssen, wenn Sie eine Datenumverteilung ausführen.
- Bei Verwendung des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** mit der Option **NOT ROLLFORWARD RECOVERABLE** verwendet die Umverteilungsoperation bei Tabellen, die XML-Spalten enthalten, die Option **INDEXING MODE DEFERRED**. Wenn eine Tabelle keine XML-Spalte enthält, verwendet die Umverteilungsoperation beim Absetzen des Befehls den angegebenen Indexierungsmodus.

Wenn diese Option *nicht* verwendet wird, wird eine umfassende Protokollierung aller Zeilenbewegungen durchgeführt, sodass die Datenbank später bei Unterbrechungen, bei Fehlern oder bei sonstigen Bedarfssituationen wiederhergestellt werden kann.

UNIFORM
Gibt an, dass die Daten gleichmäßig auf Hashpartitionen verteilt werden (wobei für jede Hashpartition angenommen wird, dass sie dieselbe Anzahl von Zeilen hat), jedoch ist nicht jeder Datenbankpartition die gleiche Anzahl von Hashpartitionen zugeordnet. Nach der Umverteilung haben alle Datenbankpartitionen in der Datenbankpartitionsgruppe annähernd die gleiche Anzahl von Hashpartitionen.

USING DISTFILE *verteilungsdateiname*
Wenn die Verteilung der Verteilungsschlüsselwerte ungleichmäßig ist, können Sie diese Option verwenden, um eine gleichmäßige Umverteilung der Daten auf die Datenbankpartitionen einer Datenbankpartitionsgruppe zu erreichen.

Verwenden Sie die im Wert *verteilungsdateiname* angegebene Datei zur Angabe der aktuellen Verteilung von Daten auf die 32.768 Hashpartitionen.

Verwenden Sie Zeilenanzahlen, Bytevolumen oder eine andere Maßeinheit, um das Datenvolumen anzugeben, das von jeder Hashpartition dargestellt wird. Das Dienstprogramm interpretiert den ganzzahligen Wert, der einer Partition zugeordnet ist, als Gewichtung dieser Partition. Wenn durch *verteilungsdateiname* eine Verteilungsdatei angegeben wird, generiert das Dienstprogramm eine Zielverteilungszuordnung, die es zu einer möglichst gleichmäßigen Umverteilung der Daten auf die Datenbankpartitionen in der Datenbankpartitionsgruppe verwendet. Nach der Umverteilung ist die Gewichtung jeder Datenbankpartition in der Datenbankpartitionsgruppe annähernd gleich (die Gewichtung der Datenbankpartition ist die Summe der Gewichtungen aller Hashpartitionen, die dieser Datenbankpartition zugeordnet sind).

Zum Beispiel könnte die Eingabeverteilungsdatei die folgenden Einträge enthalten:

```
10223
1345
112000
0
100
...
```

In dem Beispiel hat die Hashpartition 2 die Gewichtung von 112.000, während der Partition 3 (mit der Gewichtung 0) überhaupt keine Daten zugeordnet sind.

Die Verteilungsdatei *verteilungsdateiname* sollte 32.768 positive ganzzahlige Werte im Zeichenformat enthalten. Die Summe der Werte darf den Wert 4.294.967.295 nicht überschreiten.

Wenn der Pfad für die Verteilungsdatei *verteilungsdateiname* nicht angegeben ist, wird das aktuelle Verzeichnis verwendet.

USING TARGETMAP *zielzuordnungsdateiname*
Die durch *zielzuordnungsdateiname* angegebene Datei wird als Zielverteilungszuordnung verwendet. Die Datenumverteilung wird entsprechend dieser Datei ausgeführt. Wenn der Pfad nicht angegeben wird, wird das aktuelle Verzeichnis verwendet.

Sie sollte 32768 Ganzzahlen enthalten, die jeweils eine gültige Datenbankpartitionsnummer darstellen. Die Nummer einer Zeile ordnet einer Datenbankparti-

tion einen Hashwert zu. Wenn also Zeile *X* den Wert *Y* enthält, müssen sich alle Datenwerte mit einem Wert für HASHEDVALUE() von *X* auf der Datenbankpartition *Y* befinden.

Wenn eine Datenbankpartition, die in der Zielzuordnung enthalten ist, nicht in der Datenbankpartitionsgruppe enthalten ist, wird ein Fehler zurückgegeben. Setzen Sie eine Anweisung ALTER DATABASE PARTITION GROUP ADD DBPARTITIONNUM ab, bevor Sie den Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** ausführen.

Wenn eine aus der Zielzuordnung ausgeschlossene Datenbankpartition in der Datenbankpartitionsgruppe enthalten *ist*, wird diese Datenbankpartition nicht in die Partitionierung mit einbezogen. Eine solche Datenbankpartition kann mit der Anweisung ALTER DATABASE PARTITION GROUP DROP DBPARTITIONNUM entweder vor oder nach der Ausführung des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** gelöscht werden.

CONTINUE
Setzt eine zuvor fehlgeschlagene oder gestoppte Operation **REDISTRIBUTE DATABASE PARTITION GROUP** fort. Wenn keine solche Operation stattgefunden hat, wird ein Fehler zurückgegeben.

ABORT
Bricht eine zuvor fehlgeschlagene oder gestoppte Operation **REDISTRIBUTE DATABASE PARTITION GROUP** ab. Wenn keine solche Operation stattgefunden hat, wird ein Fehler zurückgegeben.

ADD

DBPARTITIONNUM *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die in die Datenbankpartitionsgruppe einzufügen sind. Alle angegebenen Partitionen dürfen noch nicht in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42728). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel ADD DBPARTITIONNUM äquivalent.

DBPARTITIONNUMS *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die in die Datenbankpartitionsgruppe einzufügen sind. Alle angegebenen Partitionen dürfen noch nicht in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42728). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel ADD DBPARTITIONNUM äquivalent.

Anmerkung:

1. Wenn eine Datenbankpartition mit dieser Option hinzugefügt wird, basieren die Namen von Containern für Tabellenbereiche auf den Containern des entsprechenden Tabellenbereichs in der vorhandenen Partition mit der niedrigsten Nummer in der Datenbankpartitionsgruppe. Falls dies zu einer Namensunverträglichkeit unter Containern führen würde, was möglich ist, wenn sich die neuen Partitionen auf demselben physischen System wie vorhandene Container befinden, sollte diese Option nicht verwendet werden. Stattdessen sollte die Anweisung ALTER DA-

TABASE PARTITION GROUP mit der Option WITHOUT TABLESPACES verwendet werden, bevor der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** ausgeführt wird. Tabellenbereichscontainer können anschließend manuell unter Angabe geeigneter Namen erstellt werden.

2. Bei der Datenumverteilung werden möglicherweise Tabellenbereiche für alle neuen Datenbankpartitionen erstellt, wenn der Parameter **ADD DBPARTITIONNUMS** angegeben wird.

DROP

DBPARTITIONNUM *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die aus der Datenbankpartitionsgruppe zu löschen sind. Alle angegebenen Partitionen müssen in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42729). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel DROP DBPARTITIONNUM äquivalent.

DBPARTITIONNUMS *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die aus der Datenbankpartitionsgruppe zu löschen sind. Alle angegebenen Partitionen müssen in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42729). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel DROP DBPARTITIONNUM äquivalent.

TABLE *tabellenname*
Gibt eine Tabellenreihenfolge für die Verarbeitung der Umverteilung an.

ONLY
Wenn der Tabellenreihenfolge das Schlüsselwort **ONLY** folgt (Standardeinstellung), werden nur die angegebenen Tabellen umverteilt. Die verbleibenden Tabellen können später durch Befehle des Typs **REDISTRIBUTE CONTINUE** verarbeitet werden. Dies ist der Standardwert.

FIRST
Wenn der Tabellenreihenfolge das Schlüsselwort **FIRST** folgt, werden die angegebenen Tabellen in der angegebenen Reihenfolge umverteilt. Anschließend werden die verbleibenden Tabellen in der Datenbankpartitionsgruppe in zufälliger Reihenfolge umverteilt.

EXCLUDE *tabellenname*
Gibt Tabellen an, die von der Verarbeitung der Umverteilung ausgeschlossen werden sollen. Sie können beispielsweise zeitweilig eine Tabelle ausschließen, bis diese so konfiguriert ist, dass sie den Anforderungen für die Datenumverteilung entspricht. Die ausgeschlossenen Tabellen können später durch Befehle des Typs **REDISTRIBUTE CONTINUE** verarbeitet werden.

STOP AT *iso-ortszeit*
Wenn diese Option angegeben wird, wird vor Beginn der Datenumverteilung für die einzelnen Tabellen der Wert *iso-ortszeit* mit der aktuellen lokalen Zeitmarke verglichen. Wenn die in *iso-ortszeit* angegebene Zeit gleich der Zeit der aktuellen lokalen Zeitmarke ist oder diese überschritten hat, stoppt das Dienstprogramm mit einer Warnung. Eine zur Stoppzeit in Ausführung befindliche Datenumverteilungsverarbeitung für Tabellen wird ohne Unterbrechung abge-

schlossen. Es wird keine neue Datenumverteilungsverarbeitung für Tabellen gestartet. Die nicht verarbeiteten Tabellen können mit der Option **CONTINUE** umverteilt werden. Der Wert in *iso-ortszeit* wird als Zeitmarke angegeben, das heißt als 7-teilige Zeichenfolge, die eine Kombination aus Datum und Uhrzeit darstellt. Das Format ist *jjjj-mm-tt-hh.mm.ss.nnnnnn* (Jahr, Monat, Tag, Stunde, Minuten, Sekunden, Mikrosekunden) für die Ortszeit.

DATA BUFFER *n*
Gibt die Anzahl von 4-KB-Seiten an, die als Pufferspeicher für die Datenübertragung innerhalb des Dienstprogramms zu verwenden sind. Dieser Befehlsparameter kann nur dann verwendet werden, wenn auch der Parameter **NOT ROLLFORWARD RECOVERABLE** angegeben ist.

Wenn der angegebene Wert kleiner als der unterstützte Mindestwert ist, wird der Mindestwert verwendet und es wird keine Warnung zurückgegeben. Wenn für die Option **DATA BUFFER** kein Wert angegeben wird, wird vom Dienstprogramm zu Beginn der Verarbeitung der einzelnen Tabellen ein intelligenter Standardwert berechnet (während der Laufzeit). Insbesondere werden standardmäßig 50 % der im Zwischenspeicher für Dienstprogramme verfügbaren Speicherkapazität zu Beginn der Umverteilung einer Tabelle verwendet und darüber hinaus verschiedene Eigenschaften der Tabelle berücksichtigt.

Dieser Speicher wird direkt aus dem Zwischenspeicher für Dienstprogramme zugeordnet, dessen Größe durch den Datenbankkonfigurationsparameter **util_heap_sz** geändert werden kann. Der Wert des Parameters **DATA BUFFER** des Befehls **REDISTRIBUTE DATABASE PARTITION GROUP** kann zeitweise den Wert von **util_heap_sz** übersteigen, falls im System mehr Speicher verfügbar ist.

INDEXING MODE
Gibt an, wie Indizes während der Umverteilung gepflegt werden. Dieser Befehlsparameter kann nur dann verwendet werden, wenn auch der Parameter **NOT ROLLFORWARD RECOVERABLE** angegeben ist.

Gültige Werte:

REBUILD
Indizes werden völlig neu erstellt. Indizes müssen bei Verwendung dieser Option nicht gültig sein. Durch diese Option werden Indizes auf der Platte zu Clustern zusammengefasst.

DEFERRED
Die REDISTRIBUTE-Operation versucht nicht, Indizes zu pflegen. Indizes werden als zu aktualisierend markiert. Ein Rebuild solcher Indizes kann beim ersten Zugriff erzwungen werden oder er erfolgt, wenn die Datenbank erneut gestartet wird.

Anmerkung: Wenn ungültige Indizes vorhanden sind, dann erstellt der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** die Indizes bei Nicht-MDC- und Nicht-ITC-Tabellen automatisch erneut, sofern Sie nicht **INDEXING MODE DEFERRED** angegeben haben. Bei einer MDC- oder ITC-Tabelle wird ein ungültiger zusammengesetzter Index, auch wenn Sie **INDEXING MODE DEFERRED** angeben, vor Beginn der Tabellenumverteilung erneut erstellt, weil das Dienstprogramm den zusammengesetzten Index zur Verarbeitung einer MDC- oder ITC-Tabelle benötigt.

PRECHECK
Überprüft, ob die Datenbankpartitionsgruppe umverteilt werden kann. Dieser Befehlsparameter kann nur dann verwendet werden, wenn auch der Parameter **NOT ROLLFORWARD RECOVERABLE** angegeben ist.

YES
Dies ist der Standardwert. Die Umverteilung wird nur dann gestartet, wenn die Überprüfung erfolgreich war. Schlägt die Überprüfung fehl, wird der Befehl beendet und es wird eine Fehlernachricht zurückgegeben, die sich auf den ersten fehlgeschlagenen Prüfwert bezieht.

NO Die Umverteilung wird sofort gestartet, und es findet keine Überprüfung statt.

ONLY
Der Befehl wird nach der Überprüfung beendet. Es findet keine Umverteilung statt. Standardmäßig wird kein Quiesce für die Datenbank durchgeführt. Wenn der Befehl **QUIESCE DATABASE** auf YES gesetzt wurde oder standardmäßig den Wert YES angenommen hat, verbleibt die Datenbank im Quiescemodus. Um die Verbindung zur Datenbank wiederherzustellen, führen Sie die Umverteilungsoperation durch oder setzen Sie den Befehl **UNQUIESCE DATABASE** ab.

QUIESCE DATABASE
Gibt an, dass alle Benutzer die Datenbank verlassen müssen und versetzt diese in den Quiescemodus. Dieser Befehlsparameter kann nur dann verwendet werden, wenn auch der Parameter **NOT ROLLFORWARD RECOVERABLE** angegeben ist.

YES
Dies ist der Standardwert. Nur Benutzer mit der Berechtigung SYSADM, SYSMAINT oder SYSCTRL oder Benutzer mit der Berechtigung QUIESCE_CONNECT haben Zugriff auf die Datenbank oder deren Objekte. Sobald die Umverteilung erfolgreich abgeschlossen ist, wird für die Datenbank ein Unquiesce durchgeführt.

NO Die Umverteilungsoperation versetzt die Datenbank nicht in den Quiescemodus, und es werden keine Benutzer gezwungen, die Datenbank zu verlassen.

Weitere Informationen finden Sie im Abschnitt zum Befehl **QUIESCE DATABASE**.

STATISTICS
Gibt an, dass das Dienstprogramm Statistiken für die Tabellen erfassen soll, die ein Statistikprofil haben. Dieser Befehlsparameter kann nur dann verwendet werden, wenn auch der Parameter **NOT ROLLFORWARD RECOVERABLE** angegeben ist.

Die Angabe dieser Option ist effizienter als eine separate Ausführung des Befehls **RUNSTATS** nach Abschluss der Datenumverteilung.

USE PROFILE
Statistiken werden für die Tabellen mit einem Statistikprofil erfasst. Für Tabellen ohne Statistikprofil erfolgt keine Statistikerfassung. Dies ist der Standardwert.

NONE
Es werden keine Statistiken für Tabellen erfasst.

Beispiele

Das folgende Beispiel zeigt eine Umverteilung der Datenbankpartitionsgruppe DBPG_1, bei der die aktuelle Datenverteilung über eine Datenverteilungsdatei mit dem Namen distfile_for_dbpg_1 angegeben wird. Verschieben Sie die Daten in zwei neue Datenbankpartitionen (6 und 7).

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
   USING DISTFILE /home/user1/data/distfile_for_dbpg_1
   ADD DATABASE PARTITION (6 TO 7)
```

Verteilen Sie die Datenbankpartitionsgruppe DBPG_2 so um, dass folgende Umstände eintreten:

- Die Umverteilung ist nicht aktualisierend wiederherstellbar.
- Daten werden gleichmäßig auf Hashpartitionen verteilt.
- Indizes werden völlig neu erstellt.
- Es werden keine statistischen Daten erfasst.
- 180000 4-KB-Seiten werden als Pufferspeicher für die Übertragung der Daten verwendet.

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_2
 NOT ROLLFORWARD RECOVERABLE
 UNIFORM
 INDEXING MODE REBUILD
 DATA BUFFER 180000
 STATISTICS NONE
```

Aufgrund der Standardwerte für die Befehlsparameter **QUIESCE DATABASE** und **PRECHECK** versetzt diese Umverteilungsoperation außerdem die Datenbank in den Quiescemodus und führt eine Vorabprüfung aus.

Hinweise

- Stellen Sie vor dem Starten einer Umverteilungsoperation sicher, dass sich die Tabellen im Normalstatus, und nicht im Status 'Load Pending' (Laden anstehend) oder 'REORG Pending' (REORG anstehend), befinden. Der Status von Tabellen kann mithilfe des Befehls **LOAD QUERY** überprüft werden.
- Wenn die Option **NOT ROLLFORWARD RECOVERABLE** angegeben wird und die Datenbank eine wiederherstellbare Datenbank ist, wird ein Tabellenbereich, wenn das Dienstprogramm zum ersten Mal auf ihn zugreift, in den Status 'Backup anstehend' versetzt. Alle Tabellen in diesem Tabellenbereich werden schreibgeschützt, bis ein Backup des Tabellenbereichs durchgeführt wird. Das Backup kann erst stattfinden, wenn die Umverteilung aller Tabellen in diesem Tabellenbereich abgeschlossen ist.
- Wenn eine Umverteilungsoperation aktiv ist, generiert sie eine Ereignisprotokolldatei, die allgemeine Informationen zur Umverteilungsoperation und Informationen wie zum Beispiel zur Start- und Endzeit der Verarbeitung der einzelnen Tabellen enthält. Diese Ereignisprotokolldatei wird an folgende Position geschrieben:
  - Der gültige Wert für das Zeitlimit für Sperren Das Verzeichnis `'homeinst/sqllib/redist'` unter Linux- und UNIX-Betriebssystemen, wobei das folgende Format für Unterverzeichnisse und den Dateinamen verwendet wird: *datenbankname*`.`*dbpartitionsgruppenname*`.`*zeitmarke*`.log`.
  - Das Verzeichnis **DB2INSTPROF**`\`*instanz*`\redist` unter Windows-Betriebssystemen (**DB2INSTPROF** ist der Wert der Registrierdatenbankvariablen **DB2INSTPROF**), wobei das folgende Format für Unterverzeichnisse und den Dateinamen verwendet wird: *datenbankname*`.`*dbpartitionsgruppenname*`.`*zeitmarke*`.log`.
  - Der Wert der Zeitmarke gibt den Zeitpunkt an, zu dem der Befehl abgesetzt wurde.
- Dieses Dienstprogramm führt während der Verarbeitung zwischendurch Commitoperationen aus.
- Alle Pakete, die eine Abhängigkeit von einer Tabelle haben, die umverteilt wurde, werden ungültig gemacht. Es wird empfohlen, solche Pakete nach Abschluss

der Umverteilung der Datenbankpartitionsgruppe explizit mit REBIND erneut zu binden. Durch den expliziten Rebind wird die Anfangsverzögerung der ersten SQL-Anforderung für das ungültige Paket vermieden. Die Nachrichtendatei der Umverteilung enthält eine Liste aller Tabellen, die umverteilt wurden.

- Standardmäßig aktualisiert das Dienstprogramm REDISTRIBUTE die Statistiken für die Tabellen, die ein Statistikprofil haben. Für Tabellen ohne Statistikprofil wird empfohlen, die Tabellen- und Indexstatistiken durch Aufrufen der API db2Runstats oder durch Ausführen des Befehls **RUNSTATS** nach Abschluss der Umverteilungsoperation separat zu aktualisieren.
- Datenbankpartitionsgruppen, die replizierte MQTs (Materialized Query Tables) oder Tabellen mit definierter Option DATA CAPTURE CHANGES enthalten, können nicht umverteilt werden.
- Eine Umverteilung ist nicht zulässig, wenn Tabellenbereiche für temporäre Benutzertabellen mit vorhandenen deklarierten temporären Tabellen oder erstellte temporäre Tabellen in der Datenbankpartitionsgruppe enthalten sind.
- Optionen wie **INDEXING MODE** werden für Tabellen, auf die sie nicht anwendbar sind, ohne Warnung ignoriert. Zum Beispiel wird **INDEXING MODE** für Tabellen ignoriert, für die keine Indizes definiert sind.
- Der Befehl **REDISTRIBUTE DATABASE PARTITION GROUP** schlägt möglicherweise fehl (SQLSTATE 55071), wenn eine Datenbankpartitionsserveranforderung ansteht oder ausgeführt wird. Der Befehl schlägt möglicherweise ebenfalls fehl (SQLSTATE 55077), wenn ein neuer Datenbankpartitionsserver online zu der Instanz hinzugefügt wird und der neue Datenbankpartitionsserver nicht allen Anwendungen bekannt ist.

Kompatibilitäten

Tabellen mit XML-Spalten, die das XML-Satzformat von DB2 Version 9.5 oder älter verwenden, können nicht umverteilt werden. Verwenden Sie die gespeicherte Prozedur ADMIN_MOVE_TABLE, um die Tabelle in das neue Format zu migrieren.

db2nchg - Konfiguration des Datenbankpartitionsservers ändern

Ändert die Konfiguration des Datenbankpartitionsservers. Dieser Schritt umfasst das Verschieben des Datenbankpartitionsservers von einer Maschine auf eine andere, das Ändern des TCP/IP-Hostnamens der Maschine und die Auswahl einer anderen logischen Portnummer oder eines anderen Netznamens für den Datenbankpartitionsserver.

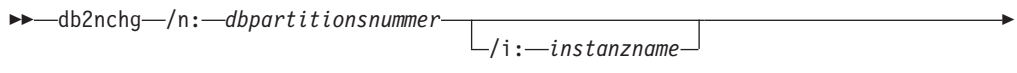
Dieser Befehl kann nur verwendet werden, wenn der Datenbankpartitionsserver gestoppt ist.

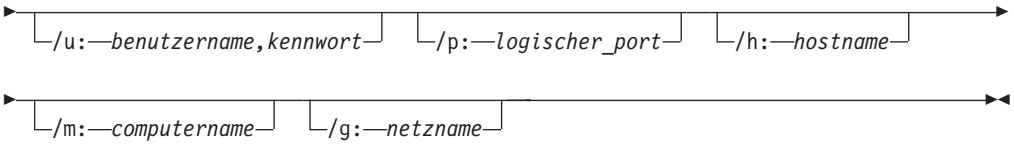
Dieser Befehl steht nur unter Windows-Betriebssystemen zur Verfügung.

Berechtigung

Lokaler Administrator

Befehlssyntax





Befehlsparameter

/n:*dbpartitionsnummer*
Gibt die Datenbankpartitionsnummer der Konfiguration des Datenbankpartitionsservers an, die zu ändern ist.

/i:*instanzname*
Gibt die Instanz an, an der dieser Datenbankpartitionsserver beteiligt ist. Wenn kein Parameter angegeben wird, wird standardmäßig die aktuelle Instanz angenommen.

/u:*benutzername,kennwort*
Gibt den Benutzernamen und das Kennwort an. Wenn kein Parameter angegeben wird, gelten standardmäßig der vorhandene Benutzername und das vorhandene Kennwort.

/p:*logischer_port*
Gibt den logischen Port für den Datenbankpartitionsserver an. Dieser Parameter muss angegeben werden, um den Datenbankpartitionsserver auf ein anderes System zu versetzen. Wenn dieser Parameter nicht angegeben wird, bleibt die logische Portnummer unverändert.

/h:*hostname*
Gibt den TCP/IP-Hostnamen an, der von FCM zur internen Kommunikation verwendet wird. Wenn dieser Parameter nicht angegeben wird, bleibt der Hostname gleich.

/m:*computername*
Gibt den Computer an, auf dem sich der Datenbankpartitionsserver befinden soll. Der Datenbankpartitionsserver kann nur dann versetzt werden, wenn die Instanz keine bereits definierten Datenbanken enthält.

/g:*netzname*
Ändert den Netznamen des Datenbankpartitionsservers. Dieser Parameter kann dazu verwendet werden, eine bestimmte IP-Adresse auf den Datenbankpartitionsserver anzuwenden, wenn auf einem Computer mehrere IP-Adressen verfügbar sind. Es kann der Netzname oder die IP-Adresse eingegeben werden.

Beispiele

Der im folgenden Beispiel gezeigte Befehl ändert den logischen Port, der Datenbankpartition 2 zugeordnet ist, die an der Instanz TESTMPP beteiligt ist, in den logischen Port 3:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

db2ncrt - Datenbankpartitionsserver einer Instanz hinzufügen

Fügt den Datenbankpartitionsserver einer Instanz hinzu.

Dieser Befehl steht nur unter Windows-Betriebssystemen zur Verfügung.

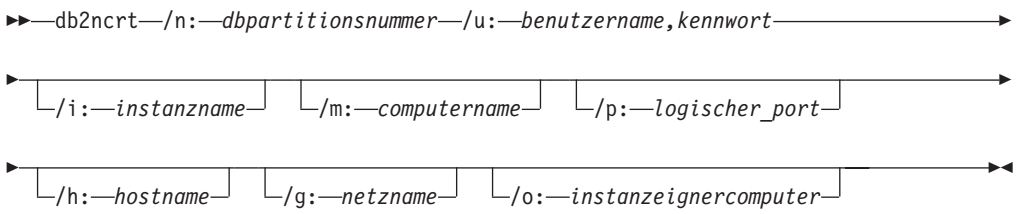
Geltungsbereich

Wenn ein Datenbankpartitionsserver einem Computer hinzugefügt wird, auf dem bereits eine Instanz vorhanden ist, wird der Datenbankpartitionsserver dem Computer als logischer Datenbankpartitionsserver hinzugefügt. Wenn ein Datenbankpartitionsserver einem Computer hinzugefügt wird, auf dem keine Instanz vorhanden ist, wird die Instanz hinzugefügt und der Computer wird zu einem neuen physischen Datenbankpartitionsserver. Dieser Befehl sollte nicht verwendet werden, wenn in einer Instanz Datenbanken vorhanden sind. Stattdessen sollte der Befehl **START DATABASE MANAGER** mit der Option **ADD DBPARTITIONNUM** abgesetzt werden. Dadurch wird sichergestellt, dass die Datenbank dem neuen Datenbankpartitionsserver ordnungsgemäß hinzugefügt wird. Es ist auch möglich, einen Datenbankpartitionsserver einer Instanz hinzuzufügen, in der bereits eine Datenbank erstellt wurde. Die Datei `db2nodes.cfg` darf nicht bearbeitet werden, da eine Änderung der Datei Inkonsistenzen in der Umgebung mit partitionierten Datenbanken zur Folge haben könnte.

Berechtigung

Berechtigung eines `lokalen Administrators` auf dem Computer, auf dem der Datenbankpartitionsserver hinzugefügt wird.

Befehlssyntax



Befehlsparameter

/n:*dbpartitionsnummer*
: Eine eindeutige Datenbankpartitionsnummer, die den Datenbankpartitionsserver angibt. Die eingegebene Nummer kann im Bereich zwischen `1` und `999` liegen.

/u:*benutzername,kennwort*
: Gibt den Namen des Anmeldekontos und das Kennwort für DB2 an.

/i:*instanzname*
: Gibt den Instanznamen an. Wenn kein Parameter angegeben wird, wird standardmäßig die aktuelle Instanz angenommen.

/m:*computername*
: Gibt den Computernamen der Windows-Workstation an, auf der sich der Datenbankpartitionsserver befindet. Dieser Parameter ist erforderlich, wenn ein Datenbankpartitionsserver auf einem fernen Computer hinzugefügt wird.

/p:*logischer_port*
: Gibt die logische Portnummer an, die für den Datenbankpartitionsserver verwendet wird. Wenn dieser Parameter nicht angegeben wird, wird der logischen Portnummer der Wert `0` zugeordnet. Bei der Erstellung eines logischen Datenbankpartitionsservers muss dieser Parameter angegeben wer-

den und es muss eine logische Portnummer ausgewählt werden, die nicht im Gebrauch ist. Beachten Sie die folgenden Einschränkungen:

- Jeder Computer muss einen Datenbankpartitionsserver haben, der den logischen Port 0 hat.
- Die Portnummer darf den für die FCM-Kommunikation im Verzeichnis `x:\winnt\system32\drivers\etc\` reservierten Portbereich nicht überschreiten. Wenn zum Beispiel ein Bereich von vier Ports für die aktuelle Instanz reserviert ist, ist die höchste Portnummer 3. Port 0 wird für den logischen Standarddatenbankpartitionsserver verwendet.

/h:*hostname*
Gibt den TCP/IP-Hostnamen an, der von FCM zur internen Kommunikation verwendet wird. Dieser Parameter ist erforderlich, wenn ein Datenbankpartitionsserver auf einem fernen Computer hinzugefügt wird.

/g:*netzname*
Gibt den Netznamen für den Datenbankpartitionsserver an. Wenn kein Parameter angegeben wird, wird standardmäßig die erste IP-Adresse verwendet, die auf dem System festgestellt wird. Dieser Parameter kann dazu verwendet werden, eine bestimmte IP-Adresse auf den Datenbankpartitionsserver anzuwenden, wenn auf einem Computer mehrere IP-Adressen verfügbar sind. Es kann der Netzname oder die IP-Adresse eingegeben werden.

/o:*instanzeignercomputer*
Gibt den Computernamen des Computers an, der Eigner der Instanz ist. Der Standardwert ist der lokale Computer. Dieser Parameter ist erforderlich, wenn der Befehl **db2ncrt** auf einem Computer aufgerufen wird, der nicht der Instanzeignercomputer ist.

Beispiele

Das folgende Beispiel zeigt einen Befehl, mit dem ein neuer Datenbankpartitionsserver der Instanz TESTMPP auf dem Instanzeignercomputer SHAYER hinzugefügt wird, wobei der neue Datenbankpartitionsserver als Datenbankpartition 2 bekannt ist und den logischen Port 1 verwendet:

```
db2ncrt /n:2 /u:QBPAULZ\paulz,g1reeky /i:TESTMPP /m:TEST /p:1 /o:SHAYER /h:TEST
```

db2ndrop - Datenbankpartitionsserver aus einer Instanz löschen

Löscht einen Datenbankpartitionsserver aus einer Instanz, in der keine Datenbanken definiert sind. Wenn ein Datenbankpartitionsserver gelöscht wird, kann die Datenbankpartitionsnummer für einen neuen Datenbankpartitionsserver verwendet werden.

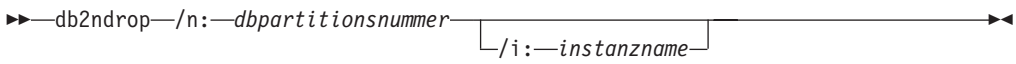
Dieser Befehl kann nur verwendet werden, wenn der Datenbankpartitionsserver gestoppt ist.

Dieser Befehl steht nur unter Windows-Betriebssystemen zur Verfügung.

Berechtigung

Berechtigung eines `lokalen Administrators` auf dem System, auf dem der Datenbankpartitionsserver gelöscht werden soll.

Befehlssyntax



Befehlsparameter

/n:*dbpartitionsnummer*
Eine eindeutige Datenbankpartitionsnummer, die den Datenbankpartitionsserver angibt.

/i:*instanzname*
Gibt den Instanznamen an. Wenn kein Parameter angegeben wird, wird standardmäßig die aktuelle Instanz angenommen.

Beispiele

```
db2ndrop /n:2 /i=KMASCI
```

Hinweise

Wenn der Instanzeigner-Datenbankpartitionsserver (Datenbankpartitionsnummer 0) aus der Instanz gelöscht wird, wird die Instanz dadurch unbrauchbar. Verwenden Sie zum Löschen der Instanz den Befehl **db2idrop**.

Dieser Befehl sollte nicht verwendet werden, wenn sich Datenbanken in der Instanz befinden. Stattdessen sollte der Befehl **db2stop drop dbpartitionnum** verwendet werden. Dadurch wird sichergestellt, dass der Datenbankpartitionsserver ordnungsgemäß aus der Umgebung mit partitionierten Datenbanken entfernt wird. Es ist auch möglich, einen Datenbankpartitionsserver in einer Instanz zu löschen, in der eine Datenbank vorhanden ist. Die Datei `db2nodes.cfg` darf nicht bearbeitet werden, da eine Änderung der Datei Inkonsistenzen in der Umgebung mit partitionierten Datenbanken zur Folge haben könnte.

Wenn ein Datenbankpartitionsserver, der dem logischen Port 0 zugeordnet ist, von einem System gelöscht werden soll, auf dem mehrere logische Datenbankpartitionsserver ausgeführt werden, müssen zunächst alle anderen Datenbankpartitionsserver gelöscht werden, die den anderen logischen Ports zugeordnet sind. Jeder Datenbankpartitionsserver muss einen Datenbankpartitionsserver haben, der dem logischen Port 0 zugeordnet ist.

Kapitel 24. SQL-Sprachelemente

Datentypen

Datenbankpartitionskompatible Datentypen

Die *Datenbankpartitionskompatibilität* wird zwischen den Basisdatentypen entsprechender Spalten von Verteilungsschlüsseln definiert. Datenbankpartitionskompatible Datentypen haben die Eigenschaft, dass zwei Variablen mit jeweils einem dieser Datentypen und dem gleichen Wert durch eine bestimmte Partitionierungsfunktion auf denselben Verteilungszuordnungsindex abgebildet werden.

Tabelle 42 auf Seite 448 zeigt die Kompatibilität von Datentypen in Datenbankpartitionen.

Die Datenbankpartitionskompatibilität besitzt folgende Merkmale:

- Für die Datentypen DATE, TIME und TIMESTAMP werden interne Formate verwendet. Sie sind untereinander nicht kompatibel, und keiner dieser Typen ist mit Zeichen- oder Grafikdatentypen kompatibel.
- Die Partitionskompatibilität wird nicht von der Optionalität der Dateneingabe für eine Spalte beeinflusst.
- Die Partitionskompatibilität wird von der Sortierfolge beeinflusst. Die localeabhängigen, UCA-basierten Sortierfolgen benötigen eine exakte Übereinstimmung in der Sortierfolge, das Attribut 'Strength (S)' der Sortierfolge wird hierbei ignoriert. Alle anderen Sortierfolgen werden im Hinblick auf das Feststellen der Partitionskompatibilität als identisch betrachtet.
- Zeichenspalten, die mit FOR BIT DATA definiert wurden, sind nur dann mit Zeichenspalten ohne FOR BIT DATA kompatibel, wenn eine andere Sortierfolge als eine localeabhängige, UCA-basierte Sortierfolge verwendet wird.
- Nullwerte kompatibler Datentypen werden auf identische Weise behandelt. Für Nullwerte nicht kompatibler Datentypen werden möglicherweise unterschiedliche Ergebnisse erzielt.
- Bei benutzerdefinierten Datentypen (UDT) wird der Basisdatentyp zur Analyse der Datenbankpartitionskompatibilität verwendet.
- Zeitmarken desselben Werts im Verteilungsschlüssel werden gleich behandelt, auch wenn ihre Genauigkeit unterschiedlich ist.
- Dezimalzahlen desselben Werts im Verteilungsschlüssel werden gleich behandelt, auch wenn ihre Anzahl an Kommastellen und ihre Genauigkeit unterschiedlich sind.
- Folgende Leerzeichen in Zeichenfolgen (CHAR, VARCHAR, GRAPHIC oder VARGRAPHIC) werden von der im System bereitgestellten Hashfunktion ignoriert.
- Wird eine localeabhängige, UCA-basierte Sortierfolge verwendet, sind CHAR, VARCHAR, GRAPHIC und VARGRAPHIC kompatible Datentypen. Werden andere Sortierfolgen verwendet, sind CHAR und VARCHAR kompatible Typen und GRAPHIC und VARGRAPHIC sind kompatible Typen, aber CHAR und VARCHAR sind keine mit GRAPHIC und VARGRAPHIC kompatiblen Typen. Die Datentypen CHAR oder VARCHAR unterschiedlicher Längen sind kompatible Datentypen.

- Gleiche DECFLOAT-Werte werden identisch behandelt, selbst wenn ihre Genauigkeit unterschiedlich ist. DECFLOAT-Werte, die numerisch gleich sind, werden identisch behandelt, auch wenn sie unterschiedlich viele signifikante Ziffern haben.
- Datentypen, die als Teil eines Verteilungsschlüssels nicht unterstützt werden, sind in Bezug auf die Datenbankpartitionskompatibilität irrelevant. Dazu gehören Spalten mit den Datentypen BLOB, CLOB, DBCLOB, XML sowie Spalten eines einzigartigen Typs, der auf einem dieser Datentypen basiert, und Spalten strukturierten Typs.

Tabelle 42. Datenbankpartitionskompatibilitäten

Operanden	Binäre Ganzzahl	Dezimalzahl	Gleitkommazahl	Dezimale Gleitkommazahl	Zeichenfolge	Grafikzeichenfolge	Datum	Zeit	Zeitmarke	Einzigartiger Datentyp
Binäre Ganzzahl	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	[1]
Dezimalzahl	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	[1]
Gleitkommazahl	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	[1]
Dezimale Gleit- komma- zahl	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	[1]
Zeichenfolge	Nein	Nein	Nein	Nein	Ja[2]	2, 3	Nein	Nein	Nein	[1]
Grafik- zeichen- folge	Nein	Nein	Nein	Nein	2, 3	Ja[2]	Nein	Nein	Nein	[1]
Datum	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein	Nein	[1]
Zeit	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein	[1]
Zeitmarke	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	[1]
Einzigartiger Datentyp	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]

Anmerkung:

[1] Ein einzigartiger Typ ist datenbankpartitionskompatibel mit dem Quellendatentyp des einzigartigen Typs und allen anderen einzigartigen Typen desselben Quellendatentyps. Bei dem Quellendatentyp des einzigartigen Typs muss es sich um einen Datentyp handeln, der als Teil eines Verteilungsschlüssels unterstützt wird. Ein benutzerdefinierter einzigartiger Datentyp (UDT) ist datenbankpartitionskompatibel mit dem Quellentyp des UDT oder jedes anderen UDT mit einem datenbankpartitionskompatiblen Quellentyp. Ein einzigartiger Typ kann nicht auf dem Datentyp BLOB, CLOB, DBCLOB oder XML basieren.

[2] Zeichen- und Grafikzeichenfolgetypen sind kompatibel, wenn sie kompatible Sortierfolgen haben.

[3] Zeichen- und Grafikzeichenfolgetypen sind kompatibel, wenn eine localeabhängige, UCA-basierte Sortierfolge verwendet wird. Ansonsten sind sie keine kompatible Typen.

Sonderregister

CURRENT MEMBER

Das Sonderregister CURRENT MEMBER gibt einen ganzzahligen Wert (Integer) an, der das Koordinatormember für die Anweisung identifiziert.

Für Anweisungen, die von einer Anwendung abgesetzt werden, ist der Koordinator das Member, zu dem die Anwendung eine Verbindung herstellt. Für Anweisungen, die von einer Routine abgesetzt werden, ist der Koordinator das Member, von dem aus die Routine aufgerufen wurde.

Bei Verwendung in einer SQL-Anweisung einer Routine wird CURRENT MEMBER nie von der aufrufenden Anweisung übernommen.

CURRENT MEMBER gibt den Wert 0 zurück, wenn die Datenbankinstanz nicht zur Unterstützung von IBM DB2 pureScale Feature oder der Datenbankpartitionierung definiert ist. Die Datenbankinstanz ist nicht für die Unterstützung dieser Umgebungen definiert, wenn keine 'db2nodes.cfg' vorhanden ist. Für eine partitionierte Datenbank oder eine DB2 pureScale-Umgebung ist die Datei 'db2nodes.cfg' vorhanden und enthält die Definitionen der Datenbankpartition und des Members.

CURRENT MEMBER kann mit der Anweisung CONNECT geändert werden. Dies ist jedoch nur unter bestimmten Umständen möglich.

Aus Gründen der Kompatibilität mit früheren Versionen von DB2 und mit anderen Datenbankprodukten kann an Stelle von MEMBER auch NODE angegeben werden.

Beispiele

Beispiel 1: Mit der folgenden Anweisung wird die Hostvariable APPL_NODE (Integer) auf die Nummer des Members gesetzt, mit dem die Anwendung verbunden ist.

```
VALUES CURRENT MEMBER
  INTO :APPL_NODE
```

Beispiel 2: Der folgende Befehl wird auf Member 0 und in einem System mit vier Membern in einer Umgebung mit partitionierten Datenbanken abgesetzt. Diese Abfrage ruft die Membernummer der aktuell verbundenen Datenbank ab.

```
db2 "values current member"

1
-----------
          0
```

Kapitel 25. SQL-Funktionen

DATAPARTITIONNUM

Die Funktion DATAPARTITIONNUM gibt die Folgenummer (SYSDATAPARTITIONS.SEQNO) der Datenpartition zurück, in der sich die Zeile befindet.

Das Schema ist SYSIBM.

spaltenname
Der qualifizierte oder nicht qualifizierte Name einer Spalte in der Tabelle. Auf Grund der Tatsache, dass Informationen auf Zeilenebene zurückgegeben werden, ist das Ergebnis stets das gleiche, unabhängig davon, welche Spalte angegeben wird. Die Spalte kann einen beliebigen Datentyp besitzen.

Wenn durch *spaltenname* eine Spalte in einer Sicht angegeben wird, muss der Ausdruck für die Spalte in der Sicht auf eine Spalte der zugrunde liegenden Basistabelle verweisen, und die Sicht muss löschfähig sein. Für einen verschachtelten oder allgemeinen Tabellenausdruck gelten dieselben Regeln wie für eine Sicht.

Datenpartitionen werden nach Bereich sortiert, wobei die Folgenummern bei 0 beginnen. Zum Beispiel gibt die Funktion DATAPARTITIONNUM den Wert 0 für eine Zeile zurück, die sich in der Datenpartition mit dem niedrigsten Bereich befindet.

Der Datentyp des Ergebnisses ist INTEGER, und das Ergebnis ist nie NULL.

Hinweise

- Diese Funktion kann beim Erstellen einer benutzerdefinierten Funktion nicht als Quellenfunktion verwendet werden. Da die Funktion jeden Datentyp als Argument akzeptiert, ist es nicht erforderlich, zusätzliche Funktionskennungen zur Unterstützung benutzerdefinierter einzigartiger Datentypen zu erstellen.
- Die Funktion DATAPARTITIONNUM kann weder in Prüfungen auf Integritätsbedingungen noch in der Definition von generierten Spalten verwendet werden (SQLSTATE-Wert 42881). Die Funktion DATAPARTITIONNUM kann nicht in der Definition einer MQT (Materialized Query Table) verwendet werden (SQLSTATE-Wert 428EC).
- Die Funktion DATAPARTITIONNUM kann nicht als Teil eines ausdrucksbasierten Schlüssels in einer Anweisung CREATE INDEX verwendet werden.

Beispiele

- *Beispiel 1:* Rufen Sie die Folgenummer der Datenpartition ab, in der sich die Zeile für EMPLOYEE.EMPNO befindet.

```
SELECT DATAPARTITIONNUM (EMPNO)
FROM EMPLOYEE
```

- *Beispiel 2:* Zum Konvertieren einer Folgenummer, die von der Funktion DATAPARTITIONNUM zurückgegeben wird (zum Beispiel 0), in einen Datenpartitionsnamen, der in anderen SQL-Anweisungen (z. B. ALTER TABLE...DETACH PARTITION) verwendet werden kann, können Sie die Katalogsicht SYSCAT.DA-

TAPARTITIONS abfragen. Fügen Sie die durch die Funktion DATAPARTITIONNUM abgerufene Folgenummer (SEQNO) in die Klausel WHERE ein, wie im folgenden Beispiel gezeigt:

```
SELECT DATAPARTITIONNAME
   FROM SYSCAT.DATAPARTITIONS
   WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
```

Diese Abfrage liefert zum Beispiel den Wert 'PART0'.

DBPARTITIONNUM

Die Funktion DBPARTITIONNUM gibt die Datenbankpartitionsnummer für eine Zeile zurück. Wenn sie zum Beispiel in einer Klausel SELECT verwendet wird, gibt sie die Datenbankpartitionsnummer für jede Zeile in der Ergebnismenge zurück.

Das Schema ist SYSIBM.

spaltenname
Der qualifizierte oder nicht qualifizierte Name einer Spalte in der Tabelle. Auf Grund der Tatsache, dass Informationen auf Zeilenebene zurückgegeben werden, ist das Ergebnis stets das gleiche, unabhängig davon, welche Spalte angegeben wird. Die Spalte kann einen beliebigen Datentyp besitzen.

Wenn durch *spaltenname* eine Spalte in einer Sicht angegeben wird, muss der Ausdruck für die Spalte in der Sicht auf eine Spalte der zugrunde liegenden Basistabelle verweisen, und die Sicht muss löschfähig sein. Für einen verschachtelten oder allgemeinen Tabellenausdruck gelten dieselben Regeln wie für eine Sicht.

Die bestimmte Zeile (und Tabelle), für die die Datenbankpartitionsnummer von der Funktion DBPARTITIONNUM zurückgegeben wird, wird aus dem Kontext der SQL-Anweisung ermittelt, die diese Funktion verwendet.

Die Datenbankpartitionsnummer, die für Übergangsvariablen und Übergangstabellen zurückgegeben wird, wird aus den aktuellen Übergangswerten der Verteilungsschlüsselspalten abgeleitet. In einem INSERT-Vortrigger gibt die Funktion zum Beispiel die projektierte Datenbankpartitionsnummer zurück, die anhand der aktuellen Werte der neuen Übergangsvariablen ermittelt wird. Die Werte der Verteilungsschlüsselspalten können jedoch von einem nachfolgenden INSERT-Vortrigger geändert werden. Daher ist es möglich, dass die endgültige Datenbankpartitionsnummer der Zeile, wenn sie in die Datenbank eingefügt wird, vom projektierten Wert abweicht.

Der Datentyp des Ergebnisses ist INTEGER, und das Ergebnis ist nie NULL. Wenn keine Datei `db2nodes.cfg` vorhanden ist, ist das Ergebnis 0.

Hinweise

- Die Funktion DBPARTITIONNUM kann nicht für replizierte Tabellen, in Prüfungen auf Integritätsbedingungen oder in der Definition von generierten Spalten verwendet werden (SQLSTATE-Wert 42881).
- Die Funktion DBPARTITIONNUM kann beim Erstellen einer benutzerdefinierten Funktion nicht als Quellenfunktion verwendet werden. Da sie jeden Datentyp als

Argument akzeptiert, ist es nicht erforderlich, zusätzliche Funktionskennungen zur Unterstützung benutzerdefinierter einzigartiger Datentypen zu erstellen.

- Die Funktion DBPARTITIONNUM kann nicht als Teil eines ausdrucksbasierten Schlüssels in einer Anweisung CREATE INDEX verwendet werden.
- ***Syntaxalternativen:*** Aus Gründen der Kompatibilität mit früheren Versionen von DB2-Produkten ist der Funktionsname NODENUMBER ein Synonym für DBPARTITIONNUM.

Beispiele

- *Beispiel 1*: Im folgenden Beispiel wird die Anzahl der Instanzen gezählt, in denen sich die Zeile für einen bestimmten Mitarbeiter in der Tabelle EMPLOYEE in einer anderen Datenbankpartition befindet als die Beschreibung der Abteilung des Mitarbeiters in der Tabelle DEPARTMENT.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- *Beispiel 2*: Im folgenden Beispiel wird ein Join der Tabellen EMPLOYEE und DEPARTMENT ausgeführt, bei dem sich die Zeilen der beiden Tabellen in derselben Datenbankpartition befinden.

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- *Beispiel 3*: Im folgenden Beispiel wird ein Vortrigger für die Tabelle EMPLOYEE verwendet, um die Mitarbeiternummer und die projektierte Datenbankpartitionsnummer einer beliebigen neuen Zeile für die Tabelle EMPLOYEE in einer Tabelle mit dem Namen EMPINSERTLOG1 zu protokollieren.

```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AS NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

Kapitel 26. SQL-Anweisungen

ALTER DATABASE PARTITION GROUP

Die Anweisung ALTER DATABASE PARTITION GROUP wird verwendet, um einer Datenbankpartitionsgruppe eine oder mehrere Datenbankpartitionen(en) hinzuzufügen oder aus einer Datenbankpartitionsgruppe eine oder mehrere Datenbankpartition(en) zu löschen.

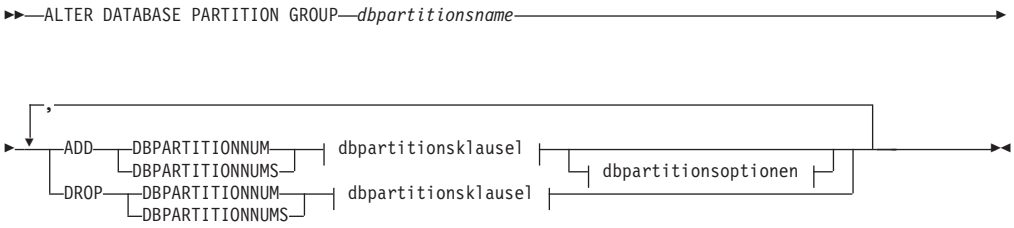
Aufruf

Diese Anweisung kann in ein Anwendungsprogramm eingebettet oder interaktiv abgesetzt werden. Es handelt sich um eine ausführbare Anweisung, die nur dann dynamisch vorbereitet werden kann, wenn das Ausführungsverhalten DYNAMICRULES für das Paket aktiviert ist (SQLSTATE-Wert 42509).

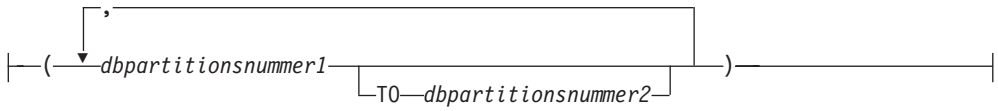
Berechtigung

Die Berechtigungs-ID der Anweisung muss über die Berechtigung SYSCTRL oder SYSADM verfügen.

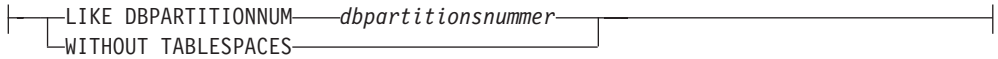
Syntax



dbpartitionsklausel:



dbpartitionsoptionen:



Beschreibung

dbpartitionsname
Gibt den Namen für die Datenbankpartitionsgruppe an. Diese Name besteht aus einem Teil. Es handelt sich um eine SQL-ID (normal oder mit Begrenzern). Er muss eine Datenbankpartitionsgruppe angeben, die im Katalog beschrieben ist. IBMCATGROUP und IBMTEMPGROUP können nicht angegeben werden (SQLSTATE-Wert 42832).

ADD DBPARTITIONNUM
Gibt die bestimmte Datenbankpartition bzw. die Datenbankpartitionen an, die der Datenbankpartitionsgruppe hinzugefügt werden sollen. DBPARTITIONNUMS ist synonym zu DBPARTITIONNUM. Alle angegebenen Datenbankpartitionen dürfen noch nicht in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42728).

DROP DBPARTITIONNUM
Gibt die bestimmte Datenbankpartition bzw. die Datenbankpartitionen an, die aus der Datenbankpartitionsgruppe gelöscht werden sollen. DBPARTITIONNUMS ist synonym zu DBPARTITIONNUM. Alle angegebenen Datenbankpartitionen müssen in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42729).

dbpartitionsklausel
Gibt die Datenbankpartition bzw. die Datenbankpartitionen an, die hinzuzufügen bzw. zu löschen sind.

dbpartitionsnummer1
Geben Sie eine bestimmte Datenbankpartitionsnummer an.

TO *dbpartitionsnummer2*
Geben Sie einen Bereich von Datenbankpartitionsnummern an. Der Wert von *dbpartitionsnummer2* muss größer oder gleich dem Wert von *dbpartitionsnummer1* sein (SQLSTATE-Wert 428A9).

dbpartitionsoptionen

LIKE DBPARTITIONNUM *dbpartitionsnummer*
Gibt an, dass die Container für die vorhandenen Tabellenbereiche in der Datenbankpartitionsgruppe die gleichen sein sollen wie die Container in der Datenbankpartition *dbpartitionsnummer*. Die angegebene Datenbankpartition muss eine Partition sein, die in der Datenbankpartitionsgruppe vor dieser Anweisung vorhanden war und die nicht in einer Klausel DROP DBPARTITIONNUM derselben Anweisung angegeben ist.

Für Tabellenbereiche, die zur Verwendung von dynamischem Speicher definiert sind (d. h. Tabellenbereiche, die mit der Klausel MANAGED BY AUTOMATIC STORAGE der Anweisung CREATE TABLESPACE erstellt wurden bzw. für die überhaupt keine Klausel MANAGED BY angegeben wurde), stimmen die Container nicht unbedingt mit denen aus der angegebenen Partition überein. Vielmehr werden Container automatisch vom Datenbankmanager auf der Basis von Speicherpfaden zugeordnet, die der Datenbank zugeordnet sind. Dies kann dazu führen, dass dieselben Container verwendet werden, oder auch nicht. Die Größe jedes Tabellenbereichs hängt von der Anfangsgröße ab, die beim Erstellen des Tabellenbereichs angegeben wurde, und stimmt möglicherweise nicht mit der aktuellen Größe des Tabellenbereichs in der angegebenen Partition überein.

WITHOUT TABLESPACES
Gibt an, dass die Container für vorhandene Tabellenbereiche in der Datenbankpartitionsgruppe in der neu hinzugefügten Datenbankpartition (bzw. solchen Partitionen) nicht erstellt werden. Container zur Verwendung mit den Tabellenbereichen, die in dieser Datenbankpartitionsgruppe definiert sind, müssen mit der Anweisung ALTER TABLESPACE mit der Klausel *dbpartitionsklausel* oder der Klausel MANAGED BY AUTOMATIC STORAGE definiert werden. Wenn diese Option nicht angegeben wird, werden in neu hinzugefügten Datenbankpartitionen für jeden Tabellenbereich, der in der Datenbankpartitionsgruppe definiert ist, die Standardcontainer angegeben.

Diese Option wird für Tabellenbereiche ignoriert, die zur Verwendung von dynamischem Speicher definiert sind (d. h. Tabellenbereiche, die mit der Klausel MANAGED BY AUTOMATIC STORAGE der Anweisung CREATE TABLESPACE erstellt wurden bzw. für die überhaupt keine Klausel MANAGED BY angegeben wurde). Es gibt keine Möglichkeit, die Erstellung von Containern für diese Tabellenbereiche zu verzögern. Container werden automatisch vom Datenbankmanager auf der Basis von Speicherpfaden zugeordnet, die der Datenbank zugeordnet sind. Die Größe jedes Tabellenbereichs hängt von der Anfangsgröße ab, die beim Erstellen des Tabellenbereichs angegeben wurde.

Regeln

- Jede Datenbankpartition, die durch eine Nummer angegeben wird, muss in der Datei `db2nodes.cfg` definiert sein (SQLSTATE-Wert 42729).
- Jeder Wert für *dbpartitionnummer* in der Klausel *dbpartitionsklausel* muss eine eindeutige Datenbankpartition angeben (SQLSTATE-Wert 42728).
- Eine gültige Datenbankpartitionsnummer liegt im Bereich zwischen 0 und 999 einschließlich (SQLSTATE-Wert 42729).
- Eine Datenbankpartition kann nicht sowohl in der Klausel ADD als auch in der Klausel DROP angegeben werden (SQLSTATE-Wert 42728).
- Es muss mindestens eine Datenbankpartition in der Datenbankpartitionsgruppe verbleiben. Die letzte Datenbankpartition kann nicht aus einer Datenbankpartitionsgruppe gelöscht werden (SQLSTATE-Wert 428C0).
- Wenn weder die Klausel LIKE DBPARTITIONNUM noch die Klausel WITHOUT TABLESPACES beim Hinzufügen einer Datenbankpartition angegeben wird, wird standardmäßig die niedrigste Datenbankpartitionsnummer der vorhandenen Datenbankpartitionen in der Datenbankpartitionsgruppe verwendet. Wenn diese zum Beispiel *2* ist, wird die Operation so fortgesetzt, als wäre LIKE DBPARTITIONNUM *2* angegeben worden. Damit eine vorhandene Datenbankpartition als Standarddatenbankpartition verwendet werden kann, müssen für sie Container für alle Tabellenbereiche in der Datenbankpartitionsgruppe definiert sein (die Spalte IN_USE der Katalogsicht SYSCAT.DBPARTITIONGROUPDEF hat nicht den Wert 'T').
- Die Anweisung ALTER DATABASE PARTITION GROUP schlägt möglicherweise fehl (SQLSTATE 55071), wenn eine Datenbankpartitionsserveranforderung ansteht oder ausgeführt wird. Die Anweisung schlägt möglicherweise ebenfalls fehl (SQLSTATE 55077), wenn ein neuer Datenbankpartitionsserver online zu der Instanz hinzugefügt wird und der neue Datenbankpartitionsserver nicht allen Anwendungen bekannt ist.

Hinweise

- Wenn einer Datenbankpartitionsgruppe eine Datenbankpartition hinzugefügt wird, wird ein Katalogeintrag für die Datenbankpartition gespeichert (Katalogsicht SYSCAT.DBPARTITIONGROUPDEF). Die Verteilungszuordnung wird unverzüglich geändert, um die neue Datenbankpartition einzufügen, und ein Anzeiger (IN_USE) gibt an, dass die Datenbankpartition in der Verteilungszuordnung enthalten ist, sofern eine der folgenden Bedingungen zutrifft:
  - Es sind keine Tabellenbereiche in der Datenbankpartitionsgruppe definiert.
  - Es sind keine Tabellen in den Tabellenbereichen definiert, die in der Datenbankpartitionsgruppe definiert sind, und die Klausel WITHOUT TABLESPACES wurde nicht angegeben.

Die Verteilungszuordnung wird nicht geändert und der Anzeiger (IN_USE) wird so eingestellt, dass er angibt, dass die Datenbankpartition nicht in der Verteilungszuordnung enthalten ist, wenn eine der folgenden Bedingungen zutrifft:

– Es sind Tabellen in Tabellenbereichen in der Datenbankpartitionsgruppe vorhanden.
– Es sind Tabellenbereiche in der Datenbankpartitionsgruppe enthalten und die Klausel WITHOUT TABLESPACES wurde angegeben (sofern nicht alle Tabellenbereiche zur Verwendung von dynamischem Speicher definiert sind, sodass die Klausel WITHOUT TABLESPACES ignoriert wird).

Zum Ändern der Verteilungszuordnung muss der Befehl REDISTRIBUTE DATABASE PARTITION GROUP verwendet werden. Dieser Befehl verteilt alle Daten um, ändert die Verteilungszuordnung und ändert den Anzeiger (IN_USE). Tabellenbereichscontainer müssen hinzugefügt werden, bevor versucht wird, Daten umzuverteilen, wenn die Klausel WITHOUT TABLESPACES angegeben wurde.

- Wenn eine Datenbankpartition aus einer Datenbankpartitionsgruppe gelöscht wird, wird der Katalogeintrag (Katalogsicht SYSCAT.DBPARTITIONGROUPDEF) für die Datenbankpartition aktualisiert. Wenn in den Tabellenbereichen, die in der Datenbankpartitionsgruppe definiert sind, keine Tabellen definiert sind, wird die Verteilungszuordnung unverzüglich geändert, um die gelöschte Datenbankpartition auszuschließen, und der Eintrag für die Datenbankpartition in der Datenbankpartitionsgruppe wird gelöscht. Wenn Tabellen vorhanden sind, wird die Verteilungszuordnung nicht geändert und der Anzeiger (IN_USE) wird so eingestellt, dass er angibt, dass die Datenbankpartition zum Löschen ansteht. Der Befehl REDISTRIBUTE DATABASE PARTITION muss zur Umverteilung der Daten und zum Löschen des Eintrags für die Datenbankpartition aus der Datenbankpartitionsgruppe verwendet werden.
- ***Syntaxalternativen***: Die nachfolgend aufgeführten Syntaxalternativen werden aus Gründen der Kompatibilität mit früheren DB2-Versionen und mit anderen Datenbankprodukten unterstützt. Diese Alternativen stellen keinen Standard dar und sollten daher nicht verwendet werden.
  – Anstelle von DBPARTITIONNUM kann das Schlüsselwort NODE angegeben werden.
  – Anstelle von DBPARTITIONNUMS kann das Schlüsselwort NODES angegeben werden.
  – Anstelle von DATABASE PARTITION GROUP kann das Schlüsselwort NODEGROUP angegeben werden.

Beispiel

Nehmen Sie an, Sie haben eine partitionierte Datenbank mit den folgenden sechs Datenbankpartitionen: 0, 1, 2, 5, 7 und 8. Zwei Datenbankpartitionen (3 und 6) werden dem System hinzugefügt.

- *Beispiel 1*: Wenn Sie die Datenbankpartitionen 3 und 6 einer Datenbankpartitionsgruppe mit dem Namen MAXGROUP hinzufügen wollen, wobei die Tabellenbereichscontainer wie die in der Datenbankpartition 2 sein sollen, sieht die Anweisung wie folgt aus:

```
ALTER DATABASE PARTITION GROUP MAXGROUP
  ADD DBPARTITIONNUMS (3,6)LIKE DBPARTITIONNUM 2
```

- *Beispiel 2*: Nehmen Sie an, Sie wollen Datenbankpartition 1 löschen und Datenbankpartition 6 der Datenbankpartitionsgruppe mit dem Namen MEDGROUP hinzufügen. Außerdem definieren Sie die Tabellenbereichscontainer für Datenbankpartition 6 mithilfe einer separaten Anweisung ALTER TABLESPACE. Dazu würden Sie die folgende Anweisung verwenden:

```
ALTER DATABASE PARTITION GROUP MEDGROUP
  ADD DBPARTITIONNUM(6)WITHOUT TABLESPACES
  DROP DBPARTITIONNUM(1)
```

CREATE DATABASE PARTITION GROUP

Die Anweisung CREATE DATABASE PARTITION GROUP definiert eine neue Datenbankpartitionsgruppe in der Datenbank, ordnet der Datenbankpartitionsgruppe Datenbankpartitionen zu und speichert die Definition der Datenbankpartitionsgruppe im Systemkatalog.

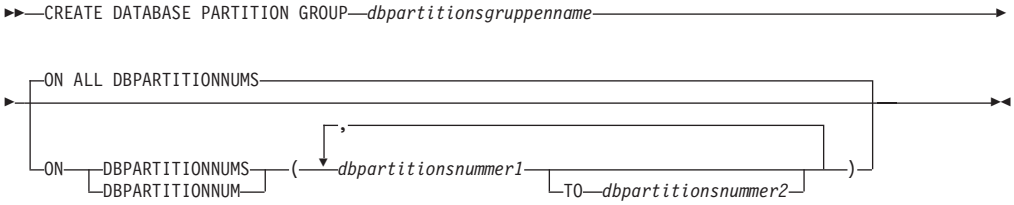
Aufruf

Diese Anweisung kann in ein Anwendungsprogramm eingebettet oder interaktiv abgesetzt werden. Es handelt sich um eine ausführbare Anweisung, die nur dann dynamisch vorbereitet werden kann, wenn das Ausführungsverhalten DYNAMICRULES für das Paket aktiviert ist (SQLSTATE-Wert 42509).

Berechtigung

Die Zugriffsrechte der Berechtigungs-ID der Anweisung müssen die Berechtigung SYSCTRL oder SYSADM mit einschließen.

Syntax



Beschreibung

dbpartitionsgruppenname
Gibt den Namen für die Datenbankpartitionsgruppe an. Diese Name besteht aus einem einzigen Teil. Es handelt sich um eine SQL-ID (normal oder mit Begrenzern). Der durch *dbpartitionsgruppenname* angegebene Wert darf keine Datenbankpartitionsgruppe bezeichnen, die bereits im Katalog vorhanden ist (SQLSTATE-Wert 42710). Der Wert für *dbpartitionsgruppenname* darf nicht mit den Zeichen 'SYS' oder 'IBM' beginnen (SQLSTATE-Wert 42939).

ON ALL DBPARTITIONNUMS
Gibt an, dass die Datenbankpartitionsgruppe über alle Datenbankpartitionen definiert wird, die in der Datenbank zu dem Zeitpunkt definiert sind (Datei `db2nodes.cfg`), zu dem die Datenbankpartitionsgruppe erstellt wird.

Wenn dem Datenbanksystem eine Datenbankpartition hinzugefügt wird, sollte die Anweisung ALTER DATABASE PARTITION GROUP so abgesetzt werden, dass sie die neue Datenbankpartition in einer Datenbankpartitionsgruppe (einschließlich IBMDEFAULTGROUP) mit einschließt. Darüber hinaus muss der Befehl REDISTRIBUTE DATABASE PARTITION GROUP ausgeführt werden, um Daten in die Datenbankpartition zu versetzen.

ON DBPARTITIONNUMS
Gibt die Datenbankpartitionen an, die in der Datenbankpartitionsgruppe enthalten sind. DBPARTITIONNUM ist synonym zu DBPARTITIONNUMS.

dbpartitionsnummer1
Geben Sie eine Datenbankpartitionsnummer an. (Aus Gründen der Kompatibilität mit früheren Versionen kann ein *Knotenname* der Form `NODEnnnnn` angegeben werden.)

TO *dbpartitionsnummer2*
Geben Sie einen Bereich von Datenbankpartitionsnummern an. Der Wert von *dbpartitionsnummer2* muss größer oder gleich dem Wert von *dbpartitionsnummer1* sein (SQLSTATE-Wert 428A9). Alle Datenbankpartitionen aus dem Bereich der angegebenen Datenbankpartitionsnummern werden in die Datenbankpartitionsgruppe aufgenommen.

Regeln

- Jede Datenbankpartition, die durch eine Nummer angegeben wird, muss in der Datei `db2nodes.cfg` definiert sein (SQLSTATE-Wert 42729).
- Jede Datenpartition *dbpartitionsnummer*, die in der Klausel ON DBPARTITIONNUMS aufgeführt wird, darf höchstens einmal angegeben werden (SQLSTATE-Wert 42728).
- Eine gültige *Datenbankpartitionsnummer* liegt im Bereich zwischen 0 und 999 einschließlich (SQLSTATE-Wert 42729).
- Die Anweisung CREATE DATABASE PARTITION GROUP schlägt möglicherweise fehl (SQLSTATE 55071), wenn eine Datenbankpartitionsserveranforderung ansteht oder ausgeführt wird. Die Anweisung schlägt möglicherweise ebenfalls fehl (SQLSTATE 55077), wenn ein neuer Datenbankpartitionsserver online zu der Instanz hinzugefügt wird und der neue Datenbankpartitionsserver nicht allen Anwendungen bekannt ist.

Hinweise

- Diese Anweisung erstellt eine Verteilungszuordnung für die Datenbankpartitionsgruppe. Für jede Verteilungszuordnung wird eine Verteilungszuordnungs-ID (PMAP_ID) generiert. Diese Information wird im Katalog gespeichert und kann aus den Katalogsichten SYSCAT.DBPARTITIONGROUPS und SYSCAT.PARTITIONMAPS abgerufen werden. Jeder Eintrag in der Verteilungszuordnung gibt die Zieldatenbankpartition an, in der sich alle Zeilen nach der Verarbeitung durch ein Hashverfahren befinden. Für eine Datenbankpartitionsgruppe mit einer Einzelpartition enthält die entsprechende Verteilungszuordnung nur einen Eintrag. Für eine Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen enthält die entsprechende Verteilungszuordnung 32768 Einträge, wobei die Datenbankpartitionsnummern den Zuordnungseinträgen standardmäßig in einem Umlaufverfahren zugeordnet werden.
- ***Syntaxalternativen***: Die nachfolgend aufgeführten Syntaxalternativen werden aus Gründen der Kompatibilität mit früheren DB2-Versionen und mit anderen Datenbankprodukten unterstützt. Diese Alternativen stellen keinen Standard dar und sollten daher nicht verwendet werden.
  - Anstelle von DBPARTITIONNUM kann das Schlüsselwort NODE angegeben werden.
  - Anstelle von DBPARTITIONNUMS kann das Schlüsselwort NODES angegeben werden.
  - Anstelle von DATABASE PARTITION GROUP kann das Schlüsselwort NODEGROUP angegeben werden.

Beispiele

Die folgenden Beispiele gehen von einer partitionierten Datenbank mit sechs Datenbankpartitionen aus, die als 0, 1, 2, 5, 7 und 8 definiert sind.

- *Beispiel 1*: Angenommen, Sie wollen eine Datenbankpartitionsgruppe mit dem Namen MAXGROUP für alle sechs Datenbankpartitionen erstellen. Dazu würden Sie die folgende Anweisung verwenden:

  ```
  CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
  ```

- *Beispiel 2*: Wenn Sie eine Datenbankpartitionsgruppe mit dem Namen MEDGROUP für die Datenbankpartitionen 0, 1, 2, 5 und 8 erstellen wollen, würden Sie folgende Anweisung verwenden:

  ```
  CREATE DATABASE PARTITION GROUP MEDGROUP
    ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
  ```

- *Beispiel 3*:Wenn Sie eine Datenbankpartitionsgruppe mit nur einer Datenbankpartition und dem Namen MINGROUP für die Datenbankpartition 7 erstellen wollen, würden Sie folgende Anweisung verwenden:

  ```
  CREATE DATABASE PARTITION GROUP MINGROUP
    ON DBPARTITIONNUM (7)
  ```

Kapitel 27. Unterstützte SQL-Verwaltungsroutinen und Verwaltungssichten

Gespeicherte Prozedur ADMIN_CMD und zugeordnete SQL-Verwaltungsroutinen

GET STMM TUNING (Befehl) mit Verwendung der Prozedur ADMIN_CMD

Dient zum Lesen der Katalogtabellen, um die Nummer des vom Benutzer bevorzugten Optimierungsmembers des Speichermanagers für automatische Leistungsoptimierung (STMM) und die aktuelle Nummer des STMM-Optimierungsmembers abzurufen.

Berechtigung

Die Berechtigungs-ID der Anweisung muss mindestens eine der folgenden Berechtigungen aufweisen:

- DBADM
- SECADM
- SQLADM
- ACCESSCTRL
- DATAACCESS
- SELECT für SYSIBM.SYSTUNINGINFO

Erforderliche Verbindung

Datenbank

Befehlssyntax

Beispiel

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

Das folgende Beispiel zeigt die Ausgabe dieser Abfrage.

```
Ergebnismenge 1
  --------------

  USER_PREFERRED_NUMBER CURRENT_NUMBER
  --------------------- --------------
                      2              2

  1 Satz/Sätze ausgewählt.

  Rückgabestatus = 0
```

Hinweise

- Die Nummer des vom Benutzer bevorzugten Optimierungsmembers des Speichermanagers für automatische Leistungsoptimierung (STMM) (USER_PREFERRED_NUMBER) wird vom Benutzer festgelegt und gibt das Member an, auf dem der Benutzer die Speicheroptimierung ausführen möchte. Während die Datenbank ausgeführt wird, wird das Optimierungsmember einige Male pro Stunde angewendet. Infolgedessen ist es möglich, dass die zurückgegebenen Werte für CURRENT_NUMBER und USER_PREFERRED_NUMBER nach einer Aktualisierung des vom Benutzer bevorzugten STMM-Members nicht synchron sind. Um dies zu beheben, können Sie entweder warten, bis der Wert für CURRENT_NUMBER asynchron aktualisiert wird, oder Sie können die Datenbank stoppen und erneut starten, um die Aktualisierung von CURRENT_NUMBER zu erzwingen.

Kompatibilitäten

Aus Gründen der Kompatibilität mit früheren Versionen können Sie Folgendes ausführen:

- **MEMBER** kann durch **DBPARTITIONNUM** ersetzt werden, außer wenn die Registrierdatenbankvariable **DB2_ENFORCE_MEMBER_SYNTAX** auf ON gesetzt ist.

UPDATE STMM TUNING (Befehl) mit Verwendung der Prozedur ADMIN_CMD

Die vom Benutzer bevorzugte Membernummer der Datenbank, auf dem die Optimierungsfunktion des Speichermanagers für automatische Leistungsoptimierung (STMM) erstellt wird, kann aktualisiert werden.

Berechtigung

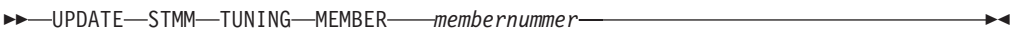
Die Berechtigungs-ID der Anweisung muss mindestens eine der folgenden Berechtigungen aufweisen:

- DBADM
- DATAACCESS
- SQLADM

Erforderliche Verbindung

Datenbank

Befehlssyntax



Befehlsparameter

membernummer
Der Wert von *membernummer* ist eine ganze Zahl.

In einer Umgebung mit partitionierten Datenbanken müssen Sie Folgendes beachten:

- Bei Angabe einer gültigen Membernummer führt der DB2-Datenbankserver die STMM-Speicheroptimierungsfunktion für dieses Member aus.

- Bei Angabe des Werts -1 oder einer nicht vorhandenen Membernummer wählt der DB2-Datenbankserver automatisch ein geeignetes Member zur Ausführung der STMM-Speicheroptimierungsfunktion aus.

In einer DB2 pureScale-Umgebung müssen Sie Folgendes beachten:

- Bei Angabe einer gültigen Membernummer führt der DB2-Datenbankserver die STMM-Speicheroptimierungsfunktion für dieses Member aus.
- Bei Angabe des Werts -2 veranlasst der DB2-Datenbankserver, dass die STMM-Optimierungsfunktionen unabhängig voneinander auf den einzelnen Membern ausgeführt werden.
- Bei Angabe des Werts -1 oder einer nicht vorhandenen Membernummer wählt der DB2-Datenbankserver automatisch ein geeignetes Member zur Ausführung der STMM-Speicheroptimierungsfunktion aus.

Beispiel

In einer Umgebung mit partitionierten Datenbanken wird die vom Benutzer bevorzugte Optimierungsdatenbankpartition des Speichermanagers für automatische Leistungsoptimierung (STMM) in Member 3 aktualisiert.

```
CALL SYSPROC.ADMIN_CMD( 'update stmm tuning member 3' )
```

Hinweise

- Der STMM-Optimierungsprozess überprüft in regelmäßigen Abständen, ob sich der Wert der Nummer für das vom Benutzer bevorzugte STMM-Optimierungsmember geändert hat. Der STMM-Optimierungsprozess wechselt zu dem vom Benutzer bevorzugten STMM-Optimierungsmember, wenn die in *membernummer* angegebene Nummer vorhanden ist und ein aktives Member bezeichnet. Nachdem dieser Befehl die Nummer des STMM-Optimierungsmembers geändert hat, erfolgt unverzüglich eine Änderung an der aktuellen STMM-Optimierungsmembernummer.
- Der Status der Befehlsausführung wird in dem SQL-Kommunikationsbereich (SQLCA) zurückgegeben, der von der Anweisung **CALL** generiert wird.
- Dieser Befehl führt eine Commitoperation für seine Änderungen in der Prozedur **ADMIN_CMD** aus.

Kompatibilitäten

Aus Gründen der Kompatibilität mit früheren Versionen können Sie Folgendes ausführen:

- **MEMBER** kann durch **DBPARTITIONNUM** ersetzt werden, außer wenn die Registrierdatenbankvariable **DB2_ENFORCE_MEMBER_SYNTAX** auf ON gesetzt ist.

SQL-Verwaltungsroutinen und Verwaltungssichten für die Konfiguration

DB_PARTITIONS

Die Tabellenfunktion DB_PARTITIONS gibt den Inhalt der Datei db2nodes.cfg im Tabellenformat zurück.

Anmerkung: Diese Tabellenfunktion gilt als veraltet und wurde durch die Verwaltungssichten DB2_MEMBER und DB2_CF sowie die Tabellenfunktion DB2_GET_INSTANCE_INFO ersetzt.

Syntax

Das Schema ist SYSPROC.

Berechtigung

Zum Ausführen der Routine ist eine der folgenden Berechtigungen erforderlich:

- Zugriffsrecht EXECUTE für die Routine
- Berechtigung DATAACCESS
- Berechtigung DBADM
- Berechtigung SQLADM

Standardzugriffsrecht PUBLIC

Bei einer nicht-restriktiven Datenbank wird der Gruppe PUBLIC das Zugriffsrecht EXECUTE erteilt, wenn die Funktion automatisch erstellt wird.

Tabellenfunktionsparameter

Die Funktion hat keine Eingabeparameter.

Beispiele

Abrufen von Informationen aus einer partitionierten Datenbankinstanz mit 4 Membern.

```
SELECT * FROM TABLE(DB_PARTITIONS()) as T
```

Das folgende Beispiel zeigt die Ausgabe dieser Abfrage:

```
PARTITION_NUMBER HOST_NAME PORT_NUMBER SWITCHNAME
---------------- --------- ----------- -------
               0 so1                 0 so1-ib0
               1 so2                 0 so2-ib0
               2 so3                 0 so3-ib0
               3 so4                 0 so4-ib0

     4 Satz/Sätze ausgewählt.
```

Rufen Sie in einer DB2 pureScale-Umgebung Informationen aus einer DB2 pureScale-Instanz mit 4 Membern und 1 Cluster-Caching-Funktion ab.

```
SELECT * FROM TABLE(DB_PARTITIONS()) as T
```

Das folgende Beispiel zeigt die Ausgabe dieser Abfrage:

```
PARTITION_NUMBER HOST_NAME  PORT_NUMBER SWITCHNAME
---------------- ---------- ----------- ----------
               0 so1                  0 so1-ib0
               0 so2                  0 so2-ib0
               0 so3                  0 so3-ib0

     3 Satz/Sätze ausgewählt.
```

Hinweise zur Verwendung

Für DB2 Enterprise Server Edition und in einer partitionierten Datenbankumgebung gibt die Tabellenfunktion DB_PARTITIONS für jeden Eintrag in der Datei `db2nodes.cfg` eine Zeile zurück.

In einer DB2 pureScale-Umgebung gibt die Tabellenfunktion DB_PARTITIONS mehrere Zeilen zurück, wobei die Spalten folgende Informationen enthalten:

- Die Spalte PARTITION_NUMBER enthält immer 0.
- Die übrigen Spalten zeigen Informationen für den Eintrag in der Datei `db2nodes.cfg` für das aktuelle Member an.

Zurückgegebene Informationen

Tabelle 43. Von der Tabellenfunktion DB_PARTITIONS zurückgegebene Informationen

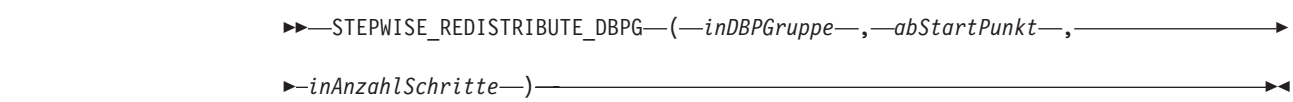
Spaltenname	Datentyp	Beschreibung
PARTITION_NUMBER	SMALLINT	partition_number - Partitionsnummer (Monitorelement)
HOST_NAME	VARCHAR(256)	host_name - Hostname (Monitorelement)
PORT_NUMBER	SMALLINT	Die Portnummer für den Datenbankpartitionsserver.
SWITCH_NAME	VARCHAR(128)	Der Name einer Hochgeschwindigkeitsverbindung bzw. eines Switch für die Kommunikation von Datenbankpartitionen.

SQL-Verwaltungsroutinen zur schrittweisen Datenumverteilung

STEPWISE_REDISTRIBUTE_DBPG (Prozedur) - Teil der Datenbankpartitionsgruppe umverteilen

Die Funktion STEPWISE_REDISTRIBUTE_DBPG führt eine Umverteilung eines Teils der Datenbankpartitionsgruppe entsprechend der Eingabe für die Prozedur und der Einstellungsdatei durch, die mit der Prozedur SET_SWRD_SETTINGS erstellt oder aktualisiert wurde.

Syntax



Das Schema ist SYSPROC.

Prozedurparameter

inDBPGruppe
Ein Eingabeargument des Typs VARCHAR (128), das den Namen der Zieldatenbankpartitionsgruppe angibt.

abStartPunkt
Ein Eingabeargument des Typs SMALLINT, das den zu verwendenden Startpunkt angibt. Wenn der Parameter auf einen positiven ganzzahligen Wert gesetzt wird und nicht NULL ist, verwendet die Prozedur STEPWISE_REDISTRIBUTE_DBPG diesen Wert anstelle des Werts *nextStep*, der in der Einstellungsdatei angegeben ist. Dies ist eine nützliche Option, wenn Sie die Prozedur STEPWISE_REDISTRIBUTE_DBPG ab einem bestimmten Schritt erneut ausführen wollen. Wenn der Parameter auf NULL gesetzt wird, wird der Wert in *nextStep* verwendet.

inAnzahlSchritte
Ein Eingabeargument des Typs SMALLINT, das die Anzahl der auszuführenden Schritte angibt. Wenn der Parameter auf einen positiven ganzzahligen Wert gesetzt wird und nicht NULL ist, verwendet die Prozedur STEPWISE_REDISTRIBUTE_DBPG diesen Wert anstelle des Werts *stageSize*, der in der Einstellungsdatei angegeben ist. Dies ist eine nützliche Option, wenn Sie die Prozedur STEPWISE_REDISTRIBUTE_DBPG mit einer anderen Anzahl von Schritten als die, die in den Einstellungen angegeben ist, erneut ausführen wollen. Wenn zum Beispiel fünf Schritte in einem geplanten Arbeitsabschnitt (Stage) enthalten sind und der Umverteilungsprozess bei Schritt 3 fehlgeschlagen ist, kann die Prozedur STEPWISE_REDISTRIBUTE_DBPG aufgerufen werden, um die übrigen drei Schritte auszuführen, nachdem die Fehlerbedingung behoben wurde. Wenn der Parameter auf NULL gesetzt wird, wird der Wert in *stageSize* verwendet. Der Wert -2 kann in dieser Prozedur verwendet werden, um anzugeben, dass die Anzahl unbegrenzt ist.

Anmerkung: Es gibt keinen Parameter zur Angabe eines entsprechenden Gegenstücks zur Option **NOT ROLLFORWARD RECOVERABLE** im Befehl **REDISTRIBUTE DATABASE PARTITION GROUP**. Die Protokollierung erfolgt immer für die Zeilendatenumverteilung, die ausgeführt wird, wenn die Prozedur STEPWISE_REDISTRIBUTE_DBPG verwendet wird.

Berechtigung

- Zugriffsrecht EXECUTE für die Prozedur STEPWISE_REDISTRIBUTE_DBPG
- SYSADM, SYSCTRL oder DBADM

Standardzugriffsrecht PUBLIC

Bei einer nicht-restriktiven Datenbank wird der Gruppe PUBLIC das Zugriffsrecht EXECUTE erteilt, wenn die Prozedur automatisch erstellt wird.

Beispiel

Das folgende Beispiel zeigt, wie die Datenbankpartitionsgruppe "IBMDEFAULTGROUP" entsprechend dem Umverteilungsplan, der in der Registrierdatenbank durch die Prozedur SET_SWRD_SETTINGS gespeichert wurde, umverteilt wird. Die Prozedur startet mit Schritt 3 und verteilt die Daten um, bis zwei Schritte im Umverteilungsplan ausgeführt sind.

```
CALL SYSPROC.STEPWISE_REDISTRIBUTE_DBPG('IBMDEFAULTGROUP', 3, 2)
```

Ein vollständiges Beispiel für die Verwendung der STEPWISE_REDISTRIBUTE-Prozeduren finden Sie in „Umverteilen von Datenbankpartitionsgruppen mithilfe der Prozedur STEPWISE_REDISTRIBUTE_DBPG” in der Dokumentation *Partitionierung und Clustering*.

Hinweise

Wenn der Registrierdatenbankwert für *processState* mit der Prozedur SET_SWRD-_SETTINGS auf den Wert 1 aktualisiert wird, nachdem die Ausführung der Prozedur STEPWISE_REDISTRIBUTE_DBPG gestartet wurde, wird der Prozess bei Beginn des nächsten Schritts gestoppt und eine Warnung zurückgegeben.

Da von dem Umverteilungsprozess eine SQL-Anweisung COMMIT aufgerufen wird, wird die Ausführung des Umverteilungsprozesses über eine Verbindung des Typs 2 nicht unterstützt.

Teil 6. Anhänge und Schlussteil

Anhang A. Installieren als Benutzer ohne Rootberechtigung

Installieren von DB2-Datenbankservern als Benutzer ohne Rootberechtigung

Die meisten DB2-Datenbankprodukte können von einem Benutzer ohne Rootberechtigung installiert werden.

Vorbereitende Schritte

Bevor Sie ein DB2-Datenbankprodukt als Benutzer ohne Rootberechtigung installieren, sollten Sie die Unterschiede zwischen Installationen mit Rootberechtigung und Installationen ohne Rootberechtigung kennen sowie die Einschränkungen bei Installationen ohne Rootberechtigung. Weitere Informationen zur Installation `ohne Rootberechtigung` finden Sie im Abschnitt „Installation ohne Rootberechtigung (Linux und UNIX) - Übersicht".

Die folgenden Voraussetzungen gelten beim Installieren eines DB2-Datenbankprodukts als Benutzer ohne Rootberechtigung:

- Sie müssen in der Lage sein, die Installations-DVD anzuhängen, oder sie muss bereits angehängt sein.
- Sie müssen über eine gültige Benutzer-ID verfügen, die als Eigner einer DB2-Instanz verwendet werden kann.

  Für Benutzer-IDs gelten die folgenden Einschränkungen und Voraussetzungen:
  - Sie müssen einer anderen primären Gruppe als 'guests', 'admins', 'users' und 'local' angehören.
  - Sie dürfen Kleinbuchstaben (a-z), Zahlen (0-9) und das Unterstreichungszeichen (_) enthalten.
  - Sie dürfen nicht länger als acht Zeichen sein.
  - Sie dürfen nicht mit IBM, SYS, SQL oder einer Zahl beginnen.
  - Sie dürfen kein in DB2 reserviertes Wort (USERS, ADMINS, GUESTS, PUBLIC oder LOCAL) sowie kein reserviertes SQL-Wort sein.
  - Es dürfen keine Benutzer-IDs mit Rootberechtigung als DB2-Instanz-ID, DAS-ID oder abgeschirmte ID verwendet werden.
  - Sie dürfen keine Zeichen mit Akzent enthalten.
  - Wenn keine neuen Benutzer-IDs erstellt, sondern vorhandene Benutzer-IDs verwendet werden, müssen folgende Bedingungen erfüllt sein:
    - Die Benutzer-IDs sind nicht gesperrt.
    - Die Kennwörter der Benutzer-IDs sind nicht abgelaufen.
- Die bestehenden Hardware- und Softwarevoraussetzungen für das Produkt das Sie installieren, gelten für Benutzer ohne Rootberechtigung genauso wie für Rootbenutzer.
- Unter AIXmuss die asynchrone E/A (Asynchronous I/O, AIO) aktiviert sein.E s wird ausdrücklich empfohlen, beim System die E/A-Abschlussports (I/O Completion Ports, IOCP) zu aktivieren.
- Ihr Ausgangsverzeichnis muss ein gültiger DB2-Pfad sein.

  Für DB2-Installationspfade gelten die folgenden Regeln:

– Sie dürfen Kleinbuchstaben (a-z), Großbuchstaben (A-Z) und das Unterstreichungszeichen (_) enthalten.
– Sie dürfen nicht länger als 128 Zeichen sein.
– Sie dürfen keine Leerzeichen enthalten.
– Sie dürfen keine Sonderzeichen der jeweiligen Landessprache enthalten.

Informationen zu diesem Vorgang

Das Installieren von DB2-Datenbankprodukten als Benutzer ohne Rootberechtigung ist für den Benutzer ohne Rootberechtigung transparent. Anders ausgedrückt: Das einzige, was ein Benutzer ohne Rootberechtigung ausführen muss, um ein DB2-Datenbankprodukt installieren zu können, ist die Anmeldung als Benutzer ohne Rootberechtigung.

Vorgehensweise

Gehen Sie wie folgt vor, um eine nicht als Root ausgeführte Installation durchzuführen:

1. Melden Sie sich als Benutzer ohne Rootberechtigung an
2. Installieren Sie das gewünschte DB2-Datenbankprodukt anhand einer der zur Verfügung stehenden Methoden. Zu den verfügbaren Optionen gehören die folgenden:
   - Der **DB2-Installationsassistent** (GUI-Installationsprogramm)
   - Der Befehl **db2setup** mit einer Antwortdatei (unbeaufsichtigte Installation Installation)

     Anmerkung: Da Benutzer ohne Rootberechtigung kein Installationsverzeichnis für DB2-Datenbankprodukte auswählen können, werden alle Vorkommen des Schlüsselworts **FILE** in Ihrer Antwortdatei ignoriert.
3. Nach der Installation des DB2-Datenbankprodukts müssen Sie eine neue Anmeldesitzung öffnen, um die DB2-Nicht-Rootinstanz zu verwenden. Sie können stattdessen auch dieselbe Anmeldesitzung verwenden, wenn Sie in der DB2-Instanzumgebung den Pfad *$HOME*/sqllib/db2profile (für Benutzer der Bourne-Shell und Korn-Shell) oder *$HOME*/sqllib/db2chsrc (für Benutzer der C-Shell) einrichten. Hierbei ist *$HOME* das Ausgangsverzeichnis des Benutzers ohne Rootberechtigung.

Nächste Schritte

Nach der Installation des DB2-Datenbankprodukts müssen Sie die Begrenzungen für Benutzerprozessressourcen (ulimits) Ihres Betriebssystems prüfen. Wenn die Mindestwerte für 'ulimit' nicht erreicht werden, kann es in der DB2-Steuerkomponente zu unerwarteten Engpässen bei den Betriebsressourcen kommen. Diese Fehler können einen Ausfall des DB2-Datenbanksystems zur Folge haben.

Anhang B. Verwenden von Backup

Backup von Daten

Verwenden Sie den Befehl **BACKUP DATABASE**, um eine Kopie der Daten einer Datenbank zu erstellen und sie auf einem anderen Datenträger zu speichern. Die Backup-Daten können im Falle eines Verlusts der Originaldaten oder eines Fehlers verwendet werden.

Sie können eine gesamte Datenbank, eine Datenbankpartition oder nur ausgewählte Tabellenbereiche mit Backup sichern.

Vorbereitende Schritte

Es braucht noch keine Verbindung zu der zu sichernden Datenbank zu bestehen: Das Backup-Datenbankdienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Backup-Operation beendet wird. Wenn bereits eine Verbindung zu der zu sichernden Datenbank besteht, wird diese Verbindung beim Absetzen des Befehls **BACKUP DATABASE** unterbrochen, und die Backup-Operation wird fortgesetzt.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln. Das Backup-Image bleibt auf dem Datenbankserver, sofern Sie kein Speicherverwaltungsprodukt wie Tivoli Storage Manager (TSM) oder DB2 Advanced Copy Services (ACS) verwenden.

Wenn Sie ein Offline-Backup durchführen und Sie die Datenbank mit dem Befehl **ACTIVATE DATABASE** aktiviert haben, müssen Sie die Datenbank inaktivieren, bevor Sie das Offline-Backup durchführen. Wenn die Datenbank über aktive Verbindungen verfügt, muss ein Benutzer mit der Berechtigung SYSADM eine Verbindung zu der Datenbank herstellen und die folgenden Befehle absetzen, um die Datenbank erfolgreich zu inaktivieren:

```
CONNECT TO aliasname_der_datenbank
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE aliasname-der-datenbank
```

In einer Umgebung mit partitionierten Datenbanken haben Sie die folgenden Möglichkeiten: Mit dem Befehl **BACKUP DATABASE** können Sie Datenbankpartitionen einzeln sichern, mit dem Befehlsparameter **ON DBPARTITIONNUM** können Sie mehrere der Datenbankpartitionen auf einmal sichern und mit dem Parameter **ALL DBPARTITIONNUMS** können Sie alle Datenbankpartitionen gleichzeitig sichern. Mit dem Befehl **LIST DBPARTITIONNUMS** können Sie die Datenbankpartitionen identifizieren, in denen Benutzertabellen enthalten sind, für die das Ausführen eines Backups sinnvoll ist.

Wenn Sie ein *Offline-Backup* in einer Umgebung mit partitionierten Datenbanken ausführen und kein SSV-Backup (SSV = Single System View, Einzelsystemsicht) verwenden, sollten Sie die Katalogpartition von allen anderen Datenbankpartitionen getrennt sichern. Zum Beispiel können Sie die Katalogpartition zuerst sichern und anschließend die anderen Datenbankpartitionen sichern. Diese Aktion ist notwendig, weil die Backup-Operation möglicherweise eine exklusive Datenbankverbindung mit der Katalogpartition erfordert, während deren die anderen Daten-

bankpartitionen keine Verbindung herstellen können. Wenn Sie ein *Online-Backup* ausführen, können alle Datenbankpartitionen (einschließlich der Katalogpartition) gleichzeitig und in beliebiger Reihenfolge gesichert werden.

Auf einem System, in dem mit verteilten Anforderungen gearbeitet wird, werden die Backup-Operationen auf die Datenbank für verteilte Anforderungen sowie auf die Metadaten angewendet, die im Katalog dieser Datenbank gespeichert sind (Wrapper, Server, Kurznamen usw.). Datenquellenobjekte (Tabellen und Sichten) werden nicht gesichert, es sei denn, diese Objekte werden in der Datenbank für verteilte Anforderungen gespeichert.

Wenn eine Datenbank mit einem vorherigen Release des Datenbankmanagers erstellt wurde, aber für sie noch kein Upgrade auf das aktuelle Release durchgeführt wurde, müssen Sie das entsprechende Upgrade für die Datenbank vornehmen. Andernfalls ist es nicht möglich, eine Backup-Kopie der Datenbank zu erstellen.

Einschränkungen

Für das Backup-Dienstprogramm gelten die folgenden Einschränkungen:

- Es ist nicht möglich, gleichzeitig ein Backup eines Tabellenbereichs und einen Restore eines Tabellenbereichs durchzuführen. Dies gilt auch dann, wenn Backup und Restore für unterschiedliche Tabellenbereiche erfolgen.
- Wenn Sie in der Lage sein wollen, in einer Umgebung mit partitionierten Datenbanken eine aktualisierende Recovery durchzuführen, müssen Sie die Datenbank auf den Knoten der Liste regelmäßig sichern. Außerdem müssen Sie über mindestens ein Backup-Image der übrigen Knoten im System verfügen (auch von Knoten, die keine Benutzerdaten für diese Datenbank enthalten). In zwei Situationen ist ein Backup-Image einer Datenbankpartition auf einem Datenbankpartitionsserver erforderlich, der keine Benutzerdaten für die Datenbank enthält:
  - Sie haben dem Datenbanksystem einen Datenbankpartitionsserver hinzugefügt, nachdem das letzte Backup erstellt wurde, und müssen eine aktualisierende Recovery auf diesem Datenbankpartitionsserver durchführen.
  - Die punktuelle Recovery wird verwendet, wozu alle Datenbankpartitionen im System den Status 'aktualisierende Recovery anstehend' aufweisen müssen.
- Online-Backup-Operationen für DMS-Tabellenbereiche sind mit den folgenden Operationen nicht kompatibel:
  - Laden
  - Reorganisation (online und offline)
  - Löschen von Tabellenbereichen
  - Tabellenverkürzung
  - Indexerstellung
  - Verwenden des Parameters NOT LOGGED INITIALLY (wird mit den Anweisungen CREATE TABLE und ALTER TABLE verwendet)
- Wenn Sie versuchen, ein Offline-Backup einer Datenbank durchzuführen, die momentan aktiv ist, empfangen Sie einen Fehler. Sie können vor einem Offline-Backup sicherstellen, dass die Datenbank nicht aktiv ist, indem Sie den Befehl **DEACTIVATE DATABASE** ausführen.

Vorgehensweise

Gehen Sie wie folgt vor, um das Backup-Dienstprogramm aufzurufen:

- Setzen Sie den Befehl **BACKUP DATABASE** im Befehlszeilenprozessor ab.

- Führen Sie die Prozedur ADMIN_CMD mit dem Parameter BACKUP DATABASE aus.
- Verwenden Sie die Anwendungsprogrammierschnittstelle (API) db2Backup.
- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **BACKUP DATABASE**.

Beispiel

Das folgende Beispiel zeigt einen Befehl **BACKUP DATABASE**, der über den CLP abgesetzt wird:

```
db2 backup database sample to c:\DB2Backups
```

Nächste Schritte

Wenn Sie ein Offline-Backup durchgeführt haben, müssen Sie nach der Ausführung die Datenbank reaktivieren:

```
ACTIVATE DATABASE sample
```

Anhang C. Katalogsichten für Umgebungen mit partitionierten Datenbanken

SYSCAT.BUFFERPOOLDBPARTITIONS

Jede Zeile stellt eine Kombination aus einem Pufferpool und einem Member dar, bei der sich die Größe des Pufferpools auf diesem Member von seiner Standardgröße für andere Member in derselben Datenbankpartitionsgruppe (wie in SYSCAT.BUFFERPOOLS dargestellt) unterscheidet.

Tabelle 44. Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS

Spaltenname	Datentyp	Nullwert?	Beschreibung
BUFFERPOOLID	INTEGER		Interne Kennung (ID) für den Pufferpool.
DBPARTITIONNUM	SMALLINT		Membernummer.
NPAGES	INTEGER		Anzahl der Seiten in diesem Pufferpool auf diesem Member.

SYSCAT.DATAPARTITIONEXPRESSION

Jede Zeile stellt einen Ausdruck für den jeweiligen Teil des Tabellenpartitionierungsschlüssels dar.

Tabelle 45. Katalogsicht SYSCAT.DATAPARTITIONEXPRESSION

Spaltenname	Datentyp	Nullwert?	Beschreibung
TABSCHEMA	VARCHAR(128)		Schemaname der partitionierten Tabelle.
TABNAME	VARCHAR(128)		Nicht qualifizierter Name der partitionierten Tabelle.
DATAPARTITIONKEYSEQ	INTEGER		Folgen-ID für den Ausdrucksschlüsselteil, beginnend bei 1.
DATAPARTITIONEXPRESSION	CLOB (32K)		Ausdruck für diesen Eintrag in der Folge, in SQL-Syntax.
NULLSFIRST	CHAR (1)		• N = Nullwerte in diesem Ausdruck werden in Vergleichen als höherer Wert interpretiert. • Y = Nullwerte in diesem Ausdruck werden in Vergleichen als niedrigerer Wert interpretiert.

SYSCAT.DATAPARTITIONS

Jede Zeile stellt eine Datenpartition dar. Dabei ist Folgendes zu beachten: Bei auf mehreren Datenbankpartitionen erstellten Tabellen beziehen sich die Datenpartitionsstatistiken auf mehrere Datenbankpartitionen.

Tabelle 46. Katalogsicht SYSCAT.DATAPARTITIONS

Spaltenname	Datentyp	Nullwert?	Beschreibung
DATAPARTITIONNAME	VARCHAR(128)		Name der Datenpartition.

Tabelle 46. Katalogsicht SYSCAT.DATAPARTITIONS (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
TABSCHEMA	VARCHAR(128)		Schemaname der Tabelle, zu der diese Datenpartition gehört.
TABNAME	VARCHAR(128)		Nicht qualifizierter Name der Tabelle, zu der diese Datenpartition gehört.
DATAPARTITIONID	INTEGER		Kennung (ID) für die Datenpartition.
TBSPACEID	INTEGER	J	Kennung (ID) für den Tabellenbereich, in dem diese Datenpartition gespeichert ist. Der Nullwert, wenn STATUS gleich 'I' ist.
PARTITIONOBJECTID	INTEGER	J	Kennung (ID) für die Datenpartition innerhalb des Tabellenbereichs.
LONG_TBSPACEID	INTEGER	J	Kennung (ID) für den Tabellenbereich, in dem lange Daten (LONG) gespeichert werden. Der Nullwert, wenn STATUS gleich 'I' ist.
ACCESS_MODE	CHAR (1)		Zugriffsbeschränkungsstatus der Datenpartition. Dieser Status bezieht sich nur auf Objekte, die sich im Status 'Festlegen der Integrität anstehend' befinden, bzw. auf Objekte, die von einer Anweisung SET INTEGRITY verarbeitet wurden. Mögliche Werte: • D = Kein Versetzen von Daten • F = Vollzugriff • N = Kein Zugriff • R = Lesezugriff
STATUS	VARCHAR (32)		• A = Datenpartition wurde neu zugeordnet. • D = Zuordnung für Datenpartition wird aufgehoben und abhängigen Objekte müssen inkrementell in Hinblick auf den Inhalt dieser Partition gewartet wer den. • I = Datenpartition mit aufgehobener Zuordnung, deren Eintrag im Katalog nur während der asynchronen Indexbereinigung beibehalten wird. Zeilen mit dem STATUS-Wert 'I' werden entfernt, wenn alle Indexdatensätze, die auf die Partition mit aufgehobener Zuordnung verweisen, gelöscht wurden. • D = Zuordnung der Datenpartition wird logisch aufgehoben. • Leere Zeichenfolge = Datenpartition ist sichtbar (Normalstatus). Byte 2 bis 32 sind für zukünftige Zwecke reserviert.
SEQNO	INTEGER		Folgenummer der Datenpartition (bei 0 beginnend).
LOWINCLUSIVE	CHAR (1)		• N = LOWKEY-Wert ist nicht einschließend. • Y = LOWKEY-Wert ist einschließend.
LOWVALUE	VARCHAR (512)		LOWKEY-Wert (eine Zeichenfolgedarstellung eines SQL-Werts) für diese Datenpartition.

Tabelle 46. Katalogsicht SYSCAT.DATAPARTITIONS (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
HIGHINCLUSIVE	CHAR (1)		• N = HIGHKEY-Wert ist nicht einschließend. • Y = HIGHKEY-Wert ist einschließend.
HIGHVALUE	VARCHAR (512)		HIGHKEY-Wert (eine Zeichenfolgedarstellung eines SQL-Werts) für diese Datenpartition.
CARD	BIGINT		Gesamtzahl von Zeilen in der Datenpartition. '-1', wenn keine Statistikdaten erfasst werden.
OVERFLOW	BIGINT		Gesamtzahl von Überlaufsätzen in der Datenpartition. '-1', wenn keine Statistikdaten erfasst werden.
NPAGES	BIGINT		Gesamtzahl von Seiten, in denen die Zeilen der Datenpartition vorliegen. '-1', wenn keine Statistikdaten erfasst werden.
FPAGES	BIGINT		Gesamtzahl von Seiten in der Datenpartition. '-1', wenn keine Statistikdaten erfasst werden.
ACTIVE_BLOCKS	BIGINT		Gesamtzahl aktiver Blöcke in der Datenpartition oder '-1'. Bezieht sich nur auf MDC-Tabellen.
INDEX_TBSPACEID	INTEGER		Kennung (ID) für den Tabellenbereich, in dem die partitionierten Indizes für diese Datenpartition gespeichert sind.
AVGROWSIZE	SMALLINT		Durchschnittslänge (in Byte) komprimierter und nicht komprimierter Zeilen in dieser Datenpartition. '-1', wenn keine Statistikdaten erfasst werden.
PCTROWSCOMPRESSED	REAL		Prozentsatz komprimierter Zeilen bezogen auf die Gesamtzahl von Zeilen in der Datenpartition. '-1', wenn keine Statistikdaten erfasst werden.
PCTPAGESAVED	SMALLINT		Grobe Angabe zum Prozentsatz von Seiten, die als Ergebnis einer Zeilenkomprimierung in der Datenpartition gespeichert wurden. Dieser Wert beinhaltet zusätzliche Byte für jede einzelne Benutzerdatenzeile der Datenpartition, berücksichtigt jedoch nicht den durch zusätzliche Byte für Wörterverzeichnisse belegten Speicherplatz. '-1', wenn keine Statistikdaten erfasst werden.
AVGCOMPRESSEDROWSIZE	SMALLINT		Durchschnittslänge (in Byte) komprimierter Zeilen in dieser Datenpartition. '-1', wenn keine Statistikdaten erfasst werden.
AVGROWCOMPRESSIONRATIO	REAL		Bei komprimierten Zeilen in der Datenpartition gibt dieser Wert die durchschnittliche Komprimierungsrate pro Zeile an, d. h. die durchschnittliche Länge nicht komprimierter Zeilen dividiert durch die durchschnittliche Länge komprimierter Zeilen. '-1', wenn keine Statistikdaten erfasst werden.

Tabelle 46. Katalogsicht SYSCAT.DATAPARTITIONS (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
STATS_TIME	TIMESTAMP	J	Zeitpunkt, zu dem zum letzten Mal Änderungen an aufgezeichneten Statistiken für dieses Objekt vorgenommen wurden. Null, wenn keine Statistikdaten erfasst werden.
LASTUSED	DATE		Datum, an dem die Datenpartition zuletzt von einer DML-Anweisung oder einem Befehl LOAD verwendet wurde. Diese Spalte wird nicht aktualisiert, wenn die Datenpartition in einer HADR-Bereitschaftsdatenbank verwendet wird. Der Standardwert ist '0001-01-01'. Dieser Wert wird asynchron höchstens einmal alle 24 Stunden aktualisiert und gibt die Nutzung innerhalb der letzten 15 Minuten möglicherweise nicht wieder.

SYSCAT.DBPARTITIONGROUPDEF

Jede Zeile stellt eine Datenbankpartition dar, die in einer Datenbankpartitionsgruppe enthalten ist.

Tabelle 47. Katalogsicht SYSCAT.DBPARTITIONGROUPDEF

Spaltenname	Datentyp	Nullwert?	Beschreibung
DBPGNAME	VARCHAR(128)		Name der Datenbankpartitionsgruppe, in der die Datenbankpartition enthalten ist.
DBPARTITIONNUM	SMALLINT		Partitionsnummer einer Datenbankpartition, die in der Datenbankpartitionsgruppe enthalten ist. Eine gültige Partitionsnummer liegt im Bereich zwischen 0 und 999 (einschließlich).

Tabelle 47. Katalogsicht SYSCAT.DBPARTITIONGROUPDEF (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
IN_USE	CHAR (1)		Status der Datenbankpartition. • A = Die neu hinzufügte Datenbankpartition ist nicht in der Verteilungszuordnung, aber die Container für die Tabellenbereiche in der Datenbankpartitionsgruppe wurden erstellt. Die Datenbankpartition wird der Verteilungszuordnung hinzugefügt, wenn eine Operation zur Umverteilung (REDISTRIBUTE) einer Datenbankpartitionsgruppe erfolgreich ausgeführt wurde. • D = Die Datenbankpartition wird gelöscht, wenn eine Operation zur Umverteilung (REDISTRIBUTE) einer Datenbankpartitionsgruppe erfolgreich ausgeführt wurde. • T = Die neu hinzugefügte Datenbankpartition ist nicht in der Verteilungszuordnung und sie wurde unter Verwendung der Klausel WITHOUT TABLESPACES hinzugefügt. Den Tabellenbereichen in der Datenbankpartitionsgruppe müssen Container hinzugefügt werden. • Y = Die Datenbankpartition ist in der Verteilungszuordnung enthalten.

SYSCAT.DBPARTITIONGROUPS

Jede Zeile stellt eine Datenbankpartitionsgruppe dar.

Tabelle 48. Katalogsicht SYSCAT.DBPARTITIONGROUPS

Spaltenname	Datentyp	Nullwert?	Beschreibung
DBPGNAME	VARCHAR(128)		Name der Datenbankpartitionsgruppe.
OWNER	VARCHAR(128)		Berechtigungs-ID des Eigners der Datenbankpartitionsgruppe.
OWNERTYPE	CHAR (1)		• S = Der Eigner ist das System. • U = Der Eigner ist ein Einzelbenutzer.
PMAP_ID	SMALLINT		Kennung (ID) für die Verteilungszuordnung in der Katalogsicht SYSCAT.PARTITIONMAPS.
REDISTRIBUTE_PMAP_ID	SMALLINT		Kennung (ID) für die Verteilungszuordnung, die momentan für die Umverteilung verwendet wird. Dieser Wert ist -1, wenn die Umverteilung momentan nicht ausgeführt ist.
CREATE_TIME	TIMESTAMP		Zeitpunkt der Erstellung der Datenbankpartitionsgruppe.
DEFINER[1]	VARCHAR(128)		Berechtigungs-ID des Eigners der Datenbankpartitionsgruppe.

Tabelle 48. Katalogsicht SYSCAT.DBPARTITIONGROUPS (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
REMARKS	VARCHAR (254)	J	Kommentare des Benutzers oder der Nullwert.

Anmerkung:

1. Die Spalte DEFINER wird nur aus Gründen der Abwärtskompatibilität bereitgestellt. Siehe Spalte OWNER.

SYSCAT.PARTITIONMAPS

Jede Zeile stellt eine Verteilungszuordnung dar, die zur Verteilung der Zeilen einer Tabelle auf die Datenbankpartitionen in einer Datenbankpartitionsgruppe auf der Basis eines Hashverfahrens zur Tabellenverteilung dient.

Tabelle 49. Katalogsicht SYSCAT.PARTITIONMAPS

Spaltenname	Datentyp	Nullwert?	Beschreibung
PMAP_ID	SMALLINT		Kennung (ID) für die Verteilungszuordnung.
PARTITIONMAP	BLOB (65536)		Verteilungszuordnung. Dabei handelt es sich um einen Vektor aus 32768 ganzzahligen, 2 Byte langen Werten für eine Datenbankpartitionsgruppe mit mehreren Partitionen. Für eine Datenbankpartitionsgruppe mit einer Einzelpartition ist nur ein Eintrag vorhanden, der die Partitionsnummer der Einzelpartition angibt.

Anhang D. Übersicht über technische Informationen zu DB2

Technische Informationen zu DB2 liegen in verschiedenen Formaten vor, die auf unterschiedliche Weise abgerufen werden können.

Die technischen Informationen zu DB2 stehen über die folgenden Tools und Methoden zur Verfügung:

- DB2 Information Center
  - Themen (zu Tasks, Konzepten und Referenzinformationen)
  - Beispielprogramme
  - Lernprogramme
- DB2-Bücher
  - PDF-Dateien (für den Download verfügbar)
  - PDF-Dateien (auf der DB2-PDF-DVD)
  - Gedruckte Bücher
- Hilfe für Befehlszeile
  - Hilfe für Befehle
  - Hilfe für Nachrichten

Anmerkung: Die Themen des DB2 Information Center 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 das DB2 Information Center 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 dazu 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 den DB2-Kundendienst 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/e-business/linkweb/publications/servlet/pbi.wss zur Verfügung steht. Über die folgende Adresse können Sie englische Handbücher im PDF-Format sowie übersetzte Versionen zu DB2 Version 10.1 herunterladen: www.ibm.com/support/docview.wss?rs=71&uid=swg27009474.

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: Das *DB2 Information Center* wird häufiger aktualisiert als die PDF- und Hardcopybücher.

Tabelle 50. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar	Verfügbarkeitsdatum
Administrative API Reference	SC27-5506-00	Ja	28. Juli 2013
Administrative Routines and Views	SC27-5507-00	Nein	28. Juli 2013
Call Level Interface Guide and Reference Volume 1	SC27-5511-00	Ja	28. Juli 2013
Call Level Interface Guide and Reference Volume 2	SC27-5512-00	Ja	28. Juli 2013
Command Reference	SC27-5508-00	Ja	28. Juli 2013
Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen	SC12-4884-00	Ja	28. Juli 2013
Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz	SC12-4902-00	Ja	28. Juli 2013
Datenbanküberwachung - Handbuch und Referenz	SC12-4885-00	Ja	28. Juli 2013
Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz	SC12-4903-00	Ja	28. Juli 2013
Datenbanksicherheit	SC12-4904-00	Ja	28. Juli 2013
DB2 Workload Management - Handbuch und Referenz	SC12-4894-00	Ja	28. Juli 2013
Developing ADO.NET and OLE DB Applications	SC27-4549-00	Ja	28. Juli 2013

Tabelle 50. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Verfügbarkeitsdatum
Developing Embedded SQL Applications	SC27-4550-00	Ja	28. Juli 2013
Developing Java Applications	SC27-5503-00	Ja	28. Juli 2013
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-5504-00	Nein	28. Juli 2013
Developing RDF Applications for IBM Data Servers	SC27-5505-00	Ja	28. Juli 2013
Developing User-defined Routines (SQL and External)	SC27-5501-00	Ja	28. Juli 2013
Getting Started with Database Application Development	GI13-2084-00	Ja	28. Juli 2013
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GI11-3309-00	Ja	28. Juli 2013
Globalisierung	SC12-4905-00	Ja	28. Juli 2013
DB2-Server - Installation	GC12-4888-00	Ja	28. Juli 2013
IBM Data Server-Clients - Installation	GC12-4889-00	Nein	28. Juli 2013
Fehlernachrichten, Band 1	SC12-4897-00	Nein	28. Juli 2013
Fehlernachrichten, Band 2	SC12-4898-00	Nein	28. Juli 2013
Net Search Extender - Verwaltung und Benutzerhandbuch	SC12-4900-00	Nein	28. Juli 2013
Partitionierung und Clustering	SC12-4906-00	Ja	28. Juli 2013
pureXML - Handbuch	SC12-4895-00	Ja	28. Juli 2013
Spatial Extender - Benutzer- und Referenzhandbuch	SC12-4899-00	Nein	28. Juli 2013
SQL Procedural Languages: Application Enablement and Support	SC27-5502-00	Ja	28. Juli 2013
SQL Reference Volume 1	SC27-5509-00	Ja	28. Juli 2013
SQL Reference Volume 2	SC27-5510-00	Ja	28. Juli 2013
Text Search	SC12-4901-00	Ja	28. Juli 2013
Fehlerbehebung und Optimieren der Datenbankleistung	SC12-4886-00	Ja	28. Juli 2013

Tabelle 50. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Verfügbarkeitsdatum
Upgrade auf DB2 Version 10.5	SC12-4887-00	Ja	28. Juli 2013
Neuerungen in DB2 Version 10.5	SC12-4893-00	Ja	28. Juli 2013
XQuery - Referenz	SC12-4896-00	Nein	28. Juli 2013

Tabelle 51. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar	Verfügbarkeitsdatum
DB2 Connect - Installation und Konfiguration von DB2 Connect Personal Edition	SC12-4890-00	Ja	28. Juli 2013
DB2 Connect - Installation und Konfiguration von DB2 Connect-Servern	SC12-4891-00	Ja	28. Juli 2013
DB2 Connect - Benutzerhandbuch	SC12-4892-00	Ja	28. Juli 2013

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2-Produkte geben 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.

Vorgehensweise

Zum Starten der Hilfe für SQL-Statuswerte müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? SQL-Status oder ? Klassencode
```

Hierbei steht *SQL-Status* für einen gültigen fünfstelligen SQL-Statuswert und *Klassencode* für die ersten beiden Ziffern dieses Statuswerts.
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 des DB2 Information Center

Die Dokumentation für andere Versionen der DB2-Produkte finden Sie in den jeweiligen Information Centers unter ibm.com.

Informationen zu diesem Vorgang

Für Themen aus DB2 Version 10.1 lautet die URL für das *DB2 Information Center* http://pic.dhe.ibm.com/infocenter/db2luw/v10r1.

Für Themen aus DB2 Version 9.8 lautet die URL der *DB2-Informationszentrale* http://pic.dhe.ibm.com/infocenter/db2luw/v9r8/.

Für Themen aus DB2 Version 9.7 lautet die URL der *DB2-Informationszentrale* http://pic.dhe.ibm.com/infocenter/db2luw/v9r7/.

Für Themen aus DB2 Version 9.5 lautet die URL der *DB2-Informationszentrale* http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Bedingungen

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit: Diese Bedingungen gelten zusätzlich zu den Nutzungsbedingungen für die IBM Website.

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 dieser 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 dieser Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Rechte: 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.

IBM Marken: IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- oder Servicenamen können Marken von IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite www.ibm.com/legal/copytrade.shtml.

Anhang E. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Die Informationen über Produkte anderer Hersteller als IBM basieren auf den zum Zeitpunkt der ersten Veröffentlichung dieses Dokuments verfügbaren Informationen und können geändert 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 IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

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 hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

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 unabhängig voneinander 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
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
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 bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen 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 Gewährleistung, 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 davon 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 kann Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes enthalten. Sie sollen nur die Funktionen des Lizenzprogramms 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 Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme 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. Die Beispielprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM haftet nicht für Schäden, die durch Verwendung der Beispielprogramme entstehen.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- oder Servicenamen können Marken von oder anderen Herstellern sein. IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Die folgenden Namen sind Marken oder eingetragene Marken anderer Unternehmen.

- Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken von Oracle und/oder ihren verbundenen Unternehmen.
- UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.
- Intel, das Intel-Logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder deren Tochtergesellschaften in den USA und anderen Ländern.
- Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

E

F

G

H

M

N

O

P

R

S

T

U

V

W

X

Z

SC12-4906-00

Spine information: