IBM DB2 10.1
for Linux, UNIX and Windows

Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz

Aktualisierung: Januar 2013

SC12-4691-01

IBM DB2 10.1
for Linux, UNIX and Windows

Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz

Aktualisierung: Januar 2013

SC12-4691-01

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in Anhang F, „Bemerkungen", auf Seite 313 gelesen werden.

Impressum

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 10.1 for Linux, UNIX, and Windows, Data Movement Utilities Guide and Reference,
IBM Form SC27-3869-01,
herausgegeben von International Business Machines Corporation, USA

Herausgegeben von:
TSC Germany
Kst. 2877
Januar 2013

Inhaltsverzeichnis

Inhalt des Handbuchs

Dieses Handbuch bietet Informationen zur Verwendung der folgenden DB2 for Linux, UNIX and Windows-Dienstprogramme für das Versetzen von Daten:

- EXPORT und IMPORT

  Die Dienstprogramme EXPORT und IMPORT versetzen Daten aus Tabellen oder Sichten in andere Datenbanken oder Tabellenkalkulationsprogramme, zwischen DB2-Datenbanken sowie zwischen DB2-Datenbanken und Hostdatenbanken mithilfe von DB2 Connect. Mit dem Dienstprogramm EXPORT können Daten von einer Datenbank in Betriebssystemdateien versetzt werden. Diese Dateien können Sie dann verwenden, um diese Daten in eine andere Datenbank zu importieren oder zu laden.

- LOAD

  Mit dem Dienstprogramm LOAD können Daten in Tabellen versetzt, vorhandene Indizes erweitert und Statistikdaten generiert werden. Das Dienstprogramm LOAD kann große Datenmengen schneller versetzen als das Dienstprogramm IMPORT. Daten, die mit dem Dienstprogramm EXPORT exportiert wurden, können mit dem Dienstprogramm LOAD geladen werden.

  Wird das Dienstprogramm LOAD in einer Umgebung mit partitionierten Datenbanken eingesetzt, können große Datenmengen verteilt und in unterschiedliche Datenbankpartitionen geladen werden.

Eine vollständige Auflistung der Optionen zum Versetzen von Daten finden Sie im Abschnitt „Optionen beim Versetzen von Daten” auf Seite 1.

Dienstprogramme und Referenzinformationen für das Versetzen von Daten

Optionen beim Versetzen von Daten

In DB2 for Linux, UNIX and Windows stehen verschiedene Optionen für das Versetzen von Daten zur Verfügung. Dieser Abschnitt enthält eine Übersicht über die verfügbaren Tools, Dienstprogramme, gespeicherten Prozeduren und Befehle für das Versetzen von Daten.

Verwenden Sie diese Tabellen als Richtlinie bei der Ermittlung der für Ihre Anforderungen am besten geeigneten Optionen für das Versetzen von Daten.

Tabelle 1. Dienstprogramm LOAD

Methode	Dienstprogramm LOAD
Zweck	Mit dem Dienstprogramm LOAD können große Datenmengen effizient in neu erstellte Tabellen oder in Tabellen, die bereits Daten enthalten, versetzt werden.
Plattformübergreifend kompatibel	Ja
Empfohlene Methode für die Verwendung	Dieses Dienstprogramm eignet sich am besten für Situationen, in denen die Leistung im Mittelpunkt steht. Dieses Dienstprogramm kann als Alternative zum Dienstprogramm IMPORT verwendet werden. Es arbeitet schneller als das Dienstprogramm IMPORT, da es formatierte Seiten direkt in die Datenbank schreibt, anstatt SQL-Anweisungen INSERT zu verwenden. Darüber hinaus bietet das Dienstprogramm LOAD die Möglichkeit, die Daten nicht zu protokollieren oder mit der Option `COPY` eine Kopie der geladenen Daten zu speichern. Ladeoperationen können die Ressourcen, wie beispielsweise CPUs und Hauptspeicherkapazitäten in SMP- und MPP-Umgebungen in vollem Umfang nutzen.
Referenzinformationen	Laden von Daten

Tabelle 2. Dienstprogramm INGEST

Methode	Dienstprogramm INGEST
Zweck	Leitet Datenströme aus Dateien und Pipes in DB2-Zieltabellen und sorgt gleichzeitig dafür, dass diese Tabellen verfügbar bleiben.
Plattformübergreifend kompatibel	Ja
Empfohlene Methode für die Verwendung	Mit diesem Dienstprogramm wird ein gutes Verhältnis zwischen Leistung und Verfügbarkeit erzielt. Ist jedoch die Verfügbarkeit wichtiger, sollte das Dienstprogramm INGEST anstelle des Dienstprogramms LOAD verwendet werden. Ähnlich dem Dienstprogramm IMPORT eignet sich INGEST in den Fällen, in denen es sich bei den Zieltabellen um aktualisierbare Sichten, Bereichsclustertabellen oder Kurznamem handelt; das Dienstprogramm INGEST bietet allerdings die bessere Leistung.
Referenzinformationen	Daten einpflegen

Tabelle 3. Dienstprogramm IMPORT

Methode	Dienstprogramm IMPORT
Zweck	Das Dienstprogramm IMPORT fügt Daten aus einer externen Datei mit einem unterstützten Dateiformat in eine Tabelle, eine Hierarchie, eine Sicht oder einen Kurznamen ein.
Plattformübergreifend kompatibel	Ja
Empfohlene Methode für die Verwendung	Das Dienstprogramm IMPORT kann in den folgenden Situationen eine geeignete Alternative zum Dienstprogramm LOAD darstellen: • Wenn es sich bei der Zieltabelle um eine Sicht handelt. • Wenn die Zieltabelle über Integritätsbedingungen verfügt und Sie nicht möchten, dass die Zieltabelle in den Status "Festlegen der Integrität anstehend" versetzt wird. • Wenn die Zieltabelle über Trigger verfügt und Sie möchten, dass diese ausgelöst werden.
Referenzinformationen	Importieren von Daten

Tabelle 4. Dienstprogramm EXPORT

Methode	Dienstprogramm EXPORT
Zweck	Das Dienstprogramm EXPORT exportiert Daten aus einer Datenbank in eines von mehreren externen Dateiformaten. Die Daten können dann zu einem späteren Zeitpunkt importiert oder geladen werden.
Plattformübergreifend kompatibel	Ja
Empfohlene Methode für die Verwendung	Dieses Dienstprogramm eignet sich am besten für Situationen, in denen Daten in einer externen Datei gespeichert werden sollen, um entweder weiter verarbeitet oder in eine andere Tabelle versetzt zu werden. High Performance Unload (HPU) ist eine Alternative hierzu, muss jedoch separat bezogen werden. Das Dienstprogramm EXPORT unterstützt XML-Spalten.
Referenzinformationen	Exportieren von Daten

Tabelle 5. db2move (Befehl)

Methode	Befehl **db2move**
Zweck	Wenn Sie das Dienstprogramm **db2move** mit der Option `COPY` verwenden, können Sie Schemaschablonen (mit oder ohne Daten) von einer Quellendatenbank in eine Zieldatenbank kopieren oder ein vollständiges Schema von einer Quellendatenbank in eine Zieldatenbank versetzen. Die Verwendung des Dienstprogramms **db2move** mit der Option `IMPORT` oder `EXPORT` vereinfacht das Versetzen einer großen Anzahl von Tabellen zwischen DB2-Datenbanken.
Plattformübergreifend kompatibel	Ja

Tabelle 5. db2move (Befehl) (Forts.)

Empfohlene Methode für die Verwendung	Wenn die Option `COPY` verwendet wird, müssen die Quellen- und die Zieltabelle unterschiedlich sein. Die Option `COPY` ist bei der Erstellung von Schemaschablonen nützlich. Verwenden Sie die Option `IMPORT` oder `EXPORT` für das Klonen von Datenbanken, wenn keine Unterstützung für plattformübergreifende Backup- und Restoreoperationen zur Verfügung steht. Die Optionen `IMPORT` oder `EXPORT` werden zusammen mit dem Befehl **`db2look`** verwendet.
Referenzinformationen	• „Kopieren eines Schemas" in *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen* • Neuerstellung von importierten Tabellen

Tabelle 6. RESTORE (Befehl)

Methode	Befehl **`RESTORE`** mit der Option `REDIRECT` und der Option `GENERATE SCRIPT`
Zweck	Mit dem Restoredienstprogramm kann eine gesamte Datenbank mithilfe eines Scripts aus einem vorhandenen Backup-Image von einem System auf ein anderes kopiert werden.
Plattformübergreifend kompatibel	Eingeschränkt. Siehe Referenzinformationen.
Empfohlene Methode für die Verwendung	Dieses Dienstprogramm eignet sich am besten für Situationen, in denen ein Backup-Image vorhanden ist.
Referenzinformationen	• „Ausführen eines umgeleiteten Restores mithilfe eines automatisch generierten Scripts" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* • „Backup- und Restoreoperationen zwischen unterschiedlichen Betriebssystemen und Hardwareplattformen" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz*

Tabelle 7. db2relocatedb (Befehl)

Methode	Befehl **`db2relocatedb`**
Zweck	Mit diesem Befehl können Sie eine Datenbank umbenennen oder eine Datenbank bzw. einen Teil einer Datenbank innerhalb desselben Systems oder auf ein anderes System verlagern.
Plattformübergreifend kompatibel	Nein
Empfohlene Methode für die Verwendung	• Dieses Dienstprogramm kann in Situationen verwendet werden, in denen Backup und Restore zu zeitaufwändig sind. • Dieses Dienstprogramm stellt eine Alternative zur Verwendung von Backup und Restore für das Versetzen von Datenbanken bzw. das Erstellen von Datenbankkopien dar. • Darüber hinaus bietet es eine zeitsparende Methode für das Klonen einer Datenbank für alternative Umgebungen, wie z. B. Testumgebungen. • Es kann für das Versetzen von Tabellenbereichscontainern in eine neue Gruppe von Speichereinheiten verwendet werden.
Referenzinformationen	„db2relocatedb - Relocate database command" (Befehl zum Verlagern einer Datenbank) im Handbuch *Command Reference*

Tabelle 8. Prozedur ADMIN_COPY_SCHEMA

Methode	Prozedur ADMIN_COPY_SCHEMA
Zweck	Diese Prozedur ermöglicht das Erstellen einer Kopie für alle Objekte in einem einzelnen Schema und das erneute Erstellen dieser Objekte in einem neuen Schema. Diese Kopieroperation kann innerhalb einer Datenbank mit oder ohne Daten ausgeführt werden.
Plattformübergreifend kompatibel	Ja
Empfohlene Methode für die Verwendung	Dieses Dienstprogramm ist für die Erstellung von Schemaschablonen geeignet. Es ist darüber hinaus nützlich, wenn Sie mit einem Schema experimentieren möchten (z. B. neue Indizes testen), ohne dass sich dies auf die Funktionsweise des Quellenschemas auswirkt. Die Hauptunterschiede zwischen der Prozedur ADMIN_COPY_SCHEMA und dem Dienstprogramm **db2move** sind: • Die Prozedur ADMIN_COPY_SCHEMA wird für eine einzelne Datenbank verwendet, das Dienstprogramm **db2move** dagegen datenbankübergreifend. • Das Dienstprogramm **db2move** schlägt beim Aufruf fehl, wenn es kein physisches Objekt, wie z. B. eine Tabelle oder einen Index, erstellen kann. Die Prozedur ADMIN_COPY_SCHEMA protokolliert die Fehler und setzt die Verarbeitung fort. • Die Prozedur ADMIN_COPY_SCHEMA verwendet die Option "vom Cursor laden", um Daten von einem Schema in ein anderes zu versetzen. Das Dienstprogramm **db2move** verwendet eine ferne Ladeoperation, ähnlich der Option "vom Cursor laden", die die Daten aus der Quellendatenbank extrahiert.
Referenzinformationen	„Kopieren eines Schemas” in *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen*

Tabelle 9. ADMIN_MOVE_TABLE, Prozedur

Methode	ADMIN_MOVE_TABLE, Prozedur
Zweck	Diese Prozedur ermöglicht das Versetzen der Daten aus einer Tabelle in ein neues Tabellenobjekt mit demselben Namen (aber möglicherweise mit anderen Speicherkenndaten), während die Daten online verfügbar bleiben und darauf zugegriffen werden kann.
Plattformübergreifend kompatibel	Ja

Tabelle 9. ADMIN_MOVE_TABLE, Prozedur (Forts.)

Empfohlene Methode für die Verwendung	Dieses Dienstprogramm automatisiert den Prozess zum Versetzen von Tabellendaten in ein neues Tabellenobjekt, während die Daten für den SELECT-, INSERT-, UPDATE- und DELETE-Zugriff online verfügbar bleiben können. Sie können auch ein Komprimierungswörterverzeichnis erstellen, während eine Tabelle versetzt wird. • Nehmen Sie nicht gleichzeitig mehrere Versetzungsoperationen in denselben Tabellenbereich vor. • Führen Sie die Prozedur aus, wenn nur wenige Aktivitäten an der Tabelle stattfinden. • Führen Sie die Versetzungsoperation in mehreren Schritten aus. Die Phasen INIT und COPY können jederzeit aufgerufen werden. Führen Sie die REPLAY-Phase mehrmals aus, damit die Zwischenspeichertabelle nicht zu groß wird, und geben Sie anschließend SWAP zu einer Zeit mit wenigen Aktivitäten an der Tabelle aus. • Ziehen Sie das Versetzen einer Tabelle im Offline-Modus in Betracht, wenn Sie mit Tabellen ohne eindeutige Indizes oder mit Tabellen ohne Index arbeiten.
Referenzinformationen	• „ADMIN_MOVE_TABLE procedure - Move an online table" (ADMIN_MOVE_TABLE (Prozedur) - Onlinetabelle versetzen) im Handbuch *Command Reference* • Versetzen von Tabellen im Online-Modus mithilfe der Prozedur ADMIN_MOVE_TABLE

Tabelle 10. Geteilte Spiegeldatenbank

Methode	Geteilte Spiegeldatenbank
Zweck	Erstellung einer Klon-, Bereitschafts- oder Backup-Datenbank
Plattformübergreifend kompatibel	Nein
Empfohlene Methode für die Verwendung	• Erstellen eines Bereitschaftssystems, um bei einem Ausfall des Primärsystems die Ausfallzeit zu reduzieren. • Verlagern von Backup-Operationen von einer Maschine mit laufender Produktion hin zu einer geteilten Datenbank. • Bereitstellen einer zeitsparenden Methode für das Klonen einer Datenbank für alternative Umgebungen, wie z. B. Testumgebungen.
Wichtige Hinweise	• Nur für DMS-Tabellenbereiche kann auf der geteilten Version der Datenbank ein Backup erstellt werden. • Dieses Dienstprogramm wird normalerweise zusammen mit einer Flashcopytechnologie verwendet, die von Speichersystemen bereitgestellt wird. • Eine Alternative besteht darin, nach dem Aussetzen der Datenbank eine Dateikopieroperation auszuführen, hierdurch wird jedoch die Menge des für die Datenbank erforderlichen Speicherplatzes verdoppelt.
Referenzinformationen	„Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz*

Dienstprogramm EXPORT

Dienstprogramm EXPORT - Übersicht

Das Dienstprogramm EXPORT extrahiert Daten mithilfe einer SQL-Anweisung SELECT oder einer XQuery-Anweisung und stellt die Informationen in eine Datei. Die Ausgabedatei können Sie zum Versetzen von Daten für eine zukünftige Import- oder Ladeoperation oder zum Bereitstellen von Daten für Analysezwecke verwenden.

Das Dienstprogramm EXPORT stellt ein ebenso einfaches wie flexibles Dienstprogramm zum Versetzen von Daten dar. Sie können das Dienstprogramm durch Ausgabe des Befehls **EXPORT** über den Befehlszeilenprozessor, durch Aufruf der gespeicherten Prozedur ADMIN_CMD oder durch Aufruf der API db2Export über eine Benutzeranwendung aktivieren.

Für eine einfache Exportoperation sind folgende Angaben erforderlich:

- Pfad und Name des Betriebssystems, in dem die zu exportierenden Daten gespeichert werden sollen
- Das Format der Daten in der Eingabedatei
  Das Dienstprogramm EXPORT unterstützt die Datenformate IXF und DEL für die Ausgabedateien.
- Eine nähere Angabe der zu exportierenden Daten
  Für die meisten Exportoperationen müssen Sie eine Anweisung SELECT bereitstellen, die die für den Export abzurufenden Daten angibt. Für den Export typisierter Tabellen ist eine explizite Ausgabe der Anweisung SELECT nicht erforderlich. Sie müssen in diesem Fall lediglich die Traversierfolge für untergeordnete Tabellen innerhalb der Hierarchie angeben.

Zum Versetzen von Daten im Format IXF können Sie das Dienstprogramm EXPORT mit DB2 Connect verwenden.

Weitere Optionen

Sie können Exportoperationen mit einer Reihe von Parametern anpassen. Änderungswerte für den Dateityp eröffnen viele Möglichkeiten und erlauben z. B. das Ändern des Datenformats und der Codepage sowie von Datums- und Zeitmarken und das Schreiben bestimmter Datentypen in separate Dateien. Mit den Parametern für **METHOD** können Sie andere Spaltennamen für die zu exportierenden Daten angeben.

Sie können Daten aus Tabellen exportieren, die Spalten mit XML-Datentyp enthalten. Mit den Parametern **XMLFILE**, **XML TO** und **XMLSAVESCHEMA** können Sie näher bestimmen, wie die exportierten Dokumente anschließend gespeichert werden sollen.

Die Leistung des Dienstprogramms EXPORT kann auf verschiedene Arten optimiert werden. Da es sich bei dem Dienstprogramm EXPORT um eine integrierte SQL-Anwendung handelt, die SQL-Abrufe intern durchführt, profitiert das Dienstprogramm EXPORT auch von Optimierungen, die bei den SQL-Operationen vorgenommen werden. Sinnvoll können z. B. große Pufferpools, Indexierungen und Sortierspeicher sein. Es empfiehlt sich darüber hinaus, das Risiko von Einheitenkonkurrenzsituationen bei den Ausgabedateien zu vermeiden, indem sie getrennt von den Containern und Protokolleinheiten platziert werden.

Nachrichtendatei

Das Dienstprogramm EXPORT schreibt Fehler- und Informationsnachrich-

ten sowie Warnungen in ASCII-Standardtextnachrichtendateien. Abgesehen vom Befehlszeilenprozessor müssen Sie bei allen Schnittstellen den Namen dieser Dateien zuvor mit dem Parameter **MESSAGES** angeben. Wenn Sie den Befehlszeilenprozessor verwenden und keine Nachrichtendatei angeben, schreibt das Dienstprogramm EXPORT die Nachrichten in die Standardausgabe.

Ab IBM® Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Exportieren von Daten. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Für das Dienstprogramm EXPORT erforderliche Zugriffsrechte und Berechtigungen

Zugriffsrechte ermöglichen Ihnen den Zugriff auf Datenbankressourcen sowie das Erstellen, Aktualisieren und Löschen dieser Ressourcen. Berechtigungsstufen bieten die Möglichkeit, Berechtigungen zu übergeordneten Pflege- und Dienstprogrammoperationen des Datenbankmanagers zuzuordnen.

Zugriffsrechte und Berechtigungen dienen gleichermaßen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte. Sie können nur auf solche Objekte zugreifen, für die Sie zugriffsberechtigt sind, d. h., für die Sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Sie benötigen die Berechtigung DATAACCESS bzw. das Zugriffsrecht CONTROL oder SELECT für jede Tabelle und Sicht, die an dem Exportvorgang beteiligt ist.

Beim Export LBAC-geschützter Daten (Label-Based Access Control, kennsatzbasierte Zugriffssteuerung) muss die Berechtigungs-ID der Sitzung zum Lesen der zu exportierenden Zeilen bzw. Spalten berechtigt sein. Geschützte Zeilen, die von der Berechtigungs-ID der Sitzung nicht gelesen werden dürfen, werden nicht exportiert. Schließt die Anweisung SELECT geschützte Spalten ein, die von der Berechtigungs-ID der Sitzung nicht gelesen werden dürfen, schlägt der Export fehl, und es wird ein Fehler (SQLSTATE 42512) zurückgegeben.

Exportieren von Daten

Mit dem Dienstprogramm EXPORT können Sie Daten aus einer Datenbank in eine Datei exportieren. Die Datei kann eines von mehreren externen Dateiformaten aufweisen. Sie können die zu exportierenden Daten durch eine SQL-Anweisung SELECT oder über hierarchische Informationen für typisierte Tabellen angeben.

Vorbereitende Schritte

Zum Exportieren von Daten aus einer Datenbank müssen Sie über die Berechtigung DATAACCESS, das Zugriffsrecht CONTROL oder das Zugriffsrecht SELECT für jede einzelne am Export beteiligte Tabelle oder Sicht verfügen.

Bevor Sie das Dienstprogramm EXPORT ausführen, müssen Sie mit der Datenbank verbunden sein, aus der die Daten exportiert werden sollen, oder in der Lage sein, implizit eine Verbindung zu dieser Datenbank herzustellen. Wenn das implizite Herstellen von Verbindungen aktiviert ist, wird eine Verbindung zur Standarddatenbank hergestellt. Beim Zugriff des Dienstprogramms auf Linux-, UNIX- oder Windows-Datenbankserver von Linux-, UNIX- oder Windows-Clients aus muss es

sich um eine Direktverbindung über die Steuerkomponente handeln. Eine Verbindung über ein DB2 Connect-Gateway oder eine Loopback-Umgebung ist nicht zulässig.

Da das Dienstprogramm die Anweisung COMMIT absetzt, sollten Sie vor der Ausführung des Dienstprogramms EXPORT alle Transaktionen beenden und alle Sperren freigeben, indem Sie eine Anweisung COMMIT oder eine Anweisung ROLLBACK absetzen. Es ist nicht erforderlich, dass andere Benutzeranwendungen, die auf die Tabelle zugreifen und hierbei eigene Verbindungen verwenden, diese Verbindungen trennen.

Tabellen, die Spalten mit strukturiertem Typ enthalten, können nicht exportiert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um das Dienstprogramm EXPORT auszuführen:

- Geben Sie den Befehl **EXPORT** im Befehlszeilenprozessor ein.
- Rufen Sie die Anwendungsprogrammierschnittstelle (API) db2Export auf.
- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **EXPORT**.

Beispiel

Einfache Exportoperationen erfordern lediglich die Angabe einer Zieldatei, eines Dateiformats und einer Quellendatei für die Anweisung SELECT.

Beispiel:

```
db2 export to dateiname of ixf select * from tabelle
```

Dabei ist *dateiname* der Name der Ausgabedatei, die erstellt und exportiert werden soll, `ixf` das Dateiformat und *tabelle* der Name der Tabelle mit den zu kopierenden Daten.

Es kann jedoch auch sinnvoll sein, eine Nachrichtendatei, in die Warnungen und Fehlernachrichten geschrieben werden können, anzugeben. Fügen Sie dazu den Parameter **MESSAGES** und den Namen einer Nachrichtendatei (z. B. `msg.txt`) hinzu. Beispiel:

```
db2 export to dateiname of ixf messages msgs.txt select * from tabelle
```

Exportieren von XML-Daten

Beim Exportieren von XML-Daten werden die dabei generierten XDM-Instanzen (XDM = XQuery Data Model) in eine oder mehrere Dateien geschrieben, die separat von der Hauptdatendatei abgelegt werden, die die exportierten relationalen Daten enthält. Dies gilt auch dann, wenn weder die Option XMLFILE noch die Option XML TO angegeben ist.

Standardmäßig werden alle exportierten QDM-Instanzen mit derselben XML-Datei verknüpft. Mit dem Dateitypmodifikator `XMLINSEPFILES` können Sie angeben, dass alle QDM-Instanzen in eine separate Datei geschrieben werden sollen.

Die XML-Daten werden jedoch in der Hauptdatendatei durch eine XML-Datenkennung (XDS = XML Data Specifier) dargestellt. Die XML-Datenkennung besteht aus einer Zeichenfolge, die durch einen XML-Tag mit dem Namen "XDS" dargestellt wird, und verfügt über Attribute, die Informationen zu den eigentlichen XML-Daten in der Spalte enthalten. Hierzu gehören z. B. Angaben zum Namen der Datei,

in der die eigentlichen XML-Daten enthalten sind, sowie Angaben zur relativen Position und Länge der XML-Daten in dieser Datei.

Die Zielpfade und Basisdateinamen der exportierten XML-Dateien können mit den Optionen XML TO und XMLFILE angegeben werden. Bei Angabe der Option XML TO oder XMLFILE lautet das Namensformat der exportierten und im Attribut FIL der XDS gespeicherten XML-Dateien `xmlfilespec.xxx.xml`, wobei `xmlfilespec` für den für die Option XMLFILE angegebenen Wert und `xxx` für eine vom Dienstprogramm EXPORT generierte Folgenummer für XML-Dateien steht. Ansonsten lautet das Namensformat der exportierten XML-Dateien `exportfilename.xxx.xml`, wobei `exportfilename` für den Namen der im Befehl EXPORT angegebenen exportierten Ausgabedatei und `xxx` für eine vom Dienstprogramm EXPORT generierte Folgenummer für XML-Dateien steht.

Standardmäßig werden exportierte XML-Dateien in den Pfad geschrieben, in dem sich auch die exportierte Datendatei befindet. Der Standardwert für den Basisdateinamen stimmt bei exportierten XML-Dateien mit dem Namen der exportierten Datendatei überein, wobei eine aus drei Ziffern bestehende Folgenummer und die Erweiterung `.xml` angefügt werden.

Beispiele

Gehen Sie in den folgenden Beispielen von einer Tabelle USER.T1 aus, die vier Spalten und zwei Zeilen enthält:

```
C1 INTEGER
   C2 XML
   C3 VARCHAR(10)
   C4 XML
```

Tabelle 11. USER.T1

C1	C2	C3	C4
2	<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</ to><from> Me</ from><heading>note1</heading> <body>Hello World!</body></ note>	'char1'	<?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00"><to>Him</ to><from> Her</ from><heading>note2</ heading><body>Hello World!</ body></note>
4	NULL	'char2'	?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00">to>Us</ to><from> Them</ from><heading>note3</heading> <body>Hello World!</body></note>

Beispiel 1

Mit dem folgenden Befehl können Sie den Inhalt von USER.T1 im ASCII-Format mit Begrenzern (DEL) in die Datei "/mypath/t1export.del" exportieren. Da die Optionen XML TO und XMLFILE nicht angegeben wurden, werden die in den Spalten C2 und C4 enthaltenen XML-Dokumente in denselben Pfad geschrieben wie die exportierte Hauptdatei "/mypath". Der Basisdateiname für diese Dateien lautet "t1export.del.xml". Die Option XMLSAVESCHEMA gibt an, dass die XML-Schemainformationen während der Exportprozedur gespeichert werden.

```
EXPORT TO /mypath/t1export.del OF DEL XMLSAVESCHEMA SELECT * FROM USER.T1
```

Die exportierte Datei "/mypath/t1export.del" enthält Folgendes:

```
   2,"<XDS FIL='t1export.del.001.xml' OFF='0' LEN='144' />","char1",
   "<XDS FIL='t1export.del.001.xml' OFF='144' LEN='145' />"
   4,,"char2","<XDS FIL='t1export.del.001.xml' OFF='289'
LEN='145' SCH='S1.SCHEMA_A' />"
```

Die exportierte XML-Datei "/mypath/t1export.del.001.xml" enthält Folgendes:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
         <from>Me</from><heading>note1</heading><body>Hello World!</body>
         </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00"><to>Him
         </to><from>Her</from><heading>note2</heading><body>Hello World!
         </body></note><?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00">
   <to>Us</to><from>Them</from>heading>note3</heading><body>
   Hello World!</body></note>
```

Beispiel 2

Mit dem folgenden Befehl können Sie den Inhalt von USER.T1 im ASCII-Format mit Begrenzern (DEL) in die Datei "t1export.del" exportieren. XML-Dokumente, die in den Spalten C2 und C4 enthalten sind, werden in den Pfad "/home/user/xmlpath" geschrieben. Den XML-Dateien wird der Basisdateiname "xmldocs" zugeordnet, und es werden mehrere exportierte XML-Dokumente in dieselbe XML-Datei geschrieben. Die Option XMLSAVESCHEMA gibt an, dass die XML-Schemainformationen während der Exportprozedur gespeichert werden.

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs XMLSAVESCHEMA SELECT * FROM USER.T1
```

Die exportierte DEL-Datei "/home/user/t1export.del" enthält Folgendes:

```
   2,"<XDS FIL='xmldocs.001.xml' OFF='0' LEN='144' />","char1",
   "<XDS FIL='xmldocs.001.xml' OFF='144' LEN='145' />"
   4,,"char2","<XDS FIL='xmldocs.001.xml' OFF='289'
LEN='145' SCH='S1.SCHEMA_A' />"
```

Die exportierte XML-Datei "/home/user/xmlpath/xmldocs.001.xml" enthält Folgendes:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
         <from>Me</from><heading>note1</heading><body>Hello World!</body>
         </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00">
         <to>Him</to><from>Her</from><heading>note2</heading><body>
         Hello World!</body></note><?xml version="1.0" encoding="UTF-8" ?>
         <note time="14:00:00"><to>Us</to><from>Them</from><heading>
   note3</heading><body>Hello World!</body></note>
```

Beispiel 3

Der folgende Befehl ist mit dem Befehl aus Beispiel 2 vergleichbar. Eine Ausnahme bildet hierbei allerdings die Tatsache, dass alle exportierten XML-Dokumente in eine separate XML-Datei geschrieben werden.

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs MODIFIED BY XMLINSEPFILES XMLSAVESCHEMA
SELECT * FROM USER.T1
```

Die exportierte Datei "/mypath/t1export.del" enthält Folgendes:

```
2,"<XDS FIL='xmldocs.001.xml' />","char1","XDS FIL='xmldocs.002.xml' />"
   4,,"char2","<XDS FIL='xmldocs.004.xml' SCH='S1.SCHEMA_A' />"
```

Die exportierte XML-Datei "/home/user/xmlpath/xmldocs.001.xml" enthält Folgendes:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
         <from>Me</from><heading>note1</heading><body>Hello World!</body>
         </note>
```

Die exportierte XML-Datei "/home/user/xmlpath/xmldocs.002.xml" enthält Folgendes:

```
?xml version="1.0" encoding="UTF-8" ?>note time="13:00:00">to>Him/to>
   from>Her/from>heading>note2/heading>body>Hello World!/body>
   /note>
```

Die exportierte XML-Datei "/home/user/xmlpath/xmldocs.004.xml" enthält Folgendes:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00"><to>Us</to>
         <from>Them</from><heading>note3</heading><body>Hello World!</body>
         </note>
```

Beispiel 4

Mit dem folgenden Befehl können Sie die Ergebnisse einer XQuery-Operation in eine XML-Datei schreiben.

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs MODIFIED BY XMLNODECLARATION select
xmlquery( '$m/note/from/text()' passing by ref c4 as "m" returning sequence)
      from USER.T1
```

Die exportierte DEL-Datei "/mypath/t1export.del" enthält Folgendes:

```
   "<XDS FIL='xmldocs.001.xml' OFF='0' LEN='3' />"
"<XDS FIL='xmldocs.001.xml' OFF='3' LEN='4' />"
```

Die exportierte XML-Datei "/home/user/xmlpath/xmldocs.001.xml" enthält Folgendes:

```
HerThem
```

Anmerkung: Das Ergebnis dieser speziellen XQuery-Operation generiert keine korrekt formatierten XML-Dokumente. Aus diesem Grund konnte die in diesem Beispiel exportierte Datei nicht direkt in eine XML-Spalte importiert werden.

Exportsitzungen - Beispiele für CLP

Beispiel 1

Das folgende Beispiel zeigt, wie Daten aus der Tabelle STAFF der Beispieldatenbank SAMPLE (mit der der Benutzer verbunden sein muss) im IXF-Format in die Datei `myfile.ixf` exportiert werden. Wenn die Datenbankverbindung nicht über DB2 Connect hergestellt wurde, werden die Indexdefinitionen (falls vorhanden) in der Ausgabedatei gespeichert. Andernfalls werden nur die Daten gespeichert:

```
db2 export to myfile.ixf of ixf messages msgs.txt select * from staff
```

Beispiel 2

Das folgende Beispiel zeigt, wie Daten über Mitarbeiter in Abteilung 20 aus der Tabelle STAFF der Beispieldatenbank SAMPLE (mit der der Benutzer verbunden sein muss) im IXF-Format in die Datei `awards.ixf` exportiert werden.

```
db2 export to awards.ixf of ixf messages msgs.txt select * from staff
where dept = 20
```

Beispiel 3

Das folgende Beispiel zeigt, wie große Objekte in eine DEL-Datei exportiert werden:

```
db2 export to myfile.del of del lobs to mylobs/
      lobfile lobs1, lobs2 modified by lobsinfile
select * from emp_photo
```

Beispiel 4
Das folgende Beispiel zeigt, wie große Objekte (LOBs) in eine DEL-Datei exportiert werden, wobei ein zweites Verzeichnis für Dateien angegeben wird, die im ersten Verzeichnis möglicherweise keinen Platz haben:

```
db2 export to myfile.del of del
      lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
select * from emp_photo
```

Beispiel 5
Das folgende Beispiel zeigt den Export von Daten in eine DEL-Datei, wobei ein einfaches Anführungszeichen als Zeichenfolgebegrenzer, ein Semikolon als Spaltenbegrenzer und ein Komma als Dezimalzeichen verwendet wird. Die gleiche Konvention muss beim Rückimportieren der Daten in die Datenbank verwendet werden:

```
db2 export to myfile.del of del
      modified by chardel'' coldel; decpt,
select * from staff
```

Aspekte des Exports von LBAC-geschützten Daten

Wenn Sie Daten exportieren, die durch eine kennsatzbasierte Zugriffssteuerung (LBAC, Label-Based Access Control) geschützt sind, werden nur die Daten exportiert, die Sie gemäß Ihren LBAC-Berechtigungsnachweisen lesen dürfen.

Wenn Ihre LBAC-Berechtigungsnachweise Ihnen das Lesen einer Zeile nicht ermöglichen, wird die betreffende Zeile nicht exportiert, es wird jedoch kein entsprechender Fehler zurückgegeben. Wenn Ihre LBAC-Berechtigungsnachweise Ihnen das Lesen einer Spalte nicht ermöglichen, schlägt die Ausführung des Dienstprogramms EXPORT fehl, und es wird eine Fehlernachricht SQLSTATE 42512 zurückgegeben.

Werte aus Spalten mit dem Datentyp DB2SECURITYLABEL werden als Rohdaten, eingeschlossen in Zeichenbegrenzer, exportiert. Enthalten die Originaldaten einen Zeichenbegrenzer, wird dieser Begrenzer verdoppelt. An den Byte, die zusammen den exportierten Wert ergeben, werden ansonsten keine Änderungen vorgenommen. Dies bedeutet, dass eine Datendatei, die Daten des Typs DB2SECURITYLABEL enthält, Zeichen für Zeilenumbruch, Formularvorschub und andere nicht druckbare ASCII-Zeichen enthalten kann.

Wenn Sie die Werte des Datentyps DB2SECURITYLABEL in eine für Benutzer lesbare Form exportieren möchten, können Sie dazu die Skalarfunktion SECLABEL_TO_CHAR in der Anweisung SELECT verwenden, die die Werte in das Zeichenfolgeformat für Sicherheitskennsätze konvertiert.

Beispiele

In den folgenden Beispielen erfolgt die Ausgabe im Format DEL und wird in die Datei `myfile.del` geschrieben. Die Daten werden aus einer Tabelle mit der Bezeichnung REPS exportiert, die mit der folgenden Anweisung erstellt wurde:

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

Bei dem folgenden Beispiel werden die Werte der Spalte "row_label" im Standardformat exportiert:

```
db2 export to myfile.del of del select * from reps
```

Die Datendatei ist in den meisten Texteditoren nur schlecht zu entziffern, da die Werte für die Spalte "row_label" in der Regel verschiedene ASCII-Steuerzeichen enthalten.

Bei dem folgenden Beispiel werden die Werte für die Spalte "row_label" im Zeichenfolgeformat für Sicherheitskennsätze exportiert:

```
db2 export to myfile.del of del select SECLABEL_TO_CHAR(row_label,
'DATA_ACCESS_POLICY'), id, name from reps
```

Der folgende Auszug stammt aus der Datendatei, die im vorangehenden Beispiel erstellt wurde. Beachten Sie hierbei, dass das Format des Sicherheitskennsatzes lesbar ist:

```
...
"Secret:():Epsilon 37", 2005, "Susan Liu"
"Secret:():(Epsilon 37,Megaphone,Cloverleaf)", 2006, "Johnny Cogent"
"Secret:():(Megaphone,Cloverleaf)", 2007, "Ron Imron"
...
```

Aspekte des Tabellenexports

Standardexportoperationen beinhalten die Ausgabe ausgewählter Daten, die in vorhandene Tabellen eingefügt oder geladen werden. Es ist jedoch auch möglich, vollständige Tabellen zu exportieren, um sie anschließend mit dem Dienstprogramm IMPORT erneut zu erstellen.

Zum Exportieren einer Tabelle müssen Sie das PC/IXF-Dateiformat angeben. Sie können die gespeicherte Tabelle anschließend einschließlich zugehöriger Indizes erneut erstellen, indem Sie das Dienstprogramm IMPORT im Modus `CREATE` verwenden. Bestimmte Informationen werden jedoch unter folgenden Umständen in der exportierten IXF-Datei nicht gespeichert:

- Die Indexspaltennamen enthalten den Hexadezimalwert 0x2B oder 0x2D.
- Die Tabelle enthält XML-Spalten.
- Bei der Tabelle handelt es sich um eine MDC-Tabelle (Multi-Dimensionally Clustered).
- Die Tabelle enthält einen Tabellenpartitionierungsschlüssel.
- Der Indexname ist auf Grund einer Codepagekonvertierung länger als 128 Byte.
- Die Tabelle ist geschützt.
- Der Befehl **`EXPORT`** enthält andere Aktionszeichenfolgen als SELECT * FROM *tabellenname*
- Sie geben den Parameter **`METHOD N`** für das Dienstprogramm EXPORT an.

Eine Liste der Tabellenattribute, die verloren gehen, enthält der Abschnitt "Aspekte des Tabellenimports". Werden Informationen nicht gespeichert, wird die Warnung SQL27984W beim erneuten Erstellen der Tabelle zurückgegeben.

Anmerkung: Der Importmodus `CREATE` ist veraltet. Verwenden Sie das Dienstprogramm **`db2look`**, um Ihre Tabellen zu erfassen und erneut zu erstellen.

Indexinformationen

Enthalten die im Index angegebenen Spaltennamen das Zeichen - oder + , werden die Indexinformationen nicht erfasst und die Warnung SQL27984W wird zurückgegeben. Das Dienstprogramm EXPORT beendet die Verarbei-

tung. Die exportierten Daten sind nicht betroffen, die Indexinformationen werden jedoch nicht in der IXF-Datei gespeichert. Sie müssen die Indizes deshalb mit Hilfe des Dienstprogramms **db2look** separat erstellen.

Einschränkungen durch den Speicherplatz
Die Exportoperation schlägt fehl, wenn der verfügbare Speicherbereich auf dem Dateisystem, auf dem die exportierte Datei erstellt werden soll, für die zu exportierenden Daten nicht ausreicht. In diesem Fall müssen Sie die Menge der ausgewählten Daten durch Angabe von Bedingungen in der Klausel WHERE begrenzen, damit die exportierte Datei auf dem Zieldateisystem Platz findet. Sie können das Dienstprogramm EXPORT mehrmals ausführen, um alle Daten zu exportieren.

Tabellen mit anderen Dateiformaten
Wenn Sie die Exportoperation nicht mit dem IXF-Dateiformat vornehmen, enthalten die Ausgabedateien zwar keine Beschreibungen der Zieltabelle, sie enthalten jedoch die Satzdaten. Um eine Tabelle mit den zugehörigen Daten erneut zu erstellen, erstellen Sie die Zieltabelle und verwenden anschließend das Dienstprogramm LOAD oder IMPORT, um die Tabelle mit Daten zu füllen. Mit dem Dienstprogramm **db2look** können Sie die ursprünglichen Tabellendefinitionen erfassen und die entsprechende Datendefinitionssprache (DDL) generieren.

Aspekte des Exports typisierter Tabellen

Mit dem DB2 Dienstprogramm EXPORT können Sie Daten aus typisierten Tabellen versetzen, um sie zu einem späteren Zeitpunkt zu importieren. Das Versetzen von Daten aus einer hierarchischen Struktur mit typisierten Tabellen in eine andere erfolgt beim Export entsprechend einer bestimmten Reihenfolge und durch Erstellung einer temporären Flachdatei.

Bei der Arbeit mit typisierten Tabellen steuert das Dienstprogramm EXPORT die Einfügungen in die Ausgabedatei. Sie müssen lediglich den Namen der Zieltabelle und bei Bedarf eine Klausel WHERE angeben. Subselect-Anweisungen können nur dadurch ausgedrückt werden, dass der Name der Zieltabelle und die Klausel WHERE angegeben wird. Die Angabe einer Anweisung "Fullselect" oder SELECT ist beim Export von Hierarchien nicht möglich.

Beibehaltung von Hierarchien mithilfe der Traversierfolge

Typisierte Tabellen können sich in einer Hierarchie befinden. Beim Versetzen von Daten zwischen verschiedenen Hierarchien gibt es verschiedene Möglichkeiten:

- Versetzen von Daten aus einer Hierarchie in eine identische Hierarchie
- Versetzen von Daten aus einer Hierarchie in einen Unterabschnitt einer größeren Hierarchie
- Versetzen von Daten aus einem Unterabschnitt einer großen Hierarchie in eine separate Hierarchie

Die Kennzeichnung von Typen in einer Hierarchie ist von der jeweiligen Datenbank abhängig. Dies bedeutet, dass derselbe Typ in verschiedenen Datenbanken unterschiedliche Kennungen hat. Darum muss beim Versetzen von Daten zwischen diesen Datenbanken ein Typenabgleich erfolgen, um sicherzustellen, dass die Daten korrekt versetzt werden.

Das für typisierte Tabellen verwendete Zuordnungsprinzip basiert auf der *Traversierfolge*. Die Traversierfolge gibt vor, dass der Durchgang durch alle übergeordneten und untergeordneten Tabellen in der Hierarchie von oben nach unten und von links nach rechts erfolgt. Bevor die einzelnen typisier-

ten Zeilen bei einer Exportoperation herausgeschrieben werden, wird eine Kennung in einen Indexwert umgesetzt. Dieser Indexwert kann eine beliebige Zahl von 1 bis zur Anzahl der relevanten Typen in der Hierarchie sein. Indexwerte werden gebildet, indem beim Durchgang durch die Hierarchie in einer bestimmten Reihenfolge (der Traversierfolge) alle Typen durchnummeriert werden. Abbildung 1 zeigt eine Hierarchie mit vier gültigen Traversierfolgen:

- Person, Employee, Manager, Architect, Student
- Person, Student, Employee, Manager, Architect
- Person, Employee, Architect, Manager, Student
- Person, Student, Employee, Architect, Manager

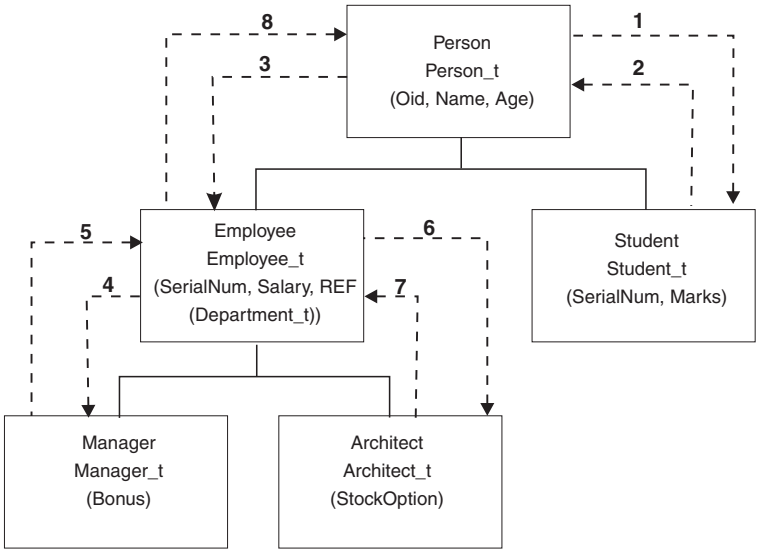


Abbildung 1. Hierarchiebeispiel

Die Traversierfolge ist wichtig beim Versetzen von Daten zwischen Tabellenhierarchien, weil dabei im Verhältnis zu anderen Daten ermittelt wird, wohin die Daten versetzt werden. Es gibt zwei Typen von Traversierfolgen: *Standard*traversierfolgen und *benutzerdefinierte* Traversierfolgen.

Standardtraversierfolge

Bei der Standardtraversierfolge verweisen alle relevanten Typen ab einem bestimmten Startpunkt in der Hierarchie auf alle erreichbaren Typen der Hierarchie. Die Standardreihenfolge schließt alle Tabellen in der Hierarchie ein, und jede Tabelle wird nach dem Schema sortiert, das im erweiterten Sortiervergleichselement verwendet wird. Die Standardtraversierfolge in Abbildung 1, angegeben durch die gepunktete Linie, ist z. B. die Folge "Person, Student, Employee, Manager, Architect".

Die Standardtraversierfolge verhält sich bei verschiedenen Dateiformaten unterschiedlich. Beim Export von Daten in das PC/IXF-Dateiformat wird ein Datensatz mit allen relevanten Typen, ihren Definitionen und den relevanten Tabellen erzeugt. Das Dienstprogramm EXPORT nimmt außerdem die Zuordnung von Indexwerten für die einzelnen Tabellen vor. Beim Arbeiten mit dem PC/IXF-Dateiformat ist es ratsam, die Standardtraversierfolge zu verwenden.

Beim Arbeiten mit dem ASC- oder DEL-Dateiformat kann die Erstellungsreihenfolge der typisierten Zeilen und typisierten Tabellen unterschiedlich sein, auch wenn die Quellen- und Zielhierarchien möglicherweise dieselbe Struktur aufweisen. Dies führt zu Zeitdifferenzen, die die Standardtraversierfolge beim Durchgang durch die Hierarchien feststellt. Die Erstellungszeit jedes Typs bestimmt, in welcher Reihenfolge der Durchgang durch die Quellen- und die Zielhierarchie erfolgt, wenn die Standardtraversierfolge verwendet wird. Sie müssen sicherstellen, dass sowohl die Reihenfolge beim Erstellen der einzelnen Typen in Quellen- und Zielhierarchie als auch die hierarchische Struktur von Quelle und Ziel identisch sind. Wählen Sie eine benutzerdefinierte Standardtraversierfolge aus, wenn diese Bedingungen nicht erfüllt werden können.

Benutzerdefinierte Traversierfolge

Bei der benutzerdefinierten Traversierfolge werden die zu verwendenden relevanten Typen vom Benutzer in einer Traversierfolgeliste definiert. Diese Reihenfolge gibt vor, wie die Hierarchie durchlaufen werden soll und welche untergeordneten Tabellen exportiert werden sollen, während bei der Standardtraversierfolge alle Tabellen der Hierarchie exportiert werden.

Beim Definieren der Traversierfolge können Sie zwar den Startpunkt und den Pfad durch die Hierarchie festlegen, es ist jedoch zu beachten, dass die untergeordneten Tabellen in einer *vordefinierten Reihenfolge* durchlaufen werden müssen. Jede Verzweigung der Hierarchie muss zunächst komplett durchlaufen werden, bevor mit einer neuen Verzweigung begonnen werden kann. Das Dienstprogramm EXPORT erkennt die Nichteinhaltung dieser Bedingung innerhalb der angegebenen Traversierfolge. Eine Methode, das Einhalten dieser Bedingung sicherzustellen, ist der Durchgang von der obersten Stufe der Hierarchie (oder der Stammtabelle) durch die untergeordneten Stufen (untergeordnete Tabellen) bis zur letzten untergeordneten Tabelle; anschließend zurück zu ihrer übergeordneten Tabelle, weiter durch die nächste "unmittelbar rechts angrenzende" untergeordnete Tabelle; danach zurück zur nächsthöheren übergeordneten Tabelle, weiter durch deren untergeordnete Tabellen usw.

Wenn Sie die Traversierfolge für die Hierarchien steuern möchten, müssen Sie sicherstellen, dass für die Dienstprogramme EXPORT und IMPORT die gleiche Traversierfolge verwendet wird.

Beispiel 1

Die folgenden Beispiele sind an der in Abbildung 1 dargestellten hierarchischen Struktur ausgerichtet. Geben Sie zum Exportieren der gesamten Hierarchie folgende Befehle ein:

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO entire_hierarchy.ixf OF IXF HIERARCHY STARTING Person
```

Beachten Sie dabei, dass die Definition des Parameters **HIERARCHY STARTING** mit `Person` angibt, dass die Standardtraversierfolge mit der Tabelle PERSON beginnen soll.

Beispiel 2

Wenn Sie zwar die gesamte Hierarchie exportieren möchten, jedoch nur die Daten für Personen über 20, müssen Sie dazu die folgenden Befehle eingeben:

```
DB2 CONNECT TO Source_db
   DB2 EXPORT TO entire_hierarchy.del OF DEL HIERARCHY (Person,
      Employee, Manager, Architect, Student) WHERE Age>=20
```

Beachten Sie dabei, dass die Definition des Parameters **HIERARCHY** mit `Person, Employee, Manager, Architect, Student` auf eine benutzerdefinierte Traversierfolge verweist.

Aspekte des Identitätsspaltenexports

Mit dem Dienstprogramm EXPORT können Sie Daten aus Tabellen exportieren, die Identitätsspalten enthalten. Die Identitätsspalten schränken jedoch die Möglichkeiten bei der Auswahl der Ausgabedateiformate ein.

Hat die Anweisung SELECT, die für die Exportoperation angegeben wird, das Format SELECT * FROM *tabellenname* und wird die Option `METHOD` nicht verwendet, wird das Exportieren der Identitätsspalteneigenschaften in IXF-Dateien unterstützt. Anschließend können die Optionen `REPLACE_CREATE` und `CREATE` des Befehls **IMPORT** verwendet werden, um die Tabelle einschließlich ihrer Identitätsspalteneigenschaften erneut zu erstellen. Wird die exportierte IXF-Datei aus einer Tabelle mit einer als GENERATED ALWAYS definierten Identitätsspalte erstellt, kann die Datendatei nur dann erfolgreich importiert werden, wenn der Änderungswert `identityignore` für den Dateityp angegeben wird. Andernfalls werden alle Zeilen zurückgewiesen, und es wird die Fehlernachricht SQL3550W ausgegeben.

Anmerkung: Die Optionen `CREATE` und `REPLACE_CREATE` des Befehls IMPORT werden nicht weiter unterstützt und in zukünftigen Releases möglicherweise entfernt.

Aspekte des LOB-Exports

Beim Exportieren von Tabellen mit LOB-Spalten (Large Objects) besteht die Standardaktion darin, die ersten 32 KB Daten pro LOB-Wert auszuwählen und diese in dieselbe Datei wie die restlichen Spaltendaten zu stellen. Beim Export von LOB-Werten über 32 KB können Sie vermeiden, dass die Daten abgeschnitten werden, indem Sie dafür sorgen, dass die LOB-Daten in eine separate Datei geschrieben werden.

Mit dem Änderungswert `lobsinfile` für den Dateityp können Sie angeben, dass LOB-Daten in eine separate Datei geschrieben werden. Dieser Änderungswert bewirkt beim Dienstprogramm EXPORT, dass die LOB-Daten in die in der Klausel `LOBS TO` angegebenen Verzeichnisse gestellt werden. Die Verwendung der Optionen `LOBS TO` und `LOBFILE` impliziert eine Aktivierung des Änderungswerts für den Dateityp `lobsinfile`. Standardmäßig werden LOB-Werte in denselben Pfad geschrieben wie die exportierten relationalen Daten. Sind LOB-Pfade mit der Option LOBS TO angegeben ist, wird das Dienstprogramm EXPORT reihum zwischen den Pfaden ausgeführt, um jeden erfolgreichen LOB-Wert in die entsprechende LOB-Datei zu schreiben. Sie können mit der Option `LOBFILE` auch Namen für die LOB-Ausgabedateien angeben. Bei Angabe der Option LOBFILE sieht das Format des LOB-Dateinamens wie folgt aus: `lob-dateispez.xxx.lob`. Dabei stellt `lob-dateispez` den Wert dar, der für die Option LOBFILE angegeben wurde, und `xxx` ist eine Folgenummer für die vom Dienstprogramm EXPORT erstellten LOB-Dateien. Andernfalls lautet das Format des LOB-Dateinamens `exportdateiname.xxx.lob`. Dabei steht `exportdateiname` für den Namen der exportierten Ausgabedatei, die für den Befehl EXPORT angegeben wurde, und `xxx` ist eine Folgenummer für die vom Dienstprogramm EXPORT erstellten LOB-Dateien.

Standardmäßig werden LOBs in eine einzige Datei geschrieben. Sie können bei Bedarf jedoch auch angeben, dass einzelne LOB-Daten in separaten Dateien gespeichert werden sollen. Das Dienstprogramm EXPORT generiert eine LOB-Positions-

kennung (LLS, LOB Location Specifier), um ein Speichern von mehreren LOBs in einer einzigen Datei zu ermöglichen. Bei der LOB-Positionskennung, die in die Exportausgabedatei geschrieben wird, handelt es sich um eine Zeichenfolge, die die Position der LOB-Daten in der Datei angibt. Die LOB-Positionskennung hat folgendes Format: `lob-dateiname.erw.nnn.mmm/`. Hierbei steht `lob-dateiname.erw` für den Namen der Datei, die das große Objekt (LOB) enthält, `nnn` für die (in Byte angegebene) relative Position des großen Objekts in der Datei und 'mmm' für die Länge des großen Objekts (in Byte). Die LOB-Positionskennung `db2exp.001:123:456/` gibt beispielsweise an, dass sich das große Objekt in der Datei `db2exp.001` befindet, an einer relativen Position von 123 Byte in der Datei beginnt und 456 Byte lang ist. Wenn in der LOB-Positionskennung die Größe 0 angegeben ist, wird angenommen, dass das große Objekt die Länge 0 hat. Lautet der Wert für die Länge -1, wird angenommen, dass das große Objekt NULL ist, und relative Position sowie Dateiname werden ignoriert.

Wenn einzelne LOB-Daten nicht in einer Datei verknüpft werden sollen, können Sie mit dem Änderungswert `lobsinsepfiles` für den Dateityp angeben, dass die einzelnen LOBs in eine separate Datei geschrieben werden.

Anmerkung: Das IXF-Dateiformat speichert die LOB-Optionen der Spalte (beispielsweise über die Protokollierung der LOB-Spalte) nicht. Dies bedeutet, dass das Dienstprogramm IMPORT eine Tabelle mit einer LOB-Spalte, deren Größe mit 1 GB oder mehr definiert ist, nicht erneut erstellen kann.

Beispiel 1
Das folgende Beispiel veranschaulicht den Export von LOBs in eine DEL-Datei. Die exportierten LOB-Dateien werden mit dem angegebenen Basisnamen `lobs1` bezeichnet.

```
db2 export to myfile.del of del lobs to mylobs/
   lobfile lobs1 modified by lobsinfile
   select * from emp_photo
```

Beispiel 2
Das folgende Beispiel veranschaulicht, wie LOBs in eine DEL-Datei exportiert werden können, wobei die einzelnen LOB-Werte jeweils in eine separate Datei und die LOB-Dateien in zwei verschiedene Verzeichnisse geschrieben werden:

```
db2 export to myfile.del of del
lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
select * from emp_photo
```

Dienstprogramm IMPORT

Dienstprogramm IMPORT - Übersicht

Das Dienstprogramm IMPORT füllt Tabellen, typisierte Tabellen und Sichten mit Daten, die über eine SQL-Anweisung INSERT eingefügt werden. Wenn die Tabelle oder Sicht, die die importierten Daten aufnehmen soll, bereits Daten enthält, können diese Daten durch die importierten Daten ersetzt oder durch sie ergänzt werden.

Bei dem Dienstprogramm IMPORT handelt es sich wie bei dem Dienstprogramm EXPORT um ein einfaches Dienstprogramm zum Versetzen von Daten. Das Dienstprogramm kann durch die Ausgabe von CLP-Befehlen, einen Aufruf der gespeicherten Prozedur ADMIN_CMD oder einen Aufruf der API db2Import über eine Benutzeranwendung aktiviert werden.

Das Dienstprogramm IMPORT unterstützt eine Reihe von Datenformaten und Funktionen für den Import:

- Das Dienstprogramm IMPORT unterstützt die Datenformate IXF, ASC und DEL.
- Verwendung von Änderungswerten für den Dateityp zum Anpassen der Importoperation
- Versetzen hierarchischer Daten und typisierter Tabellen
- Protokollieren aller Aktivitäten, Aktualisieren von Indizes, Überprüfen von Integritätsbedingungen und Auslösen von Triggern
- Angabe von Spaltennamen für die Tabellen oder Sichten, in die die Daten eingefügt werden sollen
- Verwendung mit DB2 Connect

Importmodus

Als Importmodus sind fünf unterschiedliche Betriebsarten verfügbar, die die beim Datenimport verwendete Methode bestimmen. Die ersten drei Betriebsarten, `INSERT`, `INSERT_UPDATE` und `REPLACE`, werden verwendet, wenn die Zieltabellen bereits vorhanden sind. Alle drei unterstützen die Datenformate IXF, ASC und DEL. Mit Kurznamen können jedoch nur die Betriebsarten `INSERT` und `INSERT_UPDATE` verwendet werden.

Tabelle 12. Übersicht über die Importmodi INSERT, INSERT_UPDATE und REPLACE

Modus	**Empfohlene Methode für die Verwendung**
INSERT	Fügt die Eingabedaten in die Zieltabelle ein, ohne die vorhandenen Daten zu ändern.
INSERT_UPDATE	Aktualisiert Zeilen mit übereinstimmenden Primärschlüsselwerten mit den Werten von Eingabezeilen. Ist keine Zeilenentsprechung vorhanden, wird die importierte Zeile in die Tabelle eingefügt.
REPLACE	Löscht alle vorhandenen Daten und fügt importierte Daten unter Beibehaltung von Tabellen- und Indexdefinitionen ein.

Die übrigen Betriebsarten, `REPLACE_CREATE` und `CREATE`, werden verwendet, wenn es sich bei den Zieltabellen um neue Tabellen handelt. Diese Betriebsarten können nur für Eingabedateien im Format PC/IXF verwendet werden, bei dem eine strukturierte Beschreibung der jeweils zu erstellenden Tabelle enthalten ist. Importe mit diesen Betriebsarten können nicht durchgeführt werden, wenn die Objekttabelle außer sich selbst noch andere abhängige Tabellen aufweist.

Anmerkung: Die Importbetriebsarten `CREATE` und `REPLACE_CREATE` sind veraltet. Verwenden Sie stattdessen das Dienstprogramm **db2look**.

Tabelle 13. Übersicht über die Importmodi REPLACE_CREATE und CREATE

Modus	**Empfohlene Methode für die Verwendung**
REPLACE_CREATE	Löscht alle vorhandenen Daten und fügt importierte Daten unter Beibehaltung von Tabellen- und Indexdefinitionen ein. Erstellt Zieltabelle und Index, soweit noch nicht vorhanden.

Tabelle 13. Übersicht über die Importmodi REPLACE_CREATE und CREATE (Forts.)

Modus	Empfohlene Methode für die Verwendung
CREATE	Erstellt Zieltabelle und Index. Sie können den Namen des Tabellenbereichs für die neue Tabelle angeben.

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Importieren von Daten. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Vorgehensweise beim Import

Die Anzahl der für den Import erforderlichen Schritte und die Dauer des Importvorgangs richtet sich nach dem Umfang der zu versetzenden Daten und den von Ihnen angegebenen Optionen. Der Importvorgang beinhaltet folgende Schritte:

1. Sperren von Tabellen
   Beim Import wird, je nachdem ob Sie einen gleichzeitigen Tabellenzugriff zulassen, eine exklusive (X) oder nicht exklusive Sperre (IX) für vorhandene Zieltabellen angefordert.
2. Suchen und Abrufen von Daten
   Beim Import wird die Klausel FROM zum Suchen der Eingabedaten verwendet. Wenn Sie im Befehl angegeben haben, dass XML- oder LOB-Daten vorhanden sind, werden diese Daten beim Import gesucht.
3. Einfügen von Daten
   Beim Import werden vorhandene Daten ersetzt oder neue Datenzeilen an eine vorhandene Tabelle angehängt.
4. Prüfung auf Integritätsbedingungen und Auslösen von Triggern
   Beim Schreiben der Daten stellt das Dienstprogramm IMPORT sicher, dass die einzelnen eingefügten Zeilen jeweils den für die Zieltabelle definierten Integritätsbedingungen entsprechen. Angaben zu zurückgewiesenen Zeilen werden in die Nachrichtendatei geschrieben. Beim Import werden darüber hinaus auch vorhandene Trigger ausgelöst.
5. Commit der Operation
   Das Dienstprogramm IMPORT speichert die vorgenommenen Änderungen und gibt die Sperren der Zieltabelle frei. Sie können auch angeben, dass beim Import in regelmäßigen Abständen ein Commit durchgeführt werden soll.

Für eine einfache Importoperation sind folgende Angaben erforderlich:

- Pfad und Name der Eingabedatei
- Name oder Aliasname der Zieltabelle oder -sicht
- Format der Daten in der Eingabedatei
- Für den Datenimport zu verwendende Methode
- Traversierfolge (beim Import hierarchischer Daten)
- Liste untergeordneter Tabellen (beim Import typisierter Tabellen)

Weitere Optionen

Sie können die Importoperationen mit einer Reihe von Parametern anpassen. Für die Klausel MODIFIED BY können Sie Änderungswerte für den

Dateityp angeben, um das Datenformat zu ändern, um dem Dienstprogramm anzugeben, wie mit den Daten zu verfahren ist, und um die Leistungswerte zu optimieren.

Vom Dienstprogramm IMPORT werden standardmäßig bis zum erfolgreichen Abschluss des Imports keine Commitoperationen durchgeführt. Ausgenommen sind lediglich bestimmte, mit `ALLOW WRITE ACCESS` definierte Importoperationen. Dies erhöht die Importgeschwindigkeit. Es kann jedoch für den gemeinsamen Zugriff, Neustartvorgänge und den aktiven Protokollspeicherbereich sinnvoll sein, eine Durchführung von Commitoperationen auch während des Imports zu definieren. Dies kann erreicht werden, indem der Parameter **`COMMITCOUNT`** auf "automatic" gesetzt wird. Bei Angabe dieses Werts ermittelt das Importprogramm intern, wann ein Commit durchgeführt werden sollte. Sie können jedoch stattdessen auch eine bestimmte Zahl für den Parameter **`COMMITCOUNT`** definieren, um anzugeben, dass ein Commit nach dem Import der angegebenen Anzahl von Datensätzen durchgeführt werden soll.

Die Importleistung kann auf verschiedene Arten optimiert werden. Da es sich bei dem Dienstprogramm IMPORT um eine integrierte SQL-Anwendung handelt, die SQL-Abrufe intern durchführt, profitiert das Dienstprogramm IMPORT auch von Optimierungen, die bei den SQL-Operationen vorgenommen werden. Mit dem Änderungswert `compound` für den Dateityp können Sie angeben, dass eine bestimmte Anzahl von Zeilen gleichzeitig eingefügt werden soll, anstatt beim Einfügen zeilenweise vorzugehen. Wenn Sie davon ausgehen, dass beim Import voraussichtlich eine große Anzahl von Warnungen generiert wird (mit der daraus resultierenden Verlangsamung des Imports), können Sie den Änderungswert `norowwarnings` für den Dateityp angeben, um Warnungen zu zurückgewiesenen Zeilen zu unterdrücken.

Nachrichtendatei

Beim Import werden ASCII-Standardtextnachrichtendateien für die zugehörigen Fehlernachrichten, Informationsnachrichten oder Warnungen geschrieben. Wird das Dienstprogramm über die API db2Import aufgerufen, ist eine vorherige Angabe des Dateinamens für diese Dateien über den Parameter **`MESSAGES`** erforderlich. Andernfalls ist diese Angabe optional. Die Nachrichtendatei für eine Importoperation kann während des Imports aufgerufen werden und ermöglicht dadurch eine einfache Überwachung des Verarbeitungsfortschritts. Sollten Importoperationen fehlschlagen, kann anhand der Angabe zur letzten erfolgreich importierten Zeile in den Nachrichtendateien ermittelt werden, an welcher Stelle ein Wiederanlauf ansetzen muss.

Anmerkung: Überschreitet der Umfang der Ausgabenachrichten, die durch eine Importoperation für eine ferne Datenbank generiert werden, eine Größe von 60 KB, behält das Dienstprogramm die ersten 30 KB und die letzten 30 KB bei.

Für das Dienstprogramm IMPORT erforderliche Zugriffsrechte und Berechtigungen

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen und auf diese zuzugreifen. Berechtigungsstufen stellen eine Methode dar, um Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zusammenzufassen. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte.

Benutzer können nur auf die Objekte zugreifen, für die sie zugriffsberechtigt sind, d. h. für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Benutzer mit der Berechtigung DATAACCESS können alle Typen von Importoperationen ausführen. In der nachfolgenden Tabelle werden weitere Berechtigungen für beim Import beteiligte Tabellen, Sichten und Kurznamen aufgeführt, die zum Ausführen des jeweiligen Importtyps erforderlich sind.

Tabelle 14. Zum Ausführen von Importoperationen erforderliche Berechtigungen

Modus	Erforderliche Berechtigung
INSERT	CONTROL oder INSERT und SELECT
INSERT_UPDATE	CONTROL oder INSERT, SELECT, UPDATE und DELETE
REPLACE	CONTROL oder INSERT, SELECT und DELETE
REPLACE_CREATE	Wenn die Zieltabelle bereits vorhanden ist: CONTROL oder INSERT, SELECT und DELETE Wenn die Zieltabelle noch nicht vorhanden ist: CREATETAB (für die Datenbank), USE (für den Tabellenbereich) und wenn das Schema nicht vorhanden ist: IMPLICIT_SCHEMA (für die Datenbank) oder wenn das Schema vorhanden ist: CREATEIN (für das Schema)
CREATE	CREATETAB (für die Datenbank), USE (für den Tabellenbereich) und wenn das Schema nicht vorhanden ist: IMPLICIT_SCHEMA (für die Datenbank) oder wenn das Schema vorhanden ist: CREATEIN (für das Schema)

Anmerkung: Die Optionen **CREATE** und **REPLACE_CREATE** des Befehls **IMPORT** sind veraltet und werden in zukünftigen Releases möglicherweise entfernt.
Zum Verwenden der Option **REPLACE** oder **REPLACE_CREATE** für eine Tabelle muss die Berechtigungs-ID der Sitzung über die Berechtigung zum Löschen der jeweiligen Tabelle verfügen.

Wenn Sie eine Hierarchie importieren möchten, richtet sich die erforderliche Berechtigung auch nach dem Modus. Bei vorhandenen Hierarchien reicht das Zugriffsrecht CONTROL für alle untergeordneten Tabellen in der Hierarchie für eine Operation **REPLACE** aus. Bei nicht vorhandenen Hierarchien reicht die Berechtigung CONTROL für alle untergeordneten Tabellen in der Hierarchie in Verbindung mit der Berechtigung CREATETAB und USE für eine Operation **REPLACE_CREATE** aus.

Darüber hinaus müssen beim Importieren in Tabellen mit kennsatzbasierter Zugriffssteuerung (LBAC) bestimmte Aspekte berücksichtigt werden. Um Daten in eine Tabelle importieren zu können, die geschützte Spalten enthält, muss die Berechtigungs-ID der Sitzung über LBAC-Berechtigungsnachweise verfügen, die Schreibzugriffe auf alle geschützten Spalten in der Tabelle zulassen. Um Daten in eine Tabelle importieren zu können, die geschützte Zeilen enthält, muss der Berechtigungs-ID der Sitzung mit Grant ein Sicherheitskennsatz für Schreibzugriff erteilt werden, der Teil der Sicherheitsrichtlinie zum Schutz der Tabelle ist.

Importieren von Daten

Das Dienstprogramm IMPORT fügt Daten aus einer externen Datei mit einem unterstützten Dateiformat in eine Tabelle, eine Hierarchie, eine Sicht oder einen Kurznamen ein.

Das Dienstprogramm LOAD ist eine schnellere Alternative, unterstützt jedoch nicht das Laden von Daten auf Hierarchieebene.

Vorbereitende Schritte

Bevor Sie das Dienstprogramm IMPORT aufrufen, müssen Sie mit der Datenbank verbunden sein, in die die Daten importiert werden sollen, oder in der Lage sein, implizit eine Verbindung zu dieser Datenbank herzustellen. Wenn das implizite Herstellen von Verbindungen aktiviert ist, wird eine Verbindung zur Standarddatenbank hergestellt.

Beim Zugriff des Dienstprogramms auf DB2 für Linux-, UNIX- oder Windows-Datenbankserver über DB2 für Linux-, UNIX- oder Windows-Clients muss eine Direktverbindung über die Steuerkomponente bestehen. Der Zugriff des Dienstprogramms kann nicht über ein DB2 Connect-Gateway oder eine Loopback-Umgebung erfolgen.

Da das Dienstprogramm eine Anweisung COMMIT oder ROLLBACK absetzt, sollten Sie vor dem Aufrufen von IMPORT alle Transaktionen beenden und alle Sperren freigeben, indem Sie eine Anweisung COMMIT absetzen oder eine ROLLBACK-Operation ausführen.

Anmerkung: Die Parameter **CREATE** und **REPLACE_CREATE** des Befehls **IMPORT** sind veraltet und werden in zukünftigen Releases möglicherweise entfernt.

Einschränkungen

Für das Dienstprogramm IMPORT gelten die folgenden Einschränkungen:

- Ist die vorhandene Tabelle eine übergeordnete Tabelle mit einem Primärschlüssel, auf den ein Fremdschlüssel einer abhängigen Tabelle verweist, können keine Daten ersetzt werden. Es ist nur möglich, die neuen Daten anzufügen.
- Es ist nicht möglich, eine IMPORT REPLACE-Operation für eine zugrunde liegende Tabelle einer MQT (Materialized Query Table, gespeicherte Abfragetabelle) auszuführen, die im Modus "Sofort aktualisieren" (REFRESH IMMEDIATE) definiert wurde.
- Sie können keine Daten in eine Systemtabelle, eine Übersichtstabelle oder in eine Tabelle mit Spalten strukturierten Typs importieren.
- Sie können keine Daten in deklarierte temporäre Tabellen importieren.
- Mit dem Dienstprogramm IMPORT können keine Sichten erstellt werden.
- Referenzielle Integritätsbedingungen und Fremdschlüsseldefinitionen werden beim Erstellen von Tabellen aus PC/IXF-Dateien nicht beibehalten. (Primärschlüsseldefinitionen *werden* beibehalten, wenn die Daten zuvor mit `SELECT *` exportiert wurden.)
- Da das Dienstprogramm IMPORT eigene SQL-Anweisungen generiert, kann die maximale Anweisungsgröße von 2 MB möglicherweise in einigen Fällen überschritten werden.

- Sie können eine partitionierte Tabelle oder eine MDC-Tabelle (Multidimensional Clustering Table - mehrdimensionale Clusteringtabelle) nicht mit dem Importparameter **CREATE** oder **REPLACE_CREATE** erneut erstellen.
- Tabellen, die XML-Spalten enthalten, können nicht erneut erstellt werden.
- Es können keine verschlüsselten Daten importiert werden.
- Die Operation IMPORT REPLACE berücksichtigt die Klausel NOT LOGGED INITIALLY nicht. Der Parameter **REPLACE** des Befehls **IMPORT** berücksichtigt die Klausel NOT LOGGED INITIALLY (NLI) der Anweisung CREATE TABLE oder die Klausel ACTIVATE NOT LOGGED INITIALLY der Anweisung ALTER TABLE nicht. Wenn innerhalb einer Transaktion mit einer Anweisung CREATE TABLE oder ALTER TABLE, bei der die Klausel NLI aufgerufen wird, ein Import mit der Aktion **REPLACE** vorgenommen wird, wird die Klausel NLI beim Import nicht berücksichtigt. In diesem Szenario werden alle Einfügungen protokolliert.

  Ausweichmaßnahme 1: Löschen Sie den Inhalt der Tabelle mit der Anweisung DELETE. Rufen Sie die Importoperation dann mit der Anweisung INSERT auf.

  Ausweichmaßnahme 2: Löschen Sie die Tabelle, und erstellen Sie sie erneut. Rufen Sie die Importoperation dann mit der Anweisung INSERT auf.

Die folgende Einschränkung gilt für das Dienstprogramm IMPORT: Überschreitet der Umfang der Ausgabenachrichten, die durch eine Importoperation für eine ferne Datenbank generiert wurden, eine Größe von 60 KB, behält das Dienstprogramm die ersten 30 KB und die letzten 30 KB bei.

Vorgehensweise

Gehen Sie wie folgt vor, um das Dienstprogramm IMPORT aufzurufen:

- Setzen Sie den Befehl **IMPORT** im Befehlszeilenprozessor ab.
- Rufen Sie die Anwendungsprogrammierschnittstelle (API) db2Import über eine Clientanwendung auf.
- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **IMPORT**.

Beispiel

Einfache Importoperationen erfordern lediglich die Angabe der Eingabedatei, des Dateiformats, des Importmodus und der Zieldatei (oder des Namens der zu erstellenden Tabelle).

Geben Sie zum Importieren von Daten über den Befehlszeilenprozessor den Befehl **IMPORT** ein:

```
db2 import from dateiname of dateiformat importmodus into tabelle
```

Dabei ist *dateiname* der Name der Eingabedatei mit den zu importierenden Daten, *dateiformat* das Dateiformat, *importmodus* der Modus und *tabelle* der Name der Tabelle, in die die Daten eingefügt werden sollen.

Es kann jedoch auch sinnvoll sein, eine Nachrichtendatei, in die Warnungen und Fehlernachrichten geschrieben werden können, anzugeben. Fügen Sie dazu den Parameter **MESSAGES** und den Namen einer Nachrichtendatei hinzu. Beispiel:

```
db2 import from dateiname of dateiformat messages nachrichtendatei importmodus into tabelle
```

Importieren von XML-Daten

Das Dienstprogramm IMPORT kann zum Importieren von XML-Daten in eine XML-Tabellenspalte eingesetzt werden, indem entweder der Tabellenname oder ein Kurzname für ein Quellendatenobjekt von DB2 for Linux, UNIX and Windows verwendet wird.

Beim Importieren von Daten in eine XML-Tabellenspalte können Sie die Option XML FROM verwenden, um die Pfade der XML-Eingabedatei oder der XML-Eingabedateien anzugeben. Für die XML-Datei "/home/user/xmlpath/xmldocs.001.xml", die zuvor exportiert wurde, können Sie beispielsweise den folgenden Befehl verwenden, um die Daten zurück in die Tabelle zu importieren:

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

Eingefügte Dokumente anhand von Schemata prüfen

Die Option XMLVALIDATE ermöglicht die Gültigkeitsprüfung von XML-Dokumenten mithilfe von XML-Schemata, während diese importiert werden. Im folgenden Beispiel wird die Gültigkeit eingehender XML-Dokumente auf der Basis der Schemainformationen überprüft, die während des Exports der XML-Dokumente gespeichert wurden:

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
            USING XDS INSERT INTO USER.T1
```

Optionen für das Parsing angeben

Mit der Option XMLPARSE können Sie angeben, ob Leerzeichen in den importierten XML-Dokumenten beim Parsing (d. h. bei der syntaktischen Analyse) beibehalten oder gelöscht werden sollen. Im folgenden Beispiel wird die Gültigkeit aller importierten XML-Dokumente auf der Basis der XML-Schemainformationen überprüft, die gespeichert wurden, als die XML-Dokumente exportiert wurden. Für diese Dokumente werden die Leerzeichen beim Parsing beibehalten.

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
WHITESPACE XMLVALIDATE USING XDS INSERT INTO USER.T1
```

CLP-Beispiele für IMPORT-Sitzungen

Beispiel 1

Das folgende Beispiel zeigt, wie Sie die Daten aus `myfile.ixf` in die Tabelle STAFF importieren können:

```
db2 import from myfile.ixf of ixf messages msg.txt insert into staff

SQL3150N  Der H-Satz in der PC/IXF-Datei enthält das Produkt "DB2
01.00", das Datum "19970220" und die Zeit "140848".

SQL3153N  Der T-Satz in der PC/IXF-Datei hat den Namen "myfile",
das Qualifikationsmerkmal "        " und die Quelle "            ".

SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei "myfile".

SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "58" Zeile(n)
aus der Eingabedatei gelesen.

SQL3221W  ...COMMIT WORK beginnt. Zähler für Eingabesätze: "58".

SQL3222W  ...COMMIT für alle Änderungen der Datenbank wurde
erfolgreich ausgeführt.

SQL3149N  "58" Zeile(n) aus der Eingabedatei wurde(n) verarbeitet. "58" Zeile(n) wurde(n)
der Tabelle hinzugefügt. "0" Zeile(n) wurde(n) zurückgewiesen.
```

Beispiel 2

Das folgende Beispiel zeigt, wie Daten in eine Tabelle mit Identitätsspalten importiert werden können:

TABLE1 enthält 4 Spalten:
- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 entspricht TABLE1, wobei jedoch C2 eine als GENERATED ALWAYS definierte Identitätsspalte ist.

Die Datensätze in DATAFILE1 (DEL-Format) lauten:

```
"Liszt"
"Holst",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

Die Datensätze in DATAFILE2 (DEL-Format) lauten:

```
"Liszt", 74.49, A
"Holst", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

Der folgende Befehl generiert Identitätswerte für die Zeilen 1 und 2, da in der Datei DATAFILE1 für diese Zeilen keine Identitätswerte zur Verfügung gestellt werden. Den Zeilen 3 und 4 wird jedoch der benutzerdefinierte Identitätswert 100 bzw. 101 zugeordnet.

```
db2 import from datafile1.del of del replace into table1
```

Um die Datei DATAFILE1 so in die Tabelle TABLE1 zu importieren, dass für alle Zeilen Identitätswerte generiert werden, setzen Sie einen der folgenden Befehle ab:

```
db2 import from datafile1.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
db2 import from datafile1.del of del modified by identityignore
   replace into table1
```

Um die Datei DATAFILE2 so in die Tabelle TABLE1 zu importieren, dass für alle Zeilen Identitätswerte generiert werden, setzen Sie einen der folgenden Befehle ab:

```
db2 import from datafile2.del of del replace into table1 (c1, c3, c4)
db2 import from datafile2.del of del modified by identitymissing
   replace into table1
```

Wird die Datei DATAFILE1 in die Tabelle TABLE2 importiert, ohne dass einer der identitätsbezogenen Änderungswerte für den Dateityp verwendet wird, werden die Zeilen 1 und 2 eingefügt, die Zeilen 3 und 4 jedoch zurückgewiesen, da diese Zeilen eigene Werte, die keine Nullwerte sind, liefern und die Identitätsspalte als GENERATED ALWAYS definiert wurde.

Beispiel 3

Das folgende Beispiel zeigt, wie Daten in eine Tabelle mit Nullanzeigern importiert werden können:

TABLE1 enthält 5 Spalten:

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 enthält 6 Elemente:

- ELE1 Positionen 01 bis 20
- ELE2 Positionen 21 bis 22
- ELE5 Positionen 23 bis 23
- ELE3 Positionen 24 bis 27
- ELE4 Positionen 28 bis 31
- ELE6 Positionen 32 bis 32
- ELE6 Positionen 33 bis 40

Datensätze:

```
1...5....10...15...20...25...30...35...40
Testdaten 1         XXN 123abcdN
Testdaten 2 und 3   QQY    wxyzN
Testdaten 4,5 und 6 WWN6789    Y
```

Mit dem folgenden Befehl werden Datensätze aus ASCFILE1 in TABLE1 importiert:

```
db2 import from ascfile1 of asc
method L (1 20, 21 22, 24 27, 28 31)
null indicators (0, 0, 23, 32)
insert into table1 (col1, col5, col2, col3)
```

Anmerkung:

1. Da COL4 nicht in der Eingabedatei bereitgestellt wird, wird sie in TABLE1 mit dem Standardwert eingefügt (sie ist mit NOT NULL WITH DEFAULT definiert).
2. Die Positionen 23 und 32 werden verwendet, um anzugeben, ob in COL2 und COL3 von TABLE1 für eine gegebene Zeile NULL geladen wird. Wenn die Nullanzeigerposition der Spalte für einen gegebenen Satz `Y` enthält, ist die Spalte NULL. Enthält sie `N`, werden die Datenwerte in den Datenpositionen der Spalte des Eingabedatensatzes (wie in L(.....) definiert) als Quelle der Spaltendaten für die Zeile verwendet. In diesem Beispiel ist keine Spalte in Zeile 1 NULL. COL2 in Zeile 2 ist NULL, und COL3 in Zeile 3 ist NULL.
3. In diesem Beispiel werden die NULL-Anzeiger für COL1 und COL5 als 0 (Null) angegeben, wodurch festgelegt wird, dass die Daten keine Nullwerte enthalten dürfen.
4. Der NULL-Anzeiger für eine bestimmte Spalte kann sich an einer beliebigen Position im Eingabedatensatz befinden. Diese Position muss jedoch angegeben werden, und der Wert `Y` oder `N` muss ebenfalls angegeben werden.

Neuerstellung von importierten Tabellen

Mit dem Modus `CREATE` des Dienstprogramms IMPORT können Sie eine Tabelle erneut erstellen, die mit dem Dienstprogramm EXPORT gesichert wurde. Dabei sind jedoch verschiedene Einschränkungen zu beachten, da viele Attribute der Eingabetabelle nicht beibehalten werden.

Beim Import können Tabellen nur erneut erstellt werden, wenn die vorangehende Exportoperation bestimmte Anforderungen erfüllt. Die ursprüngliche Tabelle muss in eine IXF-Datei exportiert werden. Wenn Sie Dateien mit dem DEL- oder ASC-Dateiformat exportieren, enthalten die Ausgabedateien zwar die Satzdaten, sie enthalten jedoch keine Beschreibungen. Um eine Tabelle mit Daten erneut zu erstellen, die in diesen Dateiformaten gespeichert wurden, erstellen Sie die Zieltabelle und verwenden anschließend das Dienstprogramm LOAD oder IMPORT, um die Tabelle aus diesen Dateien zu füllen. Mit dem Dienstprogramm **db2look** können Sie die ursprünglichen Tabellendefinitionen erfassen und die entsprechende Datendefinitionssprache (DDL) generieren. Darüber hinaus darf die Anweisung SELECT für den Export nur bestimmte Aktionszeichenfolgen enthalten. Es dürfen z. B. keine Spaltennamen in der Klausel SELECT verwendet werden, und es ist nur die Verwendung von "SELECT *" zulässig.

Anmerkung: Der Importmodus `CREATE` ist veraltet. Verwenden Sie das Dienstprogramm **db2look**, um Ihre Tabellen zu erfassen und erneut zu erstellen.

Beibehaltene Attribute

In der erneut erstellten Tabelle werden folgende Attribute der Originaltabelle beibehalten:

- Name des Primärschlüssels und Definition
- Spalteninformationen, z. B.:
  - Spaltenname
  - Spaltendatentypen, einschließlich benutzerdefinierte einzigartige Datentypen, die als Basistyp beibehalten werden
  - Identitätseigenschaften
  - Längen (außer bei Typen lob_file)
  - Codepage (sofern zutreffend)
  - Identitätsoptionen
  - Angabe, ob die Spalte laut Definition Nullwerte enthalten darf oder nicht
  - Standardwerte für Konstanten, sofern vorhanden, jedoch keine anderen Arten von Standardwerten
- Indexinformationen, z. B.:
  - Indexname
  - Name des Indexerstellers
  - Spaltennamen sowie die Angabe, ob die einzelnen Spalten in aufsteigender oder absteigender Reihenfolge sortiert sind
  - Angabe, ob der Index als eindeutig definiert ist
  - Angabe, ob es sich um einen Clusterindex handelt
  - Angabe, ob der Index Rückwärtsdurchsuchen zulässt
  - Werte für PCTFREE
  - Werte für MINPCTUSED

Anmerkung: Enthalten die im Index angegebenen Spaltennamen das Zeichen - oder +, werden keine Indexinformationen beibehalten. In diesem Fall wird die Warnung SQL27984W zurückgegeben.

Nicht beibehaltene Attribute

Die erneut erstellte Tabelle behält verschiedene Attribute der Originaltabelle nicht bei, z. B. folgende Attribute:

- Angabe, ob die Quelle eine normale Tabelle, eine MQT (Materialized Query Table, gespeicherte Abfragetabelle), eine Sicht oder eine Spaltengruppe aus einer oder allen diesen Quellen ist
- Eindeutige Integritätsbedingungen und andere Typen von Integritätsbedingungen oder Triggern (Integritätsbedingungen über Primärschlüssel ausgeschlossen)
- Tabelleninformationen, z. B.:
  - MQT-Definition (sofern zutreffend)
  - MQT-Optionen (sofern zutreffend)
  - Optionen für den Tabellenbereich (diese Informationen können jedoch mit dem Befehl **IMPORT** angegeben werden)
  - Dimensionen für mehrdimensionales Clustering (MDC)
  - Dimensionen für partitionierte Tabellen
  - Tabellenpartitionierungsschlüssel
  - Eigenschaft NOT LOGGED INITIALLY
  - Prüfung auf Integritätsbedingung
  - Tabellencodepage
  - Eigenschaften geschützter Tabellen
  - Optionen für die Komprimierung von Tabellen oder Werten
- Spalteninformationen, z. B.:
  - Gegebenenfalls vorhandene Standardwerte (außer Werte für Konstanten)
  - LOB-Optionen (sofern vorhanden)
  - XML-Eigenschaften
  - Verweisklausel der Anweisung CREATE TABLE (sofern vorhanden)
  - Referenzielle Integritätsbedingungen (sofern vorhanden)
  - Prüfung auf Integritätsbedingungen (sofern vorhanden)
  - Optionen für generierte Spalten (sofern vorhanden)
  - Von Datenbankbereichsfolgen abhängige Spalten
  - Implizit verdeckte Eigenschaft
- Indexinformationen, z. B.:
  - INCLUDE-Spalten (sofern vorhanden)
  - Indexname, wenn es sich bei dem Index um einen Primärschlüsselindex handelt
  - Absteigende Reihenfolge der Schlüssel, wenn es sich bei dem Index um einen Primärschlüsselindex handelt (die Standardeinstellung ist die aufsteigende Reihenfolge)
  - Indexspaltennamen, die den Hexadezimalwert 0x2B oder 0x2D enthalten
  - Indexnamen, die nach der Codepagekonvertierung mehr als 128 Byte enthalten
  - Wert für PCTFREE2
  - Eindeutige Integritätsbedingungen

Anmerkung: Diese Liste ist nicht vollständig und deshalb lediglich eine Orientierungshilfe.

Wenn die Importoperation fehlschlägt und die Nachricht SQL3311N zurückgegeben wird, können Sie die Tabelle auf jeden Fall mit dem Änderungswert `forcecreate` für den Dateityp erneut erstellen. Dieser Änderungswert ermöglicht es Ihnen, die Tabelle mit fehlenden oder beschränkt gültigen Informationen zu erstellen.

Aspekte des Imports typisierter Tabellen

Mit dem Dienstprogramm IMPORT können Daten sowohl aus typisierten Tabellen als auch in typisierten Tabellen unter Beibehaltung der vorhandenen Datenhierarchie versetzt werden. Bei Bedarf können Tabellenhierarchie und Typhierarchie beim Import auch erstellt werden.

Das Versetzen von Daten aus einer hierarchischen Struktur mit typisierten Tabellen in eine andere erfolgt entsprechend einer bestimmten Traversierfolge und durch Erstellung einer temporären Flachdatei während des Exports. Das Dienstprogramm IMPORT wiederum steuert Größe und Position der zu versetzenden Hierarchie mit den Parametern **CREATE**, **INTO** *tabellenname*, **UNDER** und **AS ROOT TABLE**. Das Dienstprogramm IMPORT steuert ebenfalls die Einfügungen in die Zieltabelle. Das Dienstprogramm IMPORT kann z. B. am Ende jedes Namens einer untergeordneten Tabelle eine Attributliste angeben, um festzulegen, dass nur bestimmte Attribute in die Zieldatenbank versetzt werden sollen. Wenn keine Attributliste angegeben ist, werden alle Spalten der einzelnen untergeordneten Tabellen versetzt.

Erneute Erstellung von Tabellen

Der verfügbare Importtyp richtet sich nach dem Dateiformat der Eingabedatei. Bei ASC- oder DEL-Datendateien muss die Zieltabelle bzw. Zielhierarchie vor dem Datenimport bereits vorhanden sein. Daten aus einer PC/IXF-Datei können jedoch importiert werden, auch wenn die Tabelle oder Hierarchie noch nicht vorhanden ist, sofern Sie die Importoperation mit der Option **CREATE** durchführen. Bei Angabe der Option **CREATE** ist jedoch zu beachten, dass die Definitionen untergeordneter Tabellen beim Import nicht geändert werden können.

Traversierfolge

Die in der Eingabedatei enthaltene Traversierfolge ermöglicht das Beibehalten der Datenhierarchien. Beim Arbeiten mit den Dienstprogrammen EXPORT und IMPORT muss dieselbe Traversierfolge verwendet werden.

Beim PC/IXF-Dateiformat muss lediglich der Name der untergeordneten Zieltabelle angegeben und die in der Datei gespeicherte Traversierfolge verwendet werden.

Werden andere Optionen als **CREATE** bei typisierten Tabellen verwendet, können Sie über die Traversierfolgeliste die Traversierfolge angeben. Diese benutzerdefinierte Traversierfolge muss mit der Folge übereinstimmen, die während der Exportoperation verwendet wurde. Das exakte Versetzen von Daten in die Zieldatenbank ist bei dem Dienstprogramm IMPORT unter folgenden Bedingungen gesichert:

- Eine identische Definition von untergeordneten Tabellen in den Quellen- und Zieldatenbanken
- Eine identische hierarchische Beziehung zwischen untergeordneten Tabellen in den Quellen- und Zieldatenbanken
- Eine identische Traversierfolge

Beim Definieren der Traversierfolge legen Sie zwar den Startpunkt und den Pfad durch die Hierarchie fest, aber jede Verzweigung der Hierarchie muss zunächst komplett verarbeitet werden, bevor mit der nächsten Verzwei-

gung begonnen werden kann. Das Dienstprogramm IMPORT erkennt die Nichteinhaltung dieser Bedingung innerhalb der angegebenen Traversierfolge.

Beispiele

Die Beispiele in diesem Abschnitt sind an der folgenden hierarchischen Struktur mit vier gültigen Traversierfolgen ausgerichtet:

- Person, Employee, Manager, Architect, Student
- Person, Student, Employee, Manager, Architect
- Person, Employee, Architect, Manager, Student
- Person, Student, Employee, Architect, Manager

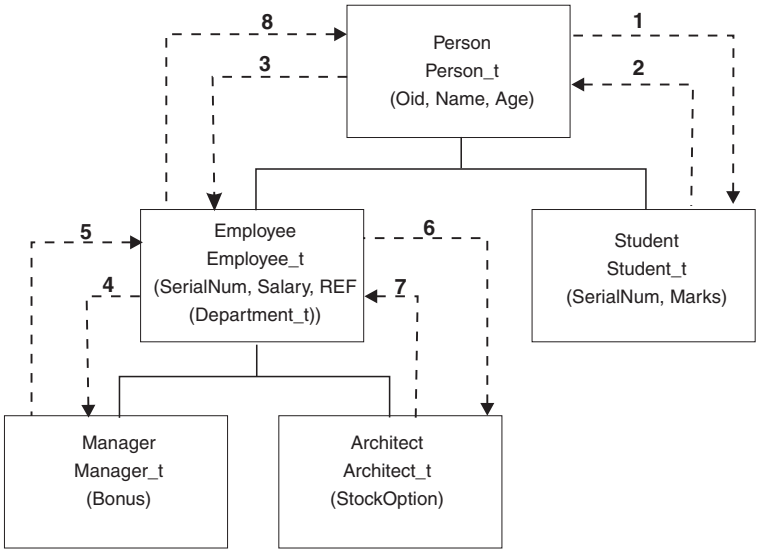


Abbildung 2. Hierarchiebeispiel

Beispiel 1
Zum erneuten Erstellen einer vollständigen Hierarchie (enthalten in der durch eine vorangehende Exportoperation erstellten Datendatei `entire_hierarchy.ixf`) beim Import müssen Sie folgende Befehle eingeben:

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.ixf OF IXF CREATE INTO
        HIERARCHY STARTING Person AS ROOT TABLE
```

Alle in der Hierarchie enthaltenen Typen werden erstellt, soweit sie noch nicht vorhanden sind. Wenn diese Typen bereits vorhanden sind, müssen sie in der Zieldatenbank die gleichen Definitionen aufweisen wie in der Quellendatenbank. Wenn die Definitionen nicht identisch sind, wird ein SQL-Fehler (SQL20013N) zurückgegeben. Da eine neue Hierarchie erstellt wird, gilt außerdem, dass keine der untergeordneten Tabellen vorhanden sein darf, die in der in die Zieldatenbank (Target_db) zu versetzenden Datendatei definiert sind. Alle angegebenen Tabellen aus der Hierarchie der Quellendatenbank werden erstellt. Zuletzt werden die Daten aus der Quellendatenbank in die richtigen untergeordneten Tabellen der Zieldatenbank importiert.

Beispiel 2
Zum erneuten Erstellen der gesamten Hierarchie der Quellendatenbank und Im-

portieren dieser Hierarchie in die Zieldatenbank unter Beibehaltung der ausgewählten Daten müssen Sie folgende Befehle eingeben:

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.del OF DEL INSERT INTO (Person,
         Employee(Salary), Architect) IN HIERARCHY (Person, Employee,
         Manager, Architect, Student)
```

Die Zieltabellen PERSON, EMPLOYEE und ARCHITECT müssen vorhanden sein. Daten werden in die untergeordneten Tabellen PERSON, EMPLOYEE und ARCHITECT importiert. Es wird also Folgendes importiert:

- Alle Spalten aus der Tabelle PERSON in die Tabelle PERSON
- Alle Spalten aus der Tabelle PERSON und die Spalte SALARY aus der Tabelle EMPLOYEE in die Tabelle EMPLOYEE
- Alle Spalten aus der Tabelle PERSON und die Spalte SALARY in EMPLOYEE sowie alle Spalten aus der Tabelle ARCHITECT in die Tabelle ARCHITECT

Die Spalten SerialNum und REF(Employee_t) werden weder in die Tabelle EMPLOYEE noch in die zugehörigen untergeordneten Tabellen importiert (hier ist ARCHITECT die einzige untergeordnete Tabelle, in die Daten importiert werden).

Anmerkung: Da ARCHITECT eine untergeordnete Tabelle von EMPLOYEE ist und als einzige Importspalte für EMPLOYEE die Spalte SALARY angegeben ist, wird SALARY als einzige spezifische Spalte der Tabelle EMPLOYEE in die Tabelle ARCHITECT importiert. Dies bedeutet, dass weder die Spalte SerialNum noch die Spalte REF(Employee_t) in die Zeilen EMPLOYEE oder ARCHITECT importiert werden.
Es werden keine Daten in die Tabellen MANAGER und STUDENT importiert.

Beispiel 3
Dieses Beispiel zeigt, wie Daten aus einer regulären Tabelle exportiert und als eine einzige untergeordnete Tabelle in eine Hierarchie importiert werden. Der Befehl **EXPORT** verarbeitet reguläre (nicht typisierte) Tabellen. Die Datendatei enthält deshalb keine Spalte "Type_id". Der Änderungswert `no_type_id` gibt diese Information an das Dienstprogramm IMPORT weiter, damit es nicht als erste Spalte die Spalte "Type_id" erwartet.

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO Student_sub_table.del OF DEL SELECT * FROM
   Regular_Student
DB2 CONNECT TO Target_db
DB2 IMPORT FROM Student_sub_table.del OF DEL METHOD P(1,2,3,5,4)
   MODIFIED BY NO_TYPE_ID INSERT INTO HIERARCHY (Student)
```

In diesem Beispiel muss die Zieltabelle STUDENT vorhanden sein. Da es sich bei STUDENT um eine untergeordnete Tabelle handelt, wird durch den Änderungswert `no_type_id` angegeben, dass die erste Spalte keine "Type_id" enthält. Sie müssen jedoch sicherstellen, dass die Tabelle STUDENT zusätzlich zu allen anderen Attributen der Tabelle eine Spalte "Object_id" enthält. "Object_id" wird als erste Spalte in jeder Zeile erwartet, die in die Tabelle STUDENT importiert wird. Die Klausel **METHOD** kehrt die Reihenfolge der beiden letzten Attribute um.

Aspekte des Imports von LBAC-geschützten Daten

Die Voraussetzung für einen erfolgreichen Import in eine Tabelle mit geschützten Zeilen sind LBAC-Berechtigungsnachweise (Label-Based Access Control). Darüber hinaus müssen Sie über einen gültigen Sicherheitskennsatz (oder einen Sicherheitskennsatz, der in einen gültigen Kennsatz konvertiert werden kann) für die Sicherheitsrichtlinie verfügen, die der Zieltabelle aktuell zugeordnet ist.

Wenn Sie nicht über gültige LBAC-Berechtigungsnachweise verfügen, schlägt der Import fehl, und es wird ein Fehler (SQLSTATE 42512) zurückgegeben. Enthalten die Eingabedaten keinen Sicherheitskennsatz oder liegt der Sicherheitskennsatz nicht im zugehörigen internen Binärformat vor, können Sie die Importoperation mithilfe verschiedener Änderungswerte für den Dateityp fortsetzen.

Wenn Sie Daten in eine Tabelle mit geschützten Zeilen importieren, enthält die Zieltabelle eine einzige Spalte mit dem Datentyp DB2SECURITYLABEL. Wenn die Eingabedatenzeile keinen Wert für diese Spalte enthält, wird die Zeile zurückgewiesen, es sei denn, der Importbefehl wurde mit dem Änderungswert `usedefaults` für den Dateityp angegeben. Wurde `usedefaults` im Importbefehl angegeben, wird der Sicherheitskennsatz, über den Sie für den Schreibzugriff verfügen, aus der Sicherheitsrichtlinie, die die Tabelle schützt, verwendet. Wenn Sie keinen Sicherheitskennsatz für den Schreibzugriff haben, wird die Zeile zurückgewiesen und mit der Verarbeitung der nächsten Zeile fortgefahren.

Wenn Sie Daten in eine Tabelle mit geschützten Zeilen importieren und die Eingabedaten einen Wert für die Spalte mit dem Datentyp DB2SECURITYLABEL enthalten, gelten dieselben Regeln wie beim Einfügen von Daten in die betreffende Tabelle. Wenn der Sicherheitskennsatz, der die zu importierende Zeile schützt, also der Kennsatz in der betreffenden Zeile der Datendatei, ein Kennsatz ist, der Ihnen den Schreibzugriff ermöglicht, wird der betreffende Sicherheitskennsatz zum Schutz der Zeile verwendet. Dies bedeutet, dass der Sicherheitskennsatz in die Spalte mit dem Datentyp DB2SECURITYLABEL geschrieben wird. Wenn Sie nicht in eine Zeile schreiben können, die durch diesen Sicherheitskennsatz geschützt wird, hat dies, je nachdem wie die Sicherheitsrichtlinie zum Schutz der Quellentabelle erstellt wurde, unterschiedliche Konsequenzen:

- Wenn die Anweisung CREATE SECURITY POLICY, mit der die Richtlinie erstellt wurde, die Option `RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL` enthielt, schlägt das Einfügen fehl und es wird ein Fehler zurückgegeben.
- Enthielt die Anweisung CREATE SECURITY POLICY diese Option nicht oder stattdessen die Option `OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL`, wird der Sicherheitskennsatz in der Datendatei für die betreffende Zeile ignoriert und stattdessen zum Schutz der Zeile der Sicherheitskennsatz verwendet, den Sie für den Schreibzugriff besitzen. In diesem Fall wird weder ein Fehler noch eine Warnung ausgegeben. Wenn Sie keinen Sicherheitskennsatz für den Schreibzugriff haben, wird die Zeile zurückgewiesen und mit der Verarbeitung der nächsten Zeile fortgefahren.

Hinweise zu Begrenzern

Wenn Sie Daten in eine Spalte des Typs DB2SECURITYLABEL importieren, wird standardmäßig davon ausgegangen, dass es sich bei dem Wert in der Datendatei um die tatsächlichen Byte handelt, die die interne Darstellung dieses Sicherheitskennsatzes bilden. Einige Rohdaten enthalten jedoch möglicherweise Zeichen für Zeilenvorschub, die vom Befehl IMPORT fälschlicherweise als Begrenzer für die Zeile interpretiert werden. Wenn dieses Problem vorliegt, können Sie mit dem Änderungswert `delprioritychar` für den Dateityp sicherstellen, dass der Zeichenbegrenzer Vorrang vor dem Zeilenbegrenzer hat. Wenn Sie `delprioritychar` verwenden, werden Zeilen- und Spaltenbegrenzer, die innerhalb der zwischen den Zeichenbegrenzern enthaltenen Daten vorliegen, nicht als Begrenzer erkannt. Auch wenn kein Wert ein Zeilenvorschubzeichen enthält, kann der Änderungswert `delprioritychar` für den Dateityp problemlos verwendet werden. Dies führt jedoch zu einer geringfügigen Verlangsamung der Importoperation.

Wenn die zu importierenden Daten im Format ASC (Abstract Syntax Checker) vorliegen, empfiehlt es sich unter Umständen, durch eine zusätzliche Maßnahme sicherzustellen, dass keine abschließenden Leerzeichen in die importierten Sicherheitskennsätze und Namen von Sicherheitskennsätzen eingefügt werden. Da beim ASCII-Format Spaltenpositionen als Begrenzer verwendet werden, kann dies beim Import in Felder variabler Länge vorkommen. Verwenden Sie zum Abschneiden von abschließenden Leerstellen den Änderungswert `striptblanks` für den Dateityp.

Vom Standard abweichende Sicherheitskennsatzwerte

Sie können auch Datendateien importieren, bei denen die Werte für die Sicherheitskennsätze Zeichenfolgen darstellen, die die Werte der Komponenten im jeweiligen Sicherheitskennsatz enthalten, z. B. *S:(ALPHA,BETA)*. Verwenden Sie hierzu den Änderungswert `seclabelchar` für den Dateityp. Bei Verwendung von `seclabelchar` werden Werte für Spalten mit dem Datentyp DB2SECURITYLABEL als Zeichenfolgekonstanten gewertet, die den Sicherheitskennsatz im Zeichenfolgeformat für Sicherheitskennsätze enthalten. Liegt eine Zeichenfolge nicht im richtigen Format vor, wird die jeweilige Zeile nicht eingefügt und eine Warnung (SQLSTATE 01H53) zurückgegeben. Stellt eine Zeichenfolge keinen gültigen Sicherheitskennsatz dar, der Bestandteil der zum Schutz der Tabelle verwendeten Sicherheitsrichtlinie ist, wird die betreffende Zeile nicht eingefügt und eine Warnung (SQLSTATE 01H53) zurückgegeben.

Sie können auch Datendateien importieren, bei denen die Werte für die Sicherheitskennsatzspalte Namen von Sicherheitskennsätzen sind. Verwenden Sie zum Importieren derartiger Dateien den Änderungswert `seclabelname` für den Dateityp. Bei Verwendung von `seclabelname` werden alle Werte für Spalten mit dem Typ DB2SECURITYLABEL als Zeichenfolgekonstanten gewertet, die den Namen vorhandener Sicherheitskennsätze enthalten. Existiert kein Sicherheitskennsatz mit dem angegebenen Namen für die zum Schutz der Tabelle verwendeten Sicherheitsrichtlinie, wird die betreffende Zeile nicht eingefügt und eine Warnung (SQLSTATE 01H53) zurückgegeben.

Beispiele

Bei allen Beispielen liegt die Eingabedatendatei `myfile.del` im Format DEL vor. Die Daten werden jeweils in eine Tabelle mit der Bezeichnung REPS importiert, die mit der folgenden Anweisung erstellt wurde:

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

Bei dem folgenden Beispiel wird davon ausgegangen, dass die Eingabedatei Sicherheitskennsätze im Standardformat enthält:

```
db2 import from myfile.del of del modified by delprioritychar insert into reps
```

Bei dem folgenden Beispiel wird davon ausgegangen, dass die Eingabedatei Sicherheitskennsätze im Zeichenfolgeformat für Sicherheitskennsätze enthält:

```
db2 import from myfile.del of del modified by seclabelchar insert into reps
```

Bei dem folgenden Beispiel wird davon ausgegangen, dass die Eingabedatei Namen von Sicherheitskennsätzen für die Sicherheitskennsatzspalte enthält:

```
db2 import from myfile.del of del modified by seclabelname insert into reps
```

Gepufferte INSERT-Operationen

In einer Umgebung mit partitionierten Datenbanken kann das Dienstprogramm IMPORT für die Verwendung gepufferter INSERT-Operationen aktiviert werden. Dadurch wird der Umfang der Nachrichtenübertragung beim Datenimport reduziert und die Leistung somit optimiert.

Die Option für gepufferte INSERT-Operationen sollten Sie nur aktivieren, wenn Sie auf eine Fehlerprotokollierung verzichten können, da Details zu fehlgeschlagenen gepufferten INSERT-Operationen nicht zurückgegeben werden.

Werden gepufferte INSERT-Operationen verwendet, setzt IMPORT den Wert für **WARNINGCOUNT** standardmäßig auf 1, sodass die Operation fehlschlägt, wenn eine beliebige Zeile zurückgewiesen wird. Wird ein Datensatz zurückgewiesen, setzt das Dienstprogramm die aktuelle Transaktion mittels eines Rollbacks zurück. Anhand der Anzahl der festgeschriebenen Datensätze kann ermittelt werden, welche Datensätze erfolgreich in die Datenbank eingefügt wurden. Die Anzahl der festgeschriebenen Datensätze kann nur ungleich null sein, wenn die Option `COMMITCOUNT` angegeben wurde.

Wird für **WARNINGCOUNT** im Befehl IMPORT explizit ein anderer Wert angegeben und wurden einige Zeilen zurückgewiesen, ist die zusammenfassende Zeilenausgabe des Dienstprogramms möglicherweise nicht korrekt. Dies liegt an einer Kombination aus der mit gepufferten INSERT-Operationen verwendeten asynchronen Fehlermeldung und der Tatsache, dass ein Fehler, der während des Einfügens einer Gruppe von Zeilen festgestellt wird, dazu führt, dass alle Zeilen der betreffenden Gruppe zurückgesetzt werden. Da das Dienstprogramm nicht zuverlässig zurückmeldet, welche Eingabedatensätze zurückgewiesen wurden, ist es schwierig festzustellen, welche Datensätze festgeschrieben wurden und welche Datensätze erneut in die Datenbank eingefügt werden müssen.

Verwenden Sie das DB2-Dienstprogramm BIND, um gepufferte INSERT-Operationen anzufordern. Das IMPORT-Paket `db2uimpm.bnd` muss mit der Option `INSERT BUF` erneut an die Datenbank gebunden werden. Zum Beispiel:

```
db2 connect to datenbankname
db2 bind db2uimpm.bnd insert buf
```

Die Funktion der gepufferten INSERT-Operationen kann nicht in Verbindung mit Importoperationen verwendet werden, bei denen der Modus `INSERT_UPDATE` angegeben ist. Diese Einschränkung wird von der Bindedatei `db2uImpInsUpdate.bnd` erzwungen. Diese Datei darf auf keinen Fall mit der Option `INSERT BUF` gebunden werden. Andernfalls schlagen Importoperationen im Modus `INSERT_UPDATE` fehl. Importoperationen im Modus `INSERT`, `REPLACE` oder `REPLACE_CREATE` werden durch das Binden der neuen Datei nicht beeinträchtigt.

Aspekte des Identitätsspaltenimports

Mit dem Dienstprogramm IMPORT können Sie Daten in eine Tabelle importieren, die eine Identitätsspalte enthält. Dabei spielt es keine Rolle, ob die Eingabedaten Identitätsspaltenwerte enthalten.

Sofern keine identitätsbezogenen Änderungswerte für den Dateityp verwendet werden, liegen der Ausführung des Dienstprogramms die folgenden Regeln zugrunde:

- Ist die Identitätsspalte als GENERATED ALWAYS definiert, wird für eine Tabellenzeile immer dann ein Identitätswert generiert, wenn die entsprechende Zeile in der Eingabedatei keinen Wert für die Identitätsspalte enthält oder explizit

einen Nullwert vorgibt. Ist für die Identitätsspalte ein Wert angegeben, der kein Nullwert ist, wird die Zeile zurückgewiesen (SQL3550W).

- Ist die Identitätsspalte als GENERATED BY DEFAULT definiert, verwendet das Dienstprogramm IMPORT benutzerdefinierte Werte, sofern diese vorhanden sind. Wenn diese Daten fehlen oder explizit Nullwerte sind, wird ein Wert generiert.

Das Dienstprogramm IMPORT führt keine zusätzliche Gültigkeitsprüfung der benutzerdefinierten Identitätswerte durch, die über die übliche Gültigkeitsprüfung für Werte mit dem Datentyp der Identitätsspalte (also SMALLINT, INT, BIGINT oder DECIMAL) hinausgeht. Falls Werte doppelt vorhanden sind, wird keine entsprechende Meldung ausgegeben. Außerdem kann der Änderungswert `compound=x` beim Importieren von Daten in eine Tabelle mit einer Identitätsspalte nicht verwendet werden.

Sie können den Import in Tabellen mit Identitätsspalte mit dem Änderungswert `identitymissing` für den Dateityp und mit dem Änderungswert `identityignore` für den Dateityp vereinfachen.

Import von Daten ohne Identitätsspalte
Der Änderungswert `identitymissing` vereinfacht das Importieren einer Tabelle mit einer Identitätsspalte, wenn die Eingabedatendatei für die Identitätsspalte keine Werte (auch keine Nullwerte) enthält. Angenommen, es wurde beispielsweise eine Tabelle mit der folgenden SQL-Anweisung definiert:

```
create table table1 (c1 char(30),
                     c2 int generated by default as identity,
                     c3 real,
                     c4 char(1))
```

Angenommen, ein Benutzer möchte Daten aus einer Datei (`import.del`) in die Tabelle TABLE1 importieren, und diese Daten wurden möglicherweise aus einer Tabelle exportiert, die keine Identitätsspalte enthält. Es folgt ein Beispiel für eine solche Datei.

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

Eine Methode für das Importieren dieser Datei wäre das explizite Auflisten der zu importierenden Spalten durch den folgenden Befehl **IMPORT**:

```
db2 import from import.del of del replace into table1 (c1, c3, c4)
```

Bei einer Tabelle mit vielen Spalten ist die Verwendung dieser Syntax jedoch eventuell umständlich und fehlerträchtig. Eine alternative Methode für das Importieren der Datei ist die folgende Verwendung des Änderungswerts für den Dateityp `identitymissing`:

```
db2 import from import.del of del modified by identitymissing
   replace into table1
```

Import von Daten mit Identitätsspalte
Der Änderungswert `identityignore` stellt in gewisser Hinsicht das Gegenteil des Änderungswerts `identitymissing` dar: Er weist das Dienstprogramm IMPORT an, dass die in der Eingabedatendatei vorhandenen Werte für die Identitätsspalte ignoriert werden sollen und dass für jede Zeile ein Identitätswert generiert werden soll. Beispiel: Ein Benutzer möchte die folgenden Daten aus einer Datei (`import.del`) in die wie zuvor beschrieben definierte Tabelle TABLE1 importieren:

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

Wenn die benutzerdefinierten Werte 1, 2 und 3 nicht für die Identitätsspalte verwendet werden sollen, könnte der Benutzer den folgenden Befehl **IMPORT** absetzen:

```
db2 import from import.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
```

Aber auch diese Methode kann möglicherweise umständlich und fehlerträchtig sein, wenn die Tabelle zu viele Spalten enthält. Der Änderungswert `identityignore` vereinfacht die Syntax folgendermaßen:

```
db2 import from import.del of del modified by identityignore
   replace into table1
```

Beim Exportieren einer Tabelle mit einer Identitätsspalte in eine IXF-Datei können die Optionen `REPLACE_CREATE` und `CREATE` des Befehls **IMPORT** verwendet werden, um die Tabelle, einschließlich ihrer Identitätsspalteneigenschaften, erneut zu erstellen. Wird eine IXF-Datei aus einer Tabelle mit einer als GENERATED ALWAYS definierten Identitätsspalte erstellt, kann die Datendatei nur dann erfolgreich importiert werden, wenn der Änderungswert `identityignore` angegeben wird. Andernfalls werden alle Zeilen zurückgewiesen (SQL3550W).

Anmerkung: Die Optionen `CREATE` und `REPLACE_CREATE` des Befehls IMPORT werden nicht weiter unterstützt und in zukünftigen Releases möglicherweise entfernt.

Aspekte des Imports von generierten Spalten

Mit dem Dienstprogramm IMPORT können Sie Daten in eine Tabelle mit generierten Spalten (Tabellen mit Identitätsspalte ausgenommen) importieren. Dabei spielt es keine Rolle, ob die Eingabedaten Werte für generierte Spalten enthalten.

Sofern keine Änderungswerte für den Dateityp verwendet werden, die sich auf generierte Spalten beziehen, liegen der Ausführung des Dienstprogramms die folgenden Regeln zugrunde:

- Für eine generierte Spalte wird immer dann ein Wert generiert, wenn die entsprechende Zeile in der Eingabedatei keinen Wert für die Spalte enthält oder einen Nullwert explizit vorgibt. Ist für eine generierte Spalte ein Wert angegeben, der kein Nullwert ist, wird die Zeile zurückgewiesen (SQL3550W).
- Generiert der Server einen Nullwert für eine generierte Spalte, die keinen Nullwert enthalten darf, wird die Datenzeile, zu der dieses Feld gehört, zurückgewiesen (SQL0407W). Dies könnte beispielsweise dann eintreten, wenn eine generierte Spalte, die keinen Nullwert enthalten darf, als Summe zweier Tabellenspalten definiert wurde und für diese Spalten in der Eingabedatei Nullwerte zur Verfügung gestellt wurden.

Sie können den Import von Daten in Tabellen mit generierten Spalten mit dem Änderungswert `identitymissing` für den Dateityp und mit dem Änderungswert `identityignore` für den Dateityp vereinfachen.

Import von Daten ohne generierte Spalten

Der Änderungswert `generatedmissing` vereinfacht das Importieren einer Tabelle mit generierten Spalten, wenn die Eingabedatendatei für alle in der Tabelle vorhandenen generierten Spalten keine Werte (auch keine Nullwerte) enthält. Angenommen, es wurde beispielsweise eine Tabelle mit der folgenden SQL-Anweisung definiert:

```
create table table1 (c1 int,
                                          c2 int,
                                          g1 int generated always as (c1 + c2),
                                          g2 int generated always as (2 * c1),
                                          c3 char(1))
```

Angenommen, ein Benutzer möchte Daten aus einer Datei (load.del) in die Tabelle TABLE1 importieren, und diese Daten wurden möglicherweise aus einer Tabelle exportiert, die keine generierten Spalten enthält. Es folgt ein Beispiel für eine solche Datei.

```
1, 5, J
2, 6, K
3, 7, I
```

Eine Methode für das Importieren dieser Datei wäre das explizite Auflisten der zu importierenden Spalten über den Befehl **IMPORT**:

```
db2 import from import.del of del replace into table1 (c1, c2, c3)
```

Bei einer Tabelle mit vielen Spalten ist die Verwendung dieser Syntax jedoch eventuell umständlich und fehlerträchtig. Eine alternative Methode für das Importieren der Datei ist die folgende Verwendung des Änderungswerts für den Dateityp generatedmissing:

```
db2 import from import.del of del modified by generatedmissing
   replace into table1
```

Import von Daten mit generierten Spalten
Der Änderungswert generatedignore stellt in gewisser Hinsicht das Gegenteil des Änderungswerts generatedmissing dar: Er weist das Dienstprogramm IMPORT an, dass die in der Eingabedatendatei vorhandenen Werte für alle generierten Spalten ignoriert und für jede Zeile Werte generiert werden sollen. Beispiel: Ein Benutzer möchte die folgenden Daten aus einer Datei (import.del) in die wie zuvor beschrieben definierte Tabelle TABLE1 importieren:

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

Die benutzerdefinierten Werte 10, 11, und 12 (für g1) sowie 15, 16 und 17 (für g2), die keine Nullwerte sind, bewirken, dass die Zeile zurückgewiesen wird (SQL3550W). Um dies zu verhindern, könnte der Benutzer den folgenden Befehl **IMPORT** absetzen:

```
db2 import from import.del of del method P(1, 2, 5)
   replace into table1 (c1, c2, c3)
```

Aber auch diese Methode kann möglicherweise umständlich und fehlerträchtig sein, wenn die Tabelle zu viele Spalten enthält. Der Änderungswert generatedignore vereinfacht die Syntax folgendermaßen:

```
db2 import from import.del of del modified by generatedignore
   replace into table1
```

Wenn Sie die Klausel INSERT_UPDATE verwenden, die generierte Spalte auch ein Primärschlüssel ist und der Änderungswert generatedignore angegeben ist, wird der Änderungswert generatedignore vom Befehl **IMPORT** berücksichtigt. Der Befehl **IMPORT** ersetzt nicht den vom Benutzer angegebenen Wert für diese Spalte in der Klausel WHERE der Anweisung UPDATE.

Aspekte des LOB-Imports

Da die Größe von Spaltenwerten beim Dienstprogramm IMPORT auf 32 KB begrenzt ist, müssen beim Import von LOBs (Large Objects) besondere Aspekte beachtet werden.

Das Dienstprogramm IMPORT behandelt Daten in der Eingabedatei standardmäßig wie in Spalten zu ladende Daten. Sind in der Haupteingabedatendatei jedoch LOB-Daten gespeichert, ist der Grenzwert für die Datengröße von 32 KB relevant. Es empfiehlt sich deshalb, LOB-Daten separat von der Hauptdatendatei zu speichern und beim Import von LOBs den Änderungswert `lobsinfile` für den Dateityp anzugeben, um einen Verlust von Daten zu vermeiden.

Die Klausel LOBS FROM impliziert die Aktivierung des Änderungswerts `lobsinfile`. Die Klausel LOBS FROM überträgt die Liste der Pfade an das Dienstprogramm IMPORT, die beim Importieren der Daten nach LOB-Dateien durchsucht werden sollen. Wird die Option LOBS FROM nicht angegeben, wird davon ausgegangen, dass sich die zu importierenden LOB-Dateien in demselben Pfad befinden wie die Eingabedatei mit den relationalen Daten.

Angabe zur Speicherung der LOB-Daten

Mit einer LOB-Positionskennung (LOB Location Specifier - LLS) können beim Import von LOB-Informationen mehrere große Objekte in einer gemeinsamen Datei gespeichert werden. Ist `lobsinfile` angegeben, generiert und speichert das Dienstprogramm EXPORT die Kennung in der Ausgabedatei für die Exportoperation, sodass der Speicherort der LOB-Daten anhand der Kennung ermittelt werden kann. Werden Daten importiert, bei denen die Option "MODIFIED BY `lobsinfile`" angegeben ist, erwartet die Datenbank jeweils eine LOB-Positionskennung für die betreffenden LOB-Spalten. Falls für eine LOB-Spalte eine andere Angabe als die LOB-Positionskennung gefunden wird, führt dies bei der Datenbank zu einer Behandlung als LOB-Datei, und die Datenbank lädt die gesamte Datei als großes Objekt.

Bei einem Import im Modus `CREATE` können Sie mit der Klausel LONG IN angeben, dass die LOB-Daten in einem separaten Tabellenbereich erstellt und gespeichert werden sollen.

Das folgende Beispiel veranschaulicht den Import einer DEL-Datei, bei der die zugehörigen LOBs in separaten Dateien gespeichert sind:

```
IMPORT FROM inputfile.del OF DEL
 LOBS FROM /tmp/data
 MODIFIED BY lobsinfile
 INSERT INTO newtable
```

Aspekte des Imports benutzerdefinierter einzigartiger Typen

Das Dienstprogramm IMPORT setzt die benutzerdefinierten einzigartigen Datentypen (UDTs) automatisch in ähnliche Basisdatentypen um. Dies erspart Ihnen das explizite Umsetzen der UDTs in die Basisdatentypen. Durch die Umsetzung werden Vergleiche zwischen UDTs und den Basisdatentypen in SQL ermöglicht.

Weitere Aspekte des Imports

Client/Server-Umgebungen und Import

Wenn Sie eine Datei in eine ferne Datenbank importieren, kann eine gespeicherte Prozedur aufgerufen werden, um den Import auf dem Server auszuführen.

In folgenden Situationen ist das Aufrufen einer gespeicherten Prozedur nicht möglich:

- Die Anwendung und die Datenbank verwenden unterschiedliche Codepages.
- Die Datei, die importiert wird, ist eine aus mehreren Teilen bestehende PC/IXF-Datei.
- Die Methode, die zum Importieren der Daten verwendet werden soll, beruht entweder auf dem Spaltennamen oder der relativen Spaltenposition.
- Die bereitgestellte Zielspaltenliste ist länger als 4 KB.
- Die Klausel LOBS FROM oder der Änderungswert `lobsinfile` ist angegeben.
- Die Klausel NULL INDICATORS ist für ASC-Dateien angegeben.

Beim Importieren mithilfe einer gespeicherten Prozedur werden in der Nachrichtendatei Nachrichten in der auf dem Server installierten Standardsprache erstellt. Die Nachrichten liegen in der Sprache der Anwendung vor, wenn Client und Server dieselbe Sprache benutzen.

Das Dienstprogramm IMPORT erstellt zwei temporäre Dateien im Unterverzeichnis `tmp` des Verzeichnisses `sqllib` (oder des Verzeichnisses, das durch die Registrierdatenbankvariable **DB2INSTPROF** angegeben wird). Eine Datei ist für die Daten vorgesehen, die andere Datei für die Nachrichten, die das Dienstprogramm IMPORT generiert.

Wenn beim Schreiben oder Öffnen von Daten auf dem Server ein Fehler gemeldet wird, prüfen Sie, ob folgende Bedingungen erfüllt sind:

- Das Verzeichnis existiert.
- Der Plattenspeicherplatz für die Dateien ist groß genug.
- Der Instanzeigner hat Schreibzugriff auf das Verzeichnis.

Vom Dienstprogramm IMPORT unterstützte Tabellensperrmodi

Das Dienstprogramm IMPORT unterstützt bei Tabellensperren den Offlinemodus (ALLOW NO ACCESS) und den Onlinemodus ALLOW WRITE ACCESS.

Der Modus `ALLOW NO ACCESS` verhindert den Zugriff auf Tabellendaten durch gleichzeitig ablaufende Anwendungen. Der Modus `ALLOW WRITE ACCESS` ermöglicht gleichzeitig ablaufenden Anwendungen sowohl Lese- als auch Schreibzugriffe auf die Zieltabelle der Importoperation. Ist kein Modus explizit angegeben, wird der Standardmodus `ALLOW NO ACCESS` beim Import verwendet. Darüber hinaus ist das Dienstprogramm IMPORT standardmäßig an die Datenbank mit der Isolationsstufe RS (Read Stability - Lesestabilität) gebunden.

Offlineimport (ALLOW NO ACCESS)

Im Modus `ALLOW NO ACCESS` fordert die Importoperation vor dem Einfügen von Zeilen eine exklusive Sperre (X) für die Zieltabelle an. Das Halten einer Sperre für eine Tabelle hat folgende Auswirkungen:

- Erstens wartet das Dienstprogramm IMPORT, wenn andere Anwendungen eine Tabellensperre halten oder für die Zieltabelle der Importoperation Zeilensperren vorliegen, bis diese Anwendungen ihre Änderungen mit COMMIT festschreiben oder mit ROLLBACK rückgängig machen.
- Zweitens warten während der Ausführung der Importoperation alle anderen Anwendungen, die eine Sperre anfordern, bis die Importoperation vollständig ausgeführt wurde.

Anmerkung: Sie können ein Zeitlimit für Sperren angeben, damit die Anwendungen (das Dienstprogramm IMPORT eingeschlossen) nicht unbegrenzt auf Sperren warten.
Durch das Anfordern einer exklusiven Sperre zu Beginn der Operation verhindert das Dienstprogramm IMPORT, dass Deadlocks durch andere Anwendungen, die mit derselben Zieltabelle arbeiten und Zeilensperren halten, entstehen.

Onlineimport (ALLOW WRITE ACCESS)

Im Modus `ALLOW WRITE ACCESS` fordert das Dienstprogramm IMPORT eine nicht exklusive Sperre (IX) für die Zieltabelle an. Das Halten dieser Sperre für die Tabelle hat folgende Auswirkungen:

- Wenn andere Anwendungen eine inkompatible Tabellensperre halten, beginnt das Dienstprogramm IMPORT erst dann mit dem Einfügen von Daten, wenn alle betreffenden Anwendungen ihre Änderungen mit COMMIT festgeschrieben oder mit ROLLBACK zurückgesetzt haben.
- Während der Ausführung der Importoperation warten alle anderen Anwendungen, die eine inkompatible Tabellensperre anfordern, bis die Importoperation die aktuelle Transaktion mit COMMIT festschreibt oder mit ROLLBACK zurücksetzt. Hierbei ist zu beachten, dass die Tabellensperre von IMPORT nicht über Transaktionsgrenzen hinaus bestehen bleibt. Aus diesem Grund muss beim Onlineimport nach jedem Commit eine Tabellensperre angefordert und unter Umständen auf diese gewartet werden.
- Wenn andere Anwendungen eine inkompatible Zeilensperre halten, stoppt das Dienstprogramm IMPORT das Einfügen von Daten so lange, bis alle betreffenden Anwendungen ihre Änderungen mit COMMIT festgeschrieben oder mit ROLLBACK zurückgesetzt haben.
- Während der Ausführung der Importoperation warten alle anderen Anwendungen, die eine inkompatible Zeilensperre anfordern, bis die Importoperation die aktuelle Transaktion mit COMMIT festschreibt oder mit ROLLBACK zurücksetzt.

Um die Online-Eigenschaften beizubehalten und das Risiko eines Deadlocks zu reduzieren, führt die Importoperation im Modus `ALLOW WRITE ACCESS` in regelmäßigen Abständen ein Commit für die aktuelle Transaktion durch und gibt alle Zeilensperren frei, bevor eine Eskalation zu einer exklusiven Tabellensperre (X) erfolgt. Wenn Sie die Häufigkeit der Commitoperationen nicht explizit definiert haben, werden die Commitoperationen beim Import so oft durchgeführt, wie es bei Angabe von `COMMITCOUNT AUTOMATIC` der Fall wäre. Es werden keine Commitoperationen durchgeführt, wenn die Commitzählung `COMMITCOUNT` auf `0` gesetzt ist.

Der Modus `ALLOW WRITE ACCESS` ist mit folgenden Importvarianten nicht kompatibel:

- Import im Modus `REPLACE`, `CREATE` oder `REPLACE_CREATE`
- Import mit gepufferten INSERT-Operationen
- Import in eine Zielsicht
- Import in eine Hierarchietabelle
- Import in eine Tabelle, bei der die Sperrgranularität auf Tabellenebene (durch Verwendung des Parameters LOCKSIZE in der Anweisung ALTER TABLE) definiert ist

Dienstprogramm LOAD

LOAD - Übersicht

Mit dem Dienstprogramm LOAD können große Datenmengen effizient in neu erstellte Tabellen oder in Tabellen, die bereits Daten enthalten, versetzt werden.

Es können die meisten Datentypen verarbeitet werden, einschließlich XML, großer Objekte (LOBs) und benutzerdefinierter Typen (UDTs).

Das Dienstprogramm LOAD arbeitet schneller als das Dienstprogramm IMPORT, da es formatierte Seiten direkt in die Datenbank schreibt, während das Dienstprogramm IMPORT die Daten mit SQL-Anweisungen INSERT einfügt.

Das Dienstprogramm LOAD startet keine Trigger und führt außer der Prüfung auf eindeutige Indizes keine Prüfungen auf referenzielle Integrität oder Integritätsbedingungen in Tabellen aus.

Der LOAD-Prozess umfasst vier Einzelphasen (siehe Abb. 3):

1. **LOAD (Laden)**
   Während der LOAD-Phase werden die Daten in die Tabelle geladen. Bei Bedarf können Indexschlüssel und Tabellenstatistiken erfasst werden. *Sicherungspunkte* oder Konsistenzpunkte werden in Intervallen eingerichtet, die Sie mit dem Parameter **SAVECOUNT** des Befehls **LOAD** angegeben haben. Es werden Nachrichten generiert, durch die Sie erfahren, wie viele Eingabezeilen zum Zeitpunkt des Sicherungspunkts erfolgreich geladen wurden.
2. **BUILD**
   Während der BUILD-Phase werden auf der Grundlage der während der LOAD-Phase erfassten Indexschlüssel Indizes erstellt. Die Indexschlüssel werden während der LOAD-Phase sortiert, und Indexstatistiken werden erfasst (wenn die Option **STATISTICS USE PROFILE** angegeben wurde und das Profil das Erfassen von Indexstatistiken angibt). Die erfassten Statistikdaten sind denen des Befehls **RUNSTATS** ähnlich.
3. **DELETE (Löschen)**
   Während der DELETE-Phase werden die Zeilen, die zu Verstößen gegen eindeutige Schlüssel oder Primärschlüssel geführt haben, aus der Tabelle entfernt. Diese gelöschten Zeilen werden in der LOAD-Ausnahmetabelle gespeichert, sofern diese Tabelle angegeben wurde.
4. **INDEX COPY (Indexkopie)**
   Während der INDEX COPY-Phase werden die Indexdaten aus einem Tabellenbereich für temporäre Systemtabellen in den ursprünglichen Tabellenbereich kopiert. Dieser Vorgang findet nur dann statt, wenn bei der Indexerstellung während einer Ladeoperation, bei der die Option **READ ACCESS** definiert war, ein Tabellenbereich für temporäre Systemtabellen angegeben wurde.

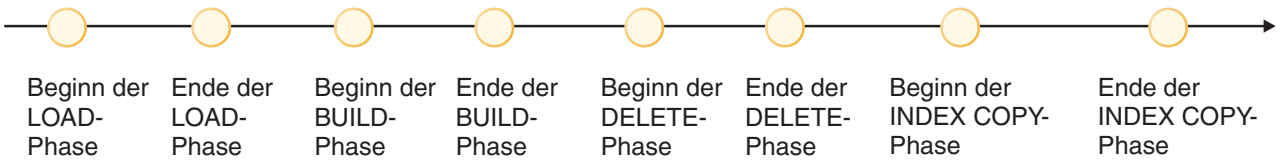


Abbildung 3. Die vier Phasen des LOAD-Prozesses: LOAD, BUILD, DELETE und INDEX COPY

Anmerkung: Nachdem das Dienstprogramm LOAD aufgerufen wurde, können Sie den Verarbeitungsfortschritt der Ladeoperation mithilfe des Befehls **LIST UTILITIES** überwachen.

Folgende Informationen werden benötigt, wenn Daten geladen werden:

- Der Pfad und der Name der Eingabedatei, benannten Pipe oder Einheit.
- Der Name oder Aliasname der Zieltabelle.
- Das Format der Eingabequelle. Mögliche Formate sind: DEL, ASC, PC/IXF oder CURSOR.
- Angabe, ob die Eingabedaten an die Tabelle angefügt werden oder die vorhandenen Daten in der Tabelle ersetzen sollen.
- Ein Nachrichtendateiname, wenn das Dienstprogramm über die Anwendungsprogrammierschnittstelle (API) db2Load aufgerufen wird.

Lademodi

- **Einfügung (INSERT)**
  In diesem Modus werden die Eingabedaten beim Laden an die Tabelle angefügt, ohne dass an den bereits vorhandenen Daten Änderungen vorgenommen werden.
- **Ersetzung (REPLACE)**
  In diesem Modus werden die bereits vorhandenen Daten beim Laden aus der Tabelle gelöscht und die Tabelle wird mit den Eingabedaten gefüllt.
- **Neustart (RESTART)**
  In diesem Modus wird ein unterbrochener Ladevorgang wieder aufgenommen. In den meisten Fällen wird der Ladevorgang mit der Phase fortgesetzt, in der zuvor ein Fehler aufgetreten ist. Wenn es sich dabei um die LOAD-Phase gehandelt hat, wird der Ladevorgang beim letzten erfolgreich überprüften Konsistenzpunkt wieder aufgenommen.
- **Beendigung (TERMINATE)**
  In diesem Modus wird eine fehlgeschlagene Ladeoperation mit einem Rollback zurückgesetzt.

Sie können die folgenden Optionen angeben:

- Die zu ladenden Daten befinden sich auf dem Client - sofern das Dienstprogramm LOAD von einem Client mit Fernverbindung aufgerufen wird. Beachten Sie, dass XML- und LOB-Daten immer aus dem Server gelesen werden, auch wenn die Option **CLIENT** angegeben wurde.
- Die Methode, die zum Laden der Daten verwendet werden soll: Spaltenposition, Spaltenname oder relative Spaltenposition.
- Die Häufigkeit, mit der das Dienstprogramm Konsistenzpunkte definieren soll.
- Die Namen der Tabellenspalten, in die die Daten eingefügt werden sollen.
- Die Angabe, ob die in der Tabelle bereits vorhandenen Daten abgefragt werden können oder nicht, während die Ladeoperation läuft.
- Die Angabe, ob die Ladeoperation entweder warten soll, bis andere Dienstprogramme oder Anwendungen die Verwendung der Tabelle beendet haben, oder vor der Fortsetzung das Beenden der anderen Anwendungen erzwingen soll.
- Einen alternativen Tabellenbereich für temporäre Tabellen, in dem der Index erstellt werden soll.
- Die Pfade und die Namen der Eingabedateien, in denen LOBs gespeichert sind.

  Anmerkung: Die LOB-Option **COMPACT** wird vom Dienstprogramm LOAD nicht unterstützt.

- Einen Nachrichtendateinamen. Bei Ladeoperationen können Sie angeben, dass Nachrichtendateien erstellt werden sollen, in denen Fehlernachrichten, Warnungen und informative Nachrichten zu diesen Operationen gespeichert werden. Geben Sie den Namen dieser Dateien mit dem Parameter MESSAGES an.

  Anmerkung:

  1. Sie können den Inhalt einer Nachrichtendatei erst anzeigen, wenn die Operation beendet ist. Wenn Sie die Ladenachrichten anzeigen möchten, während eine Ladeoperation ausgeführt wird, können Sie den Befehl **LOAD QUERY** verwenden.
  2. Jede Nachricht in einer Nachrichtendatei beginnt in einer neuen Zeile und enthält Informationen, die von der DB2-Funktion zum Abrufen von Nachrichten bereitgestellt werden.
- Angabe, ob Spaltenwerte, die geladen werden, ein impliziertes Dezimalzeichen haben oder nicht.
- Angabe, ob das Dienstprogramm die Menge des freien Speicherbereichs nach dem Laden einer Tabelle ändern soll.
- Angabe, ob während des LOAD-Prozesses Statistikdaten zusammengestellt werden. Diese Option wird nur unterstützt, wenn die Ladeoperation im Ersetzungsmodus (REPLACE) ausgeführt wird. Statistiken werden gemäß dem für die Tabelle definierten Profil erfasst. Das Profil muss vom Befehl **RUNSTATS** erstellt werden, bevor der Befehl **LOAD** ausgeführt wird. Ist das Profil nicht vorhanden und die Ladeoperation wird aufgefordert, Statistiken gemäß dem Profil zu erfassen, schlägt die Ladeoperation fehl und es wird eine Fehlernachricht zurückgegeben.

  Wenn Daten an eine Tabelle angefügt werden, werden keine Statistikdaten erfasst. Rufen Sie zum Erfassen von aktuellen Statistikdaten zu einer angefügten Tabelle nach Fertigstellung des Ladeprozesses das Dienstprogramm **RUNSTATS** auf. Wenn Statistikdaten zu einer Tabelle mit einem eindeutigen Index zusammengestellt und während der DELETE-Phase doppelte Schlüssel gelöscht werden, erfolgt keine Aktualisierung der Statistikdaten zur Berücksichtigung der gelöschten Datensätze. Wenn Sie eine signifikante Anzahl von doppelten Datensätzen erwarten, dürfen Sie während der Ladeoperation keine Statistikdaten erfassen. Rufen Sie stattdessen nach Fertigstellung des Ladeprozesses das Dienstprogramm **RUNSTATS** auf.
- Angabe, ob eine Kopie der vorgenommenen Änderungen erhalten bleiben soll. Diese wird benötigt, um eine aktualisierende Recovery der Datenbank zu ermöglichen. Diese Option wird nicht unterstützt, wenn für die Datenbank die aktualisierende Recovery inaktiviert ist, d.h. wenn die Datenbankkonfigurationsparameter **logarchmeth1** und **logarchmeth2** auf OFF gesetzt sind. Wenn keine Kopie erstellt wird und die aktualisierende Recovery aktiviert ist, verbleibt der Tabellenbereich nach Fertigstellung der Ladeoperation im Status "Backup anstehend".

  Für vollständig wiederherstellbare Datenbanken ist eine Protokollierung erforderlich. Das Dienstprogramm LOAD schließt die mit dem Laden von Daten verbundene Protokollierung fast völlig aus. Anstelle der Protokollierung haben Sie die Möglichkeit, eine Kopie des geladenen Tabellenteils zu erstellen. Bei einer Datenbankumgebung, die die Datenbankrecovery nach einem Fehler zulässt, haben Sie folgende Möglichkeiten:

  – Explizites Anfordern der Erstellung einer Kopie des geladenen Tabellenteils
  – Nach Fertigstellung der Ladeoperation Erstellen eines Backups der Tabellenbereiche, in denen sich die Tabelle befindet

  Wird der Datenbankkonfigurationsparameter **logindexbuild** gesetzt und wird die Ladeoperation mit der Wiederherstellbarkeitsoption **COPY YES** und der Indexierungsoption INCREMENTAL aufgerufen, dann werden beim Laden alle Indexän-

derungen protokolliert. Der Vorteil bei der Verwendung dieser Optionen liegt darin, dass bei der aktualisierenden Recovery der Protokollsätze für diese Ladeoperation auch die Indizes wiederhergestellt werden, was normalerweise nur der Fall wäre, wenn beim Laden der Indexierungsmodus REBUILD verwendet worden wäre.

Wenn Sie eine Tabelle laden, die bereits Daten enthält, und die Datenbank nicht wiederherstellbar ist, müssen Sie sicherstellen, dass Sie über eine Backup-Kopie der Datenbank oder über die Tabellenbereiche für die geladene Tabelle verfügen, bevor Sie das Dienstprogramm LOAD aufrufen, damit Sie Fehler beheben können.

Wenn Sie eine Folge von mehreren Ladeoperationen für eine wiederherstellbare Datenbank ausführen möchten, wird die Operationsfolge schneller durchlaufen, wenn Sie jede Ladeoperation als nicht wiederherstellbar definieren und am Ende der Ladefolge ein Backup erstellen, als wenn Sie jede der Ladeoperationen mit der Option **COPY YES** aufrufen. Sie können die Option NONRECOVERABLE verwenden, um festzulegen, dass eine Ladetransaktion als nicht wiederherstellbar gekennzeichnet werden soll und dass es nicht möglich sein wird, sie durch eine nachfolgende Operation zur aktualisierenden Recovery wiederherzustellen. Das Dienstprogramm zur aktualisierenden Recovery überspringt die Transaktion und kennzeichnet die Tabelle, in die die Daten geladen wurden, als "ungültig". Vom Dienstprogramm werden auch alle nachfolgenden Transaktionen für diese Tabelle ignoriert. Nach Fertigstellung der aktualisierenden Recovery kann eine solche Tabelle nur gelöscht werden (siehe Abb. 4). Mit dieser Option werden Tabellenbereiche nach der Ladeoperation nicht in den Status "Backup anstehend" gesetzt, und während der Ladeoperation muss keine Kopie der geladenen Daten erstellt werden.

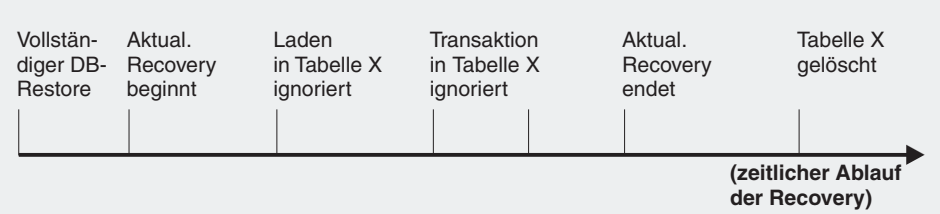


Abbildung 4. Nicht wiederherstellbare Verarbeitung bei einer aktualisierenden Recovery

- Den vollständig qualifizierten Pfad, der beim Erstellen von temporären Dateien während einer Ladeoperation verwendet werden soll. Der Name wird mit dem Parameter **TEMPFILES PATH** des Befehls **LOAD** angegeben. Der Standardwert ist der Datenbankpfad. Der Pfad befindet sich auf der Servermaschine, und der Zugriff auf diesen Pfad erfolgt ausschließlich durch die DB2-Instanz. Daher muss jede Angabe eines Pfadnamens in diesem Parameter die Verzeichnisstruktur des Servers und nicht des Clients wiedergeben. Außerdem benötigt der DB2-Instanzeigner Lese- und Schreibberechtigung für den Pfad.

Für das Dienstprogramm LOAD erforderliche Zugriffsrechte und Berechtigungen

Zugriffsrechte ermöglichen es Benutzern, Datenbankressourcen zu erstellen oder auf diese zuzugreifen. Berechtigungsstufen stellen eine Methode dar, um Berechtigungen sowie übergeordnete Pflege- und Dienstprogrammoperationen des Datenbankmanagers zusammenzufassen. Sie dienen zusammen zur Steuerung des Zugriffs auf den Datenbankmanager und seine Datenbankobjekte. Benutzer können nur auf die Objekte zugreifen, für die sie zugriffsberechtigt sind, d. h., für die sie über das erforderliche Zugriffsrecht oder die erforderliche Berechtigung verfügen.

Sie benötigen eine der folgenden Berechtigungen, damit Sie Daten in eine Tabelle laden können:

- Berechtigung DATAACCESS
- Berechtigung LOAD oder DBADM für die Datenbank und
  - Zugriffsrecht INSERT für die Tabelle, wenn das Dienstprogramm LOAD im Einfügemodus (`INSERT`), im Beendigungsmodus (`TERMINATE`) zur Beendigung einer vorherigen LOAD INSERT-Operation oder im Neustartmodus (`RESTART`) zum erneuten Starten einer vorherigen LOAD INSERT-Operation aufgerufen wird.
  - Zugriffsrecht INSERT und DELETE für die Tabelle, wenn das Dienstprogramm LOAD im Ersetzungsmodus (`REPLACE`), im Beendigungsmodus (`TERMINATE`) zur Beendigung einer vorherigen LOAD REPLACE-Operation oder im Neustartmodus (`RESTART`) zum erneuten Starten einer vorherigen LOAD REPLACE-Operation aufgerufen wird.
  - Zugriffsrecht INSERT für die Ausnahmetabelle, wenn eine solche Tabelle im Rahmen der Ladeoperation verwendet wird.
  - Zugriffsrecht SELECT für SYSCAT.TABLES in einigen Fällen, in denen LOAD die Katalogtabellen abfragt.

Da der Instanzeigner auch Eigner aller LOAD-Prozesse (und im Allgemeinen auch aller DB2-Serverprozesse) ist und alle diese Prozesse die Kennung des Instanzeigners verwenden, um auf benötigte Dateien zuzugreifen, muss der Instanzeigner über Lesezugriff auf Eingabedatendateien verfügen. Diese Eingabedatendateien müssen für den Instanzeigner unabhängig davon, wer den Befehl aufruft, lesbar sein.

Bei Angabe der Option `REPLACE` muss die Berechtigungs-ID der Sitzung über die Berechtigung zum Löschen der Tabelle verfügen.

Bei den Betriebssystemen Windows und Windows.NET, unter denen DB2 als Windows-Dienst ausgeführt wird, müssen Sie, falls Sie Daten aus Dateien laden, die sich auf einem Netzlaufwerk befinden, den DB2-Dienst zur Ausführung unter einem Benutzereintrag konfigurieren, der über Lesezugriff auf diese Dateien verfügt.

Anmerkung:

- Um Daten in eine Tabelle laden zu können, die geschützte Spalten enthält, muss die Berechtigungs-ID der Sitzung über LBAC-Berechtigungsnachweise verfügen, die Schreibzugriff auf alle geschützten Spalten in der Tabelle zulassen.
- Um Daten in eine Tabelle laden zu können, die geschützte Zeilen enthält, muss der Berechtigungs-ID der Sitzung mit Grant ein Sicherheitskennsatz für Schreibzugriff erteilt werden, der Teil der Sicherheitsrichtlinie zum Schutz der Tabelle ist.

Berechtigung LOAD

Benutzer, die über die Berechtigung LOAD auf Datenbankebene sowie über das Zugriffsrecht INSERT für eine Tabelle verfügen, können den Befehl **LOAD** verwenden, um Daten in eine Tabelle zu laden.

Anmerkung: Benutzer mit der Berechtigung DATAACCESS verfügen über die volle Berechtigung für die Verwendung des Befehls LOAD.

Benutzer, die über die Berechtigung LOAD auf Datenbankebene sowie über das Zugriffsrecht INSERT auf eine Tabelle verfügen, können den Befehl **LOAD RESTART**

oder **LOAD TERMINATE** verwenden, wenn bei der vorangegangenen Ladeoperation Daten eingefügt (LOAD INSERT) wurden.

Benutzer, die über die Berechtigung LOAD auf Datenbankebene sowie über die Zugriffsrechte INSERT und DELETE für eine Tabelle verfügen, können den Befehl **LOAD REPLACE** verwenden.

Wenn bei der vorangegangenen Ladeoperation Daten ersetzt (LOAD REPLACE) wurden, muss diesem Benutzer auch das Zugriffsrecht DELETE erteilt werden, damit dieser den Befehl **LOAD RESTART** oder **LOAD TERMINATE** verwenden kann.

Wenn die Ausnahmetabellen im Rahmen der LOAD-Operation verwendet werden, muss der Benutzer über das Zugriffsrechte INSERT für die Ausnahmetabellen verfügen.

Der Benutzer mit dieser Berechtigung kann die Befehle **QUIESCE TABLESPACES FOR TABLE**, **RUNSTATS** und **LIST TABLESPACES** ausführen.

Laden von Daten

Mit dem Dienstprogramm LOAD können große Datenmengen effizient in neu erstellte Tabellen oder in Tabellen, die bereits Daten enthalten, versetzt werden.

Vorbereitende Schritte

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 die Anweisung COMMIT absetzt, sollten Sie vor dem Aufrufen des Dienstprogramms LOAD alle Transaktionen beenden und alle Sperren freigeben, indem Sie entweder eine Anweisung COMMIT oder eine Anweisung ROLLBACK absetzen.

Daten werden in der Reihenfolge geladen, in der sie in der Eingabedatei enthalten sind. Ausgenommen sind MDC-Tabellen (Multidimensional Clustering Table - mehrdimensionale Clusteringtabelle), partitionierte Tabellen und Ladeoperationen mit dem Änderungswert **anyorder** für den Dateityp. Falls Sie eine bestimmte Reihenfolge wünschen, müssen die Daten vor dem Durchführen der Ladeoperation sortiert werden. Wenn Cluster erforderlich sind, müssen die Daten vor dem Laden nach dem Clusterindex sortiert werden. Beim Laden von Daten in MDC-Tabellen ist vor der Ladeoperation keine Sortierung erforderlich, und die Daten werden gemäß der MDC-Tabellendefinition in Gruppen (Clustern) zusammengefasst. Beim Laden von Daten in partitionierte Tabellen ist vor der Ladeoperation keine Sortierung erforderlich, und die Daten werden gemäß der Tabellendefinition partitioniert.

Einschränkungen

Im Folgenden finden Sie einige der für das Dienstprogramm LOAD geltenden Einschränkungen:

- Das Laden von Daten in Kurznamen wird nicht unterstützt.
- Das Laden von Daten in typisierte Tabellen oder in Tabellen mit Spalten strukturierten Typs wird nicht unterstützt.
- Das Laden von Daten in deklarierte temporäre Tabellen und erstellte temporäre Tabellen wird nicht unterstützt.

- XML-Daten können nur serverseitig gelesen werden; wenn die XML-Dateien aus dem Client gelesen werden sollen, verwenden Sie das Dienstprogramm IMPORT.
- In einem Tabellenbereich, der sich im Status "Backup anstehend" befindet, können keine Tabellen erstellt oder gelöscht werden.
- Sie können keine Daten in eine Datenbank laden, auf die über DB2 Connect oder eine Version des Servers vor DB2 Version 2 zugegriffen wird. Optionen, die nur mit dem aktuellen Release verfügbar sind, können nicht mit einem Server aus einem früheren Release verwendet werden.
- Wenn während einer **LOAD REPLACE**-Operation ein Fehler auftritt, gehen die Originaldaten in der Tabelle verloren. Behalten Sie eine Kopie der Eingabedaten, damit die Ladeoperation erneut gestartet werden kann.
- Für neu geladene Zeilen werden keine Trigger aktiviert. Geschäftsregeln, die Triggern zugeordnet sind, werden vom Dienstprogramm LOAD nicht umgesetzt.
- Das Laden verschlüsselter Daten wird nicht unterstützt.

Im Folgenden finden Sie einige der Einschränkungen, die für das Dienstprogramm LOAD beim Laden in eine partitionierte Tabelle gelten:

- 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 oder einer Operation im Status "Festlegen der Integrität anstehend" verwendete Ausnahmetabelle kann nicht partitioniert werden.
- 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.

Vorgehensweise

Gehen Sie wie folgt vor, um das Dienstprogramm LOAD aufzurufen:

- Setzen Sie den Befehl **LOAD** im Befehlszeilenprozessor ab.
- Rufen Sie die Anwendungsprogrammierschnittstelle (API) db2Load über eine Clientanwendung auf.
- Öffnen Sie den Taskassistenten in IBM Data Studio für den Befehl **LOAD**.

Beispiel

Es folgt ein Beispiel für den Befehl **LOAD**, der über den CLP abgesetzt wird:

```
db2 load from stafftab.ixf of ixf messages staff.msgs
   insert into userid.staff copy yes use tsm data buffer 4000
```

In diesem Beispiel gilt:

- Alle Warnungen oder Fehlernachrichten werden in die Datei `staff.msgs` gestellt.
- Eine Kopie der vorgenommenen Änderungen wird in Tivoli Storage Manager (TSM) gespeichert.
- Bei der Ladeoperation sind viertausend Seiten Pufferspeicherbereich zu verwenden.

Es folgt ein weiteres Beispiel für den Befehl **LOAD**, der über den CLP abgesetzt wird:

```
db2 load from stafftab.ixf of ixf messages staff.msgs
   tempfiles path /u/myuser replace into staff
```

In diesem Beispiel gilt:

- Die Tabellendaten werden ersetzt.
- Der Parameter **TEMPFILES PATH** wird verwendet, um /u/myuser als Server-Pfad anzugeben, in den temporäre Dateien geschrieben werden.

Anmerkung: In diesen Beispielen werden relative Pfadnamen für die LOAD-Eingabedatei verwendet. Relative Pfadnamen sind nur für Aufrufe eines Clients auf derselben Datenbankpartition wie die Datenbank zulässig. Es empfiehlt sich, vollständig qualifizierte Pfadnamen zu verwenden.

Nächste Schritte

Nachdem das Dienstprogramm LOAD aufgerufen wurde, können Sie den Verarbeitungsfortschritt der Ladeoperation mithilfe des Befehls **LIST UTILITIES** überwachen. Wird eine Ladeoperation im Modus **INSERT**, **REPLACE** oder **RESTART** ausgeführt, steht Unterstützung für eine ausführliche Überwachung des Verarbeitungsfortschritts zur Verfügung. Setzen Sie den Befehl **LIST UTILITIES** mit dem Parameter **SHOW DETAILS** ab, um ausführliche Informationen zur aktuellen LOAD-Phase anzuzeigen. Für Ladeoperationen, die im Modus **TERMINATE** ausgeführt werden, stehen keine Details zur Verfügung. Der Befehl **LIST UTILITIES** zeigt in diesem Fall lediglich an, dass gerade ein Dienstprogramm LOAD im Modus TERMINATE ausgeführt wird.

Eine Ladeoperation erhält eindeutige Integritätsbedingungen, Bereichsvorgaben für partitionierte Tabellen, generierte Spalten sowie LBAC-Sicherheitsregeln. Bei allen anderen Integritätsbedingungen wird die Tabelle zu Beginn der Ladeoperation in den Status "Festlegen der Integrität anstehend" versetzt. Nachdem die Ladeoperation vollständig ausgeführt wurde, muss der Status "Festlegen der Integrität anstehend" für die Tabelle mithilfe der Anweisung SET INTEGRITY aufgehoben werden.

Laden von XML-Daten

Mit dem Dienstprogramm LOAD können große XML-Datenmengen effizient in Tabellen versetzt werden.

Beim Laden von Daten in eine XML-Tabellenspalte können Sie die Option XML FROM verwenden, um die Pfade der XML-Eingabedatei oder der XML-Eingabedateien anzugeben. Um beispielsweise Daten aus der Datei /home/user/xmlpath/xmlfile1.xml zu laden, könnten Sie den folgenden Befehl verwenden:

```
LOAD FROM data1.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

Die ASCII-Eingabedatei mit Begrenzern data1.del enthält eine XML-Datenkennung (XDS), die die Speicherposition der zu ladenden XML-Daten angibt. Die folgende XDS beispielsweise beschreibt ein XML-Dokument an der relativen Adresse (Offset) bei 123 Byte in der Datei xmldata.ext mit einer Länge von 456 Byte:

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' />
```

Das Laden von XML-Daten mithilfe eines deklarierten Cursors wird unterstützt. Im folgenden Beispiel wird ein Cursor deklariert und mit dem Befehl **LOAD** verwendet, um Daten aus der Tabelle CUSTOMERS in der Tabelle LEVEL1_CUSTOMERS hinzuzufügen:

```
DECLARE cursor_income_level1 CURSOR FOR
   SELECT * FROM customers
   WHERE XMLEXISTS('$DOC/customer[income_level=1]');

LOAD FROM cursor_income_level1 OF CURSOR INSERT INTO level1_customers;
```

Der Dateitypänderungswert ANYORDER des Befehls **LOAD** wird beim Laden von XML-Daten in eine XML-Spalte unterstützt.

Während des Ladens werden für Spalten des Typs XML keine Verteilungsstatistiken erfasst.

Laden von XML-Daten in einer Umgebung mit partitionierten Datenbanken

Bei Tabellen, die über Datenbankpartitionen verteilt sind, können Sie XML-Daten aus XML-Datendateien parallel in die Tabellen laden. Beim Laden von XML-Daten in Tabellen müssen die XML-Datendateien über Lesezugriff auf alle Datenbankpartitionen verfügen, in denen das Laden stattfindet.

Eingefügte Dokumente anhand von Schemata prüfen

Die Option XMLVALIDATE ermöglicht die Gültigkeitsprüfung von XML-Dokumenten mithilfe von XML-Schemata während des Ladens. Im folgenden Beispiel wird die Gültigkeit eingehender XML-Dokumente auf der Basis des Schemas überprüft, das durch die XDS in der ASCII-Eingabedatei mit Begrenzern data2.del identifiziert wird:

```
LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
     USING XDS INSERT INTO USER.T2
```

In diesem Fall enthält die XDS ein Attribut vom Typ SCH mit der vollständig qualifizierten SQL-Kennung des für die Gültigkeitsprüfung zu verwendenden XML-Schemas "S1.SCHEMA_A":

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' SCH='S1.SCHEMA_A' />
```

Optionen für das Parsing angeben

Mit der Option XMLPARSE können Sie angeben, ob Leerzeichen in den geladenen XML-Dokumenten beim Parsing (d. h. bei der syntaktischen Analyse) beibehalten oder gelöscht werden sollen. Im folgenden Beispiel wird die Gültigkeit aller geladenen XML-Dokumente auf der Basis des Schemas mit der SQL-Kennung "S2.SCHEMA_A" überprüft, und für diese Dokumente werden die Leerzeichen beim Parsing beibehalten:

```
LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
WHITESPACE XMLVALIDATE USING SCHEMA S2.SCHEMA_A INSERT INTO USER.T1
```

CLP-Beispiele für LOAD-Sitzungen

Beispiel 1

TABLE1 enthält 5 Spalten:

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 enthält 6 Elemente:

- ELE1 Positionen 01 bis 20
- ELE2 Positionen 21 bis 22
- ELE3 Positionen 23 bis 23
- ELE4 Positionen 24 bis 27
- ELE5 Positionen 28 bis 31
- ELE6 Positionen 32 bis 32
- ELE7 Positionen 33 bis 40

Datensätze:

```
   1...5...10...15...20...25...30...35...40
Testdaten 1         XXN 123abcdN
   Testdaten 2 und 3   QQY    XXN
   Testdaten 4,5 und 6 WWN6789    Y
```

Mit dem folgenden Befehl wird die Tabelle aus der Datei geladen:

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
      method L (1 20, 21 22, 24 27, 28 31)
      null indicators (0,0,23,32)
insert into table1 (col1, col5, col2, col3)
```

Anmerkung:

1. Die Angabe `striptblanks` im Parameter MODIFIED BY bewirkt, dass Leerzeichen in VARCHAR-Spalten (zum Beispiel COL1, die 11, 17 und 19 Byte lang ist, in den Zeilen 1, 2 bzw. 3) abgeschnitten werden.
2. Die Angabe `reclen=40` im Parameter MODIFIED BY legt fest, dass es am Ende der Eingabedatensätze kein Zeilenvorschubzeichen gibt und jeder Datensatz 40 Byte lang ist. Die letzten 8 Byte werden nicht zum Laden der Tabelle verwendet.
3. Da COL4 nicht in der Eingabedatei bereitgestellt wird, wird sie in TABLE1 mit dem Standardwert eingefügt (sie ist mit NOT NULL WITH DEFAULT definiert).
4. Die Positionen 23 und 32 werden verwendet, um anzugeben, ob in COL2 und COL3 von TABLE1 für eine gegebene Zeile NULL geladen wird. Wenn die Nullanzeigerposition der Spalte für einen gegebenen Satz `Y` enthält, ist die Spalte NULL. Enthält sie `N`, werden die Datenwerte in den Datenpositionen der Spalte des Eingabedatensatzes (wie in L(.....) definiert) als Quelle der Spaltendaten für die Zeile verwendet. In diesem Beispiel ist keine Spalte in Zeile 1 NULL. COL2 in Zeile 2 ist NULL, und COL3 in Zeile 3 ist NULL.
5. In diesem Beispiel werden die NULL-Anzeiger für COL1 und COL5 als 0 (Null) angegeben, wodurch festgelegt wird, dass die Daten keine Nullwerte enthalten dürfen.
6. Der NULL-Anzeiger für eine bestimmte Spalte kann sich an einer beliebigen Position im Eingabedatensatz befinden. Diese Position muss jedoch angegeben werden, und der Wert `Y` oder `N` muss ebenfalls angegeben werden.

Beispiel 2 (unter Verwendung von Speicherauszugsdateien)

Tabelle FRIENDS ist wie folgt definiert:

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

Wenn versucht wird, die folgenden Datensätze in diese Tabelle zu laden:

```
   23, 24, bobby
   , 45, john
4,, mary
```

wird die zweite Zeile zurückgewiesen, weil das erste INT NULL ist und die Spaltendefinition NOT NULL lautet. Spalten, die Anfangszeichen enthalten, die nicht mit dem DEL-Format konsistent sind, führen zu einem Fehler, und der Datensatz wird zurückgewiesen. Solche Datensätze können in eine Speicherauszugsdatei geschrieben werden.

DEL-Daten, die in einer Spalte außerhalb von Zeichenbegrenzern vorkommen, werden ignoriert, führen jedoch zur Generierung einer Warnung. Beispiel:

```
22,34,"bob"
24,55,"sam" sdf
```

Das Dienstprogramm lädt "sam" in die dritte Spalte der Tabelle, und die Zeichen "sdf" werden in einer Warnung markiert. Der Datensatz wird nicht zurückgewiesen. Anderes Beispiel:

```
22 3, 34,"bob"
```

Das Dienstprogramm lädt `22,34,"bob"` und generiert eine Warnung, dass einige Daten in Spalte eins nach `22` ignoriert wurden. Der Datensatz wird nicht zurückgewiesen.

Beispiel 3 (Laden einer Tabelle mit einer Identitätsspalte)

TABLE1 enthält 4 Spalten:

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 entspricht TABLE1, wobei jedoch C2 eine als GENERATED ALWAYS definierte Identitätsspalte ist.

Die Datensätze in DATAFILE1 (DEL-Format) lauten:

```
"Liszt"
"Holst",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

Die Datensätze in DATAFILE2 (DEL-Format) lauten:

```
"Liszt", 74.49, A
"Holst", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

Anmerkung:

1. Der folgende Befehl generiert Identitätswerte für die Zeilen 1 und 2, da in der Datei DATAFILE1 für diese Zeilen keine Identitätswerte zur Verfügung gestellt werden. Den Zeilen 3 und 4 wird jedoch der benutzerdefinierte Identitätswert 100 bzw. 101 zugeordnet.

   ```
   db2 load from datafile1.del of del replace into table1
   ```

2. Um die Datei DATAFILE1 so in die Tabelle TABLE1 zu laden, dass für alle Zeilen Identitätswerte generiert werden, setzen Sie einen der folgenden Befehle ab:

```
db2 load from datafile1.del of del method P(1, 3, 4)
         replace into table1 (c1, c3, c4)
   db2load from datafile1.del of del modified by identityignore
   replace into table1
```

3. Um die Datei DATAFILE2 so in die Tabelle TABLE1 zu laden, dass für alle Zeilen Identitätswerte generiert werden, setzen Sie einen der folgenden Befehle ab:

   ```
   db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
      db2 load from datafile2.del of del modified by identitymissing
      replace into table1
   ```

4. Um die Datei DATAFILE1 so in die Tabelle TABLE2 zu laden, dass den Zeilen 3 und 4 die Identitätswerte 100 bzw. 101 zugeordnet werden, setzen Sie den folgenden Befehl ab:

   ```
   db2 load from datafile1.del of del modified by identityoverride
      replace into table2
   ```

   In diesem Fall werden die Zeilen 1 und 2 zurückgewiesen, da das Dienstprogramm angewiesen wurde, die durch das System generierten Identitätswerte mit den benutzerdefinierten Werten zu überschreiben. Sind jedoch keine benutzerdefinierten Werte vorhanden, muss die Zeile zurückgewiesen werden, da Identitätsspalten implizit nicht NULL sind.

5. Wird die Datei DATAFILE1 in die Tabelle TABLE2 geladen, ohne dass einer der identitätsbezogenen Änderungswerte für den Dateityp verwendet wird, werden die Zeilen 1 und 2 geladen, die Zeilen 3 und 4 jedoch zurückgewiesen, da diese Zeilen eigene Werte, die nicht NULL sind, liefern und die Identitätsspalte als GENERATED ALWAYS definiert wurde.

Beispiel 3 (Laden über CURSOR)

Die Tabelle MY.TABLE1 enthält 3 Spalten:

- ONE INT
- TWO CHAR(10)
- THREE DATE

Die Tabelle MY.TABLE2 enthält 3 Spalten:

- ONE INT
- TWO CHAR(10)
- THREE DATE

Der Cursor MYCURSOR ist wie folgt definiert:

```
declare mycursor cursor for select * from my.table1
```

Mit dem folgenden Befehl werden alle Daten aus der Tabelle MY.TABLE1 in die Tabelle MY.TABLE2 geladen:

```
load from mycursor of cursor method P(1,2,3) insert into
    my.table2(one,two,three)
```

Anmerkung:

1. In einem einzelnen Befehl LOAD kann nur ein Cursorname angegeben werden. Dies bedeutet, dass die Angabe `load from mycurs1, mycurs2 of cursor...` nicht zulässig ist.
2. Die einzigen gültigen Werte für METHOD beim Laden über einen Cursor sind die Werte `P` und `N`.

3. Im vorstehenden Beispiel könnten die Angabe METHOD P und die Spaltenliste für das Einfügen (1,2,3) übergangen werden, weil sie Standardwerte darstellen.
4. MY.TABLE1 kann eine Tabelle, eine Sicht, ein Aliasname oder ein Kurzname sein.

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 der Veröffentlichung *Neuerungen in DB2 Version 10.1*.

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.

Aspekte des Ladens von LBAC-geschützten Daten

Die Voraussetzung für eine erfolgreichen Ladeoperation in eine Tabelle mit geschützten Zeilen sind LBAC-Berechtigungsnachweise (Label-Based Access Control). Darüber hinaus müssen Sie über einen gültigen Sicherheitskennsatz (oder einen Sicherheitskennsatz, der in einen gültigen Kennsatz konvertiert werden kann) für die Sicherheitsrichtlinie verfügen, die der Zieltabelle aktuell zugeordnet ist.

Wenn Sie nicht über gültige LBAC-Berechtigungsnachweise verfügen, schlägt das Laden fehl, und es wird ein Fehler (SQLSTATE 42512) zurückgegeben. Enthalten die Eingabedaten keinen Sicherheitskennsatz oder liegt der Sicherheitskennsatz nicht im zugehörigen internen Binärformat vor, können Sie die Ladeoperation mithilfe verschiedener Änderungswerte für den Dateityp fortsetzen.

Wenn Sie Daten in eine Tabelle mit geschützten Zeilen laden, enthält die Zieltabelle eine einzige Spalte mit dem Datentyp DB2SECURITYLABEL. Wenn die Eingabedatenzeile keinen Wert für diese Spalte enthält, wird die Zeile zurückgewiesen, es sei denn, der Ladebefehl wurde mit dem Änderungswert `usedefaults` für den Dateityp angegeben. Wurde `usedefaults` im Ladebefehl angegeben, wird der Sicher-

heitskennsatz, über den Sie für den Schreibzugriff verfügen, aus der Sicherheitsrichtlinie, die die Tabelle schützt, verwendet. Wenn Sie keinen Sicherheitskennsatz für den Schreibzugriff haben, wird die Zeile zurückgewiesen und mit der Verarbeitung der nächsten Zeile fortgefahren.

Wenn Sie Daten in eine Tabelle mit geschützten Zeilen laden und die Eingabedaten einen Wert für die Spalte mit dem Datentyp DB2SECURITYLABEL enthalten, gelten dieselben Regeln wie beim Einfügen von Daten in die betreffende Tabelle. Wenn der Sicherheitskennsatz, der die zu ladende Zeile schützt, also der Kennsatz in der betreffenden Zeile der Datendatei, ein Kennsatz ist, der Ihnen den Schreibzugriff ermöglicht, wird der betreffende Sicherheitskennsatz zum Schutz der Zeile verwendet. Dies bedeutet, dass der Sicherheitskennsatz in die Spalte mit dem Datentyp DB2SECURITYLABEL geschrieben wird. Wenn Sie nicht in eine Zeile schreiben können, die durch diesen Sicherheitskennsatz geschützt wird, hat dies, je nachdem wie die Sicherheitsrichtlinie zum Schutz der Quellentabelle erstellt wurde, unterschiedliche Konsequenzen:

- Wenn die Anweisung CREATE SECURITY POLICY, mit der die Richtlinie erstellt wurde, die Option `RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL` enthielt, wird die Zeile zurückgewiesen.
- Enthielt die Anweisung CREATE SECURITY POLICY diese Option nicht oder stattdessen die Option `OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL`, wird der Sicherheitskennsatz in der Datendatei für die betreffende Zeile ignoriert und stattdessen zum Schutz der Zeile der Sicherheitskennsatz verwendet, den Sie für den Schreibzugriff besitzen. In diesem Fall wird weder ein Fehler noch eine Warnung ausgegeben. Wenn Sie keinen Sicherheitskennsatz für den Schreibzugriff haben, wird die Zeile zurückgewiesen und mit der Verarbeitung der nächsten Zeile fortgefahren.

Hinweise zu Begrenzern

Wenn Sie Daten in eine Spalte des Typs DB2SECURITYLABEL laden, wird standardmäßig davon ausgegangen, dass es sich bei dem Wert in der Datendatei um die tatsächlichen Byte handelt, die die interne Darstellung dieses Sicherheitskennsatzes bilden. Einige Rohdaten enthalten jedoch möglicherweise Zeichen für Zeilenvorschub, die vom Befehl **LOAD** fälschlicherweise als Begrenzer für die Zeile interpretiert werden. Wenn dieses Problem vorliegt, können Sie mit dem Änderungswert `delprioritychar` für den Dateityp sicherstellen, dass der Zeichenbegrenzer Vorrang vor dem Zeilenbegrenzer hat. Wenn Sie `delprioritychar` verwenden, werden Zeilen- und Spaltenbegrenzer, die innerhalb der zwischen den Zeichenbegrenzern enthaltenen Daten vorliegen, nicht als Begrenzer erkannt. Auch wenn kein Wert ein Zeilenvorschubzeichen enthält, kann der Änderungswert `delprioritychar` für den Dateityp problemlos verwendet werden. Dies führt jedoch zu einer geringfügigen Verlangsamung der Ladeoperation.

Wenn die zu ladenden Daten im Format ASC (Abstract Syntax Checker) vorliegen, müssen Sie eventuell durch eine zusätzliche Maßnahme sicherzustellen, dass keine abschließenden Leerzeichen in die geladenen Sicherheitskennsätze und Namen von Sicherheitskennsätzen eingefügt werden. Da beim ASCII-Format Spaltenpositionen als Begrenzer verwendet werden, kann dies beim Laden in Felder variabler Länge vorkommen. Verwenden Sie zum Abschneiden von abschließenden Leerstellen den Änderungswert `striptblanks` für den Dateityp.

Vom Standard abweichende Sicherheitskennsatzwerte

Sie können auch Datendateien laden, bei denen die Werte für die Sicherheitskennsätze Zeichenfolgen darstellen, die die Werte der Komponenten im jeweiligen Sicherheitskennsatz enthalten, z. B. *S:(ALPHA,BETA)*. Verwenden Sie hierzu den Änderungswert `seclabelchar` für den Dateityp. Bei Verwendung von `seclabelchar` werden Werte für Spalten mit dem Datentyp DB2SECURITYLABEL als Zeichenfolgekonstanten gewertet, die den Sicherheitskennsatz im Zeichenfolgeformat für Sicherheitskennsätze enthalten. Liegt eine Zeichenfolge nicht im richtigen Format vor, wird die jeweilige Zeile nicht eingefügt und eine Warnung (SQLSTATE 01H53) zurückgegeben. Stellt eine Zeichenfolge keinen gültigen Sicherheitskennsatz dar, der Bestandteil der zum Schutz der Tabelle verwendeten Sicherheitsrichtlinie ist, wird die betreffende Zeile nicht eingefügt und eine Warnung (SQLSTATE 01H53) zurückgegeben.

Sie können auch Datendateien laden, bei denen die Werte für die Sicherheitskennsatzspalte Namen von Sicherheitskennsätzen sind. Verwenden Sie zum Laden derartiger Dateien den Änderungswert `seclabelname` für den Dateityp. Bei Verwendung von `seclabelname` werden alle Werte für Spalten mit dem Typ DB2SECURITYLABEL als Zeichenfolgekonstanten gewertet, die den Namen vorhandener Sicherheitskennsätze enthalten. Existiert kein Sicherheitskennsatz mit dem angegebenen Namen für die zum Schutz der Tabelle verwendeten Sicherheitsrichtlinie, wird die betreffende Zeile nicht geladen und eine Warnung (SQLSTATE 01H53) zurückgegeben.

Zurückgewiesene Zeilen

Zeilen, die während des Ladens zurückgewiesen werden, werden entweder an eine Speicherauszugsdatei oder eine Ausnahmetabelle gesendet (wenn sie im Befehl **LOAD** angegeben sind), je nach der Ursache für das Zurückweisen der Zeilen. Zeilen, die aufgrund von Parsing-Fehlern zurückgewiesen werden, werden an die Speicherauszugsdatei gesendet. Zeilen, die die Sicherheitsrichtlinien verletzen, werden an die Ausnahmetabelle gesendet.

Anmerkung: Es kann keine Ausnahmetabelle angegeben werden, wenn die Zieltabelle eine XML-Spalte enthält.

Beispiele

Bei allen Beispielen liegt die Eingabedatendatei `myfile.del` im Format DEL vor. Die Daten werden jeweils in eine Tabelle mit der Bezeichnung REPS geladen, die mit der folgenden Anweisung erstellt wurde:

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

Bei dem folgenden Beispiel wird davon ausgegangen, dass die Eingabedatei Sicherheitskennsätze im Standardformat enthält:

```
db2 load from myfile.del of del modified by delprioritychar insert into reps
```

Bei dem folgenden Beispiel wird davon ausgegangen, dass die Eingabedatei Sicherheitskennsätze im Zeichenfolgeformat für Sicherheitskennsätze enthält:

```
db2 load from myfile.del of del modified by seclabelchar insert into reps
```

Bei dem folgenden Beispiel wird davon ausgegangen, dass die Eingabedatei Namen von Sicherheitskennsätzen für die Sicherheitskennsatzspalte enthält:

```
db2 load from myfile.del of del modified by seclabelname insert into reps
```

Aspekte des Ladens von Identitätsspalten

Mit dem Dienstprogramm LOAD können Sie Daten in eine Tabelle laden, die eine Identitätsspalte enthält. Dabei spielt es keine Rolle, ob die Eingabedaten Identitätsspaltenwerte enthalten.

Sofern keine identitätsbezogenen Änderungswerte für den Dateityp verwendet werden, liegen der Ausführung des Dienstprogramms die folgenden Regeln zugrunde:

- Ist die Identitätsspalte als GENERATED ALWAYS definiert, wird für eine Tabellenzeile immer dann ein Identitätswert generiert, wenn die entsprechende Zeile in der Eingabedatei keinen Wert für die Identitätsspalte enthält oder explizit einen Nullwert vorgibt. Ist für die Identitätsspalte ein Wert angegeben, der kein Nullwert ist, wird die Zeile zurückgewiesen (SQL3550W).
- Ist die Identitätsspalte als GENERATED BY DEFAULT definiert, verwendet das Dienstprogramm LOAD benutzerdefinierte Werte, sofern diese vorhanden sind. Wenn diese Daten fehlen oder explizit Nullwerte sind, wird ein Wert generiert.

Das Dienstprogramm LOAD führt keine zusätzliche Gültigkeitsprüfung der benutzerdefinierten Identitätswerte durch, die über die übliche Gültigkeitsprüfung für Werte mit dem Datentyp der Identitätsspalte (also SMALLINT, INT, BIGINT oder DECIMAL) hinausgeht. Falls Werte doppelt vorhanden sind, wird keine entsprechende Meldung ausgegeben.

In den meisten Fällen kann das Dienstprogramm LOAD nicht garantieren, dass Identitätsspaltenwerte den Zeilen in derselben Reihenfolge zugeordnet werden, wie diese Zeilen in der Datendatei angezeigt werden. Da die Zuordnung von Identitätsspaltenwerten vom Dienstprogramm LOAD parallel verwaltet wird, werden diese Werte in beliebiger Reihenfolge zugeordnet. Dabei gelten die folgenden Ausnahmen:

- In Einzelpartitionsdatenbanken werden Zeilen nicht parallel verarbeitet, wenn `CPU_PARALLELISM` auf 1 gesetzt ist. In diesem Fall werden Identitätsspaltenwerte implizit in derselben Reihenfolge zugeordnet, in der die Zeilen in der Datendatei angezeigt werden.
- In Mehrpartitionsdatenbanken werden Identitätsspaltenwerte in derselben Reihenfolge zugeordnet, in der die Zeilen in der Datendatei angezeigt werden, sofern die Identitätsspalte der Verteilungsschlüssel ist und ein einziger Partitionierungsagent vorhanden ist (wenn Sie also nicht mehrere Partitionierungsagenten bzw. den Änderungswert für den Dateityp `anyorder` angeben).

Wenn eine Tabelle in eine partitionierte Datenbank geladen wird und die Tabelle über eine Identitätsspalte im Partitionierungsschlüssel verfügt und der Änderungswert `identityoverride` nicht angegeben wurde, darf die Option `SAVECOUNT` nicht angegeben werden. Wenn der Partitionierungsschlüssel eine Identitätsspalte umfasst und die Identitätswerte generiert werden, dann ist zum Neustart des Ladevorgangs über die LOAD-Phase in mindestens einer Datenbankpartition der Neustart des gesamten Ladevorgangs vom Anfang der LOAD-Phase aus erforderlich. Dies bedeutet, dass keine Konsistenzpunkte vorhanden sein können.

Anmerkung: Eine Ladeoperation mit `RESTART` ist nicht zulässig, wenn alle im Folgenden aufgeführten Kriterien zutreffen:

- Die zu ladende Tabelle befindet sich in einer Umgebung mit partitionierten Datenbanken und enthält mindestens eine Identitätsspalte, die sich entweder im Verteilungsschlüssel befindet oder auf die über eine generierte Spalte verwiesen wird, die Teil des Verteilungsschlüssels ist.

- Der Änderungswert identityoverride ist nicht angegeben.
- Die zuvor ausgeführte, fehlgeschlagene Ladeoperation umfasste das Laden von Datenbankpartitionen, dieser Vorgang ist jedoch nach der LOAD-Phase fehlgeschlagen.

Stattdessen sollte eine Ladeoperation mit TERMINATE oder REPLACE ausgeführt werden.

Sie können das Laden von Daten in Tabellen mit Identitätsspalte mit den drei sich gegenseitig ausschließenden Änderungswerten für den Dateityp identitymissing, identityignore und identityoverride vereinfachen.

Laden von Daten ohne Identitätsspalten

Der Änderungswert identitymissing vereinfacht das Laden einer Tabelle mit einer Identitätsspalte, wenn die Eingabedatendatei für die Identitätsspalte keine Werte (auch keine Nullwerte) enthält. Angenommen, es wurde beispielsweise eine Tabelle mit der folgenden SQL-Anweisung definiert:

```
create table table1 (c1 varchar(30),
                     c2 int generated by default as identity,
                     c3 decimal(7,2),
                     c4 char(1))
```

Wenn Sie Daten aus einer Datei (load.del) in die Tabelle TABLE1 laden wollen und diese Datei aus einer Tabelle exportiert wurde, die keine Identitätsspalte enthält, betrachten Sie das folgende Beispiel:

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

Eine Methode zum Laden dieser Datei wäre das explizite Auflisten der zu ladenden Spalten durch den folgenden Befehl **LOAD**:

```
db2 load from load.del of del replace into table1 (c1, c3, c4)
```

Bei einer Tabelle mit vielen Spalten ist die Verwendung dieser Syntax jedoch eventuell umständlich und fehlerträchtig. Eine alternative Methode für das Laden der Datei ist die folgende Verwendung des Änderungswerts identitymissing für den Dateityp:

```
db2 load from load.del of del modified by identitymissing
   replace into table1
```

Dieser Befehl führt dazu, dass die drei Spalten in der Datendatei in die Spalten c1, c3 und c4 von TABLE1 geladen werden. Für jede Zeile in c2 wird dabei ein Wert generiert.

Laden von Daten mit Identitätsspalten

Der Änderungswert identityignore weist das Dienstprogramm LOAD an, dass die in der Eingabedatendatei vorhandenen Daten für die Identitätsspalte ignoriert werden sollen und dass für jede Zeile ein Identitätswert generiert werden soll. Beispiel: Ein Benutzer möchte die wie zuvor definierte Tabelle TABLE1 aus einer Datendatei (load.del) mit den folgenden Daten laden:

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

Wenn die benutzerdefinierten Werte 1, 2 und 3 nicht für die Identitätsspalte verwendet werden, können Sie den folgenden Befehl **LOAD** absetzen:

```
db2 load from load.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
```

Aber auch diese Methode kann möglicherweise umständlich und fehlerträchtig sein, wenn die Tabelle zu viele Spalten enthält. Der Änderungswert `identityignore` vereinfacht die Syntax folgendermaßen:

```
db2 load from load.del of del modified by identityignore
   replace into table1
```

Laden von Daten mit vom Benutzer angegebenen Werten

Der Änderungswert `identityoverride` wird verwendet, um benutzerdefinierte Werte in eine Tabelle mit einer als GENERATED ALWAYS definierten Identitätsspalte zu laden. Dies kann bei der Migration von Daten aus einem anderen Datenbanksystem, bei der die Tabelle als GENERATED ALWAYS definiert sein muss, oder beim Laden einer Tabelle aus Daten nützlich sein, die mit der Option `DROPPED TABLE RECOVERY` des Befehls **ROLLFORWARD DATABASE** wiederhergestellt wurden. Bei Verwendung dieses Änderungswerts werden alle Zeilen ohne Daten (oder mit Nulldaten) für die Identitätsspalte zurückgewiesen (SQL3116W). Sie sollten außerdem beachten, dass bei Verwendung dieses Änderungswerts ein Verstoß gegen die Eindeutigkeitseigenschaft von Spalten eintreten kann, die mit GENERATED ALWAYS definiert sind. In dieser Situation müssen Sie eine Ladeoperation mit `TERMINATE` und anschließend eine Ladeoperation mit `INSERT` oder `REPLACE` ausführen.

Aspekte des Ladens generierter Spalten

Sie können Daten in eine Tabelle mit generierten Spalten (Tabellen mit Identitätsspalte ausgenommen) laden. Dabei spielt es keine Rolle, ob die Eingabedaten Werte für generierte Spalten enthalten. Das Dienstprogramm LOAD generiert die Spaltenwerte.

Sofern keine Änderungswerte für den Dateityp verwendet werden, die sich auf generierte Spalten beziehen, liegen der Ausführung des Dienstprogramms LOAD die folgenden Regeln zugrunde:

- Für generierte Spalten werden Werte erstellt, wenn in der entsprechende Zeile der Datendatei ein Wert für die Spalte fehlt oder ein Nullwert angegeben ist. Ist für eine generierte Spalte ein Wert angegeben, der kein Nullwert ist, wird die Zeile zurückgewiesen (SQL3550W).
- Wenn für eine generierte Spalte, die keine Nullwerte enthalten kann, ein Nullwert erstellt wird, wird die gesamte Datenzeile zurückgewiesen (SQL0407N). Dies könnte beispielsweise dann eintreten, wenn eine generierte Spalte, die keinen Nullwert enthalten kann, als Summe zweier Tabellenspalten definiert ist und diese Spalten in der Datendatei Nullwerte enthalten.

Sie können das Laden von Daten in Tabellen mit einer generierten Spalte mit den drei sich gegenseitig ausschließenden Änderungswerten für den Dateityp `generatedmissing`, `generatedignore` und `generatedoverride` vereinfachen:

Laden von Daten ohne generierte Spalten

Der Änderungswert `generatedmissing` vereinfacht das Laden einer Tabelle mit generierten Spalten, wenn die Eingabedatendatei für alle in der Tabelle vorhandenen generierten Spalten keine Werte (auch keine Nullwerte) enthält. Angenommen, es wurde beispielsweise eine Tabelle mit der folgenden SQL-Anweisung definiert:

```
CREATE TABLE table1 (c1 INT,
                                                c2 INT,
                                                g1 INT GENERATED ALWAYS AS (c1 + c2),
                                                g2 INT GENERATED ALWAYS AS (2 * c1),
                                                c3 CHAR(1))
```

Wenn Sie Daten aus einer Datei (`load.del`) in die Tabelle TABLE1 laden wollen und diese Datei aus einer Tabelle exportiert wurde, die keine generierten Spalten enthält, betrachten Sie das folgende Beispiel:

```
1, 5, J
2, 6, K
3, 7, I
```

Eine Methode für das Laden dieser Datei wäre das explizite Auflisten der zu ladenden Spalten durch den folgenden Befehl LOAD:

```
DB2 LOAD FROM load.del of del REPLACE INTO table1 (c1, c2, c3)
```

Bei einer Tabelle mit vielen Spalten ist die Verwendung dieser Syntax jedoch eventuell umständlich und fehlerträchtig. Eine alternative Methode für das Laden der Datei ist die folgende Verwendung des Änderungswerts für den Dateityp `generatedmissing`:

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedmissing
      REPLACE INTO table1
```

Dieser Befehl führt dazu, dass die drei Spalten der Datendatei in die Spalten c1, c2 und c3 von TABLE1 geladen werden. Aufgrund des Änderungswerts `generatedmissing` werden die Werte der Spalten g1 und g2 von TABLE1 automatisch generiert und werden keiner der Datendateispalten zugeordnet.

Laden von Daten mit generierten Spalten

Der Änderungswert `generatedignore` weist das Dienstprogramm LOAD an, dass die in der Eingabedatendatei vorhandenen Werte für alle in der Zieltabelle vorhandenen generierten Spalten ignoriert und in jede generierte Spalte die berechneten Werte geladen werden sollen. Beispiel: Sie möchten die wie zuvor definierte Tabelle TABLE1 aus einer Datendatei (`load.del`) mit den folgenden Daten laden:

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

Die benutzerdefinierten Werte 10, 11 und 12 (für g1) sowie 15, 16 und 17 (für g2), die keine Nullwerte sind, bewirken, dass die Zeile zurückgewiesen wird (SQL3550W), wenn keine Änderungswerte für den Dateityp verwendet werden, die den generierten Spalten zugeordnet sind. Um dies zu verhindern, könnte der Benutzer den folgenden Befehl LOAD absetzen:

```
DB2 LOAD FROM load.del of del method P(1, 2, 5)
      REPLACE INTO table1 (c1, c2, c3)
```

Aber auch diese Methode kann möglicherweise umständlich und fehlerträchtig sein, wenn die Tabelle zu viele Spalten enthält. Der Änderungswert `generatedignore` vereinfacht die Syntax folgendermaßen:

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedignore
      REPLACE INTO table1
```

Dieser Befehl führt dazu, dass die Spalten der Datendatei in die Spalten c1 (mit Daten 1, 2, 3), c2 (mit Daten 5, 6, 7) und c3 (mit Daten J, K, I) von TABLE1 geladen werden. Aufgrund des Änderungswerts `generatedignore`

werden die Werte der Spalten g1 und g2 von TABLE1 automatisch generiert und die Datendateispalten (10, 11, 12 sowie 15, 16, 17) werden ignoriert.

Laden von Daten mit vom Benutzer angegebenen Werten

Der Änderungswert `generatedoverride` wird verwendet, um benutzerdefinierte Werte in eine Tabelle mit generierten Spalten zu laden. Dies kann bei der Migration von Daten aus einem anderen Datenbanksystem oder beim Laden einer Tabelle aus Daten nützlich sein, die mit der Option `RECOVER DROPPED TABLE` des Befehls **ROLLFORWARD DATABASE** wiederhergestellt wurden. Bei Verwendung dieses Änderungswerts werden alle Zeilen ohne Daten (oder mit Nulldaten) für generierte Spalten, die keine Nullwerte enthalten können, zurückgewiesen (SQL3116W).

Bei Verwendung dieses Änderungswertes wird die Tabelle nach der Ladeoperation in den Status "Festlegen der Integrität anstehend" versetzt. Um den Status "Festlegen der Integrität anstehend" für die Tabelle aufzuheben, ohne die benutzerdefinierten Werte zu überprüfen, setzen Sie den folgenden Befehl ab:

```
SET INTEGRITY FOR tabellenname GENERATED COLUMN IMMEDIATE
      UNCHECKED
```

Um den Status "Festlegen der Integrität anstehend" für die Tabelle aufzuheben und die Überprüfung der benutzerdefinierten Werte zu erzwingen, setzen Sie den folgenden Befehl ab:

```
SET INTEGRITY FOR tabellenname IMMEDIATE CHECKED
```

Befindet sich eine generierte Spalte in einem der Partitionierungs-, Dimensions- oder Verteilungsschlüssel, wird der Änderungswert `generatedoverride` ignoriert, und das Dienstprogramm LOAD generiert Werte, als wäre der Änderungswert `generatedignore` angegeben. Diese Vorgehensweise wird angewendet, um ein Szenario zu vermeiden, in dem der benutzerdefinierte Spaltenwert zu einem Konflikt mit der Definition der generierten Spalte führt. Dies würde dazu führen, dass der Ergebnisdatensatz an der falschen physischen Position (wie beispielsweise in der falschen Datenpartition, dem falschen MDC-Block oder der falschen Datenbankpartition) abgelegt werden würde.

Anmerkung: In einem Fall unterstützt LOAD Werte von generierten Spalten NICHT, und zwar dann nicht, wenn einer der Ausdrücke der generierten Spalten eine benutzerdefinierte Funktion enthält, die abgeschirmt (FENCED) ist. Wird versucht, Daten in eine solche Tabelle zu laden, schlägt die Ladeoperation fehl. Sie können für diese Typen von generierten Spalten jedoch eigene Werte angeben, indem Sie den Änderungswert `generatedoverride` für den Dateityp verwenden.

Versetzen von Daten mithilfe des Dateityps CURSOR

Durch die Angabe des Dateityps CURSOR bei Verwendung des Befehls **LOAD** können Sie die Ergebnisse einer SQL-Abfrage direkt in eine Zieltabelle laden, ohne eine temporäre exportierte Datei zu erstellen.

Darüber hinaus können Sie Daten aus einer weiteren Datenbank laden, indem Sie in der SQL-Abfrage auf einen Kurznamen verweisen, indem Sie die Option `DATABASE` der Anweisung DECLARE CURSOR verwenden oder indem Sie bei der API-Schnittstelle den Datenträgereintrag 'sqlu_remotefetch_entry' benutzen.

Für das Versetzen von Daten mithilfe des Dateityps CURSOR stehen drei Methoden zur Verfügung. Bei der ersten Methode wird der Befehlszeilenprozessor (CLP) verwendet, bei der zweiten Methode die API und bei der dritten die Prozedur ADMIN_CMD. Die Hauptunterschiede zwischen dem CLP und der Prozedur ADMIN_CMD werden in der folgenden Tabelle gezeigt.

Tabelle 15. Unterschiede zwischen dem CLP und der Prozedur ADMIN_CMD.

Unterschiede	**Befehlszeilenprozessor (CLP)**	**Prozedur ADMIN_CMD**
Syntax	Die Abfrageanweisung und die Quellendatenbank, die vom Cursor verwendet werden, werden außerhalb des Befehls LOAD mithilfe der Anweisung DECLARE CURSOR definiert.	Die Abfrageanweisung und die Quellendatenbank, die vom Cursor verwendet werden, werden innerhalb des Befehls LOAD wie folgt definiert: LOAD from (DATABASE datenbankaliasname abfrageanweisung)
Benutzerberechtigung für den Zugriff auf eine andere Datenbank	Befinden sich die Daten in einer anderen Datenbank als derjenigen, zu der momentan eine Verbindung besteht, muss das Schlüsselwort DATABASE in der Anweisung DECLARE CURSOR verwendet werden. In derselben Anweisung können auch die Benutzer-ID und das Kennwort angegeben werden. Werden Benutzer-ID und Kennwort in der Anweisung DECLARE CURSOR nicht angegeben, werden die Benutzer-ID und das Kennwort, die für die Verbindung zur Quellendatenbank explizit angegeben wurden, für den Zugriff auf die Zieldatenbank verwendet.	Befinden sich die Daten in einer anderen Datenbank als derjenigen, zu der momentan eine Verbindung besteht, muss das Schlüsselwort DATABASE im Befehl LOAD vor der Abfrageanweisung verwendet werden. Für den Zugriff auf die Zieldatenbank sind die Benutzer-ID und das Kennwort erforderlich, die für die Verbindung zur Quellendatenbank explizit angegeben wurden. Für die Quellendatenbank selbst kann keine Benutzer-ID und kein Kennwort angegeben werden. Daher kann die Prozedur ADMIN_CMD nicht zum Laden verwendet werden, wenn beim Herstellen der Verbindung zur Datenbank keine Benutzer-ID und kein Kennwort angegeben wurden oder wenn die Benutzer-ID und das Kennwort, die angegeben wurden, nicht für die Authentifizierung der Quellendatenbank verwendet werden können.

Damit eine LOAD FROM CURSOR-Operation über den Befehlszeilenprozessor (CLP) ausgeführt werden kann, muss zunächst ein Cursor für eine SQL-Abfrage deklariert werden. Sobald ein Cursor deklariert ist, können Sie den Befehl **LOAD** absetzen. Hierbei verwenden Sie den Namen des deklarierten Cursors als Wert für *cursorname* und die Angabe CURSOR für den Dateityp.

Beispiel:

1. Angenommen, eine Quellentabelle und eine Zieltabelle mit den folgenden Definitionen befinden sich beide in derselben Datenbank:

   Die Tabelle ABC.TABLE1 enthält 3 Spalten:

   - ONE INT
   - TWO CHAR(10)
   - THREE DATE

   Die Tabelle ABC.TABLE2 enthält 3 Spalten:

   - ONE VARCHAR
   - TWO INT
   - THREE DATE

   Durch Ausführung der folgenden CLP-Befehle werden alle Daten aus der Tabelle ABC.TABLE1 in die Tabelle ABC.TABLE2 geladen:

   ```
   DECLARE mycurs CURSOR FOR SELECT TWO, ONE, THREE FROM abc.table1
         LOAD FROM mycurs OF cursor INSERT INTO abc.table2
   ```

   Anmerkung: Das vorstehende Beispiel zeigt, wie Daten aus einer SQL-Abfrage über den Befehlszeilenprozessor geladen werden. Das Laden aus einer SQL-Abfrage kann jedoch auch über die API db2Load erfolgen. Definieren Sie den Wert *piSourceList* der sqlu_media_list-Struktur zur Verwendung der Struktur *sqlu_statement_entry* und des Datenträgertyps SQLU_SQL_STMT, und definieren Sie den Wert *piFileType* als SQL_CURSOR.
2. Angenommen, die Quellentabelle und die Zieltabelle mit den folgenden Definitionen befinden sich in unterschiedlichen Datenbanken:

Tabelle ABC.TABLE1 in Datenbank 'dbsource' enthält 3 Spalten:

- ONE INT
- TWO CHAR(10)
- THREE DATE

Tabelle ABC.TABLE2 in Datenbank 'dbtarget' enthält 3 Spalten:

- ONE VARCHAR
- TWO INT
- THREE DATE

Wenn Sie die Föderation aktiviert und die Datenquelle (dsdbsource) katalogisiert haben, können Sie einen Kurznamen für die Quellendatenbank deklarieren und anschließend einen Cursor für diesen Kurznamen deklarieren und den Befehl LOAD mit der Option FROM CURSOR aufrufen, wie in folgendem Beispiel gezeigt wird:

```
CREATE NICKNAME myschema1.table1 FOR dsdbsource.abc.table1
DECLARE mycurs CURSOR FOR SELECT TWO,ONE,THREE FROM myschema1.table1
LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

Sie haben auch die Möglichkeit, die Option DATABASE der Anweisung DECLARE CURSOR zu verwenden, wie in folgendem Beispiel gezeigt wird:

```
DECLARE mycurs CURSOR DATABASE dbsource USER dsciaraf USING mypasswd
FOR SELECT TWO,ONE,THREE FROM abc.table1
   LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

Die Verwendung der Option DATABASE der Anweisung DECLARE CURSOR (bei Verwendung der API von LOAD auch als Datenträgertyp 'remotefetch' bekannt) bietet einige Vorteile gegenüber der Verwendung eines Kurznamens:

Leistung

Das Abrufen von Daten unter Verwendung des Datenträgertyps 'remotefetch' ist nahtlos in eine Ladeoperation integriert. Beim Abrufen eines Datensatzes gibt es weniger Übergangsebenen als bei der Verwendung eines Kurznamens. Ein weiterer Vorteil: Wenn die Quellentabelle und die Zieltabelle in einer Mehrpartitionsdatenbank auf identische Weise verteilt werden, kann das Dienstprogramm LOAD das Abrufen von Daten parallelisieren. Dies kann weiter zur Verbesserung der Leistung beitragen.

Hoher Bedienungskomfort

Es ist nicht erforderlich, die Funktion für Föderation zu aktivieren, eine ferne Datenquelle zu definieren oder einen Kurznamen zu deklarieren. Es ist lediglich erforderlich, die Option DATABASE (und gegebenenfalls die Optionen USER und USING) anzugeben.

Während diese Methode für katalogisierte Datenbanken verwendet werden kann, bietet die Verwendung von Kurznamen eine zuverlässige Möglichkeit, Daten aus verschiedenen Datenquellen abzurufen, die nicht ohne Weiteres katalogisiert werden können.

Zur Unterstützung dieser Funktionalität von 'remotefetch' verwendet das Dienstprogramm LOAD eine Infrastruktur, die die Funktion SOURCEUSEREXIT unterstützt. Das Dienstprogramm LOAD startet einen Prozess, der als Anwendung ausgeführt wird, um die Verbindung zur Quellendatenbank zu verwalten und den Datenabruf durchzuführen. Diese Anwendung ist einer eigenen Transaktion zugeordnet und nicht der Transaktion, unter der das Dienstprogramm LOAD ausgeführt wird.

Anmerkung:

1. Das vorstehende Beispiel zeigt, wie Daten aus einer SQL-Abfrage für eine katalogisierte Datenbank über den Befehlszeilenprozessor (CLP) unter Verwendung der Option DATABASE der Anweisung DECLARE CURSOR geladen werden können. Das Laden aus einer SQL-Abfrage für eine katalogisierte Datenbank kann jedoch auch über die API db2Load erfolgen, indem die Werte *piSourceList* und *piFileType* der *db2LoadStruct*-Struktur so definiert werden, dass der Datenträgereintrag 'sqlu_remotefetch_entry' bzw. der Datenträgertyp SQLU_REMOTEFETCH verwendet wird.
2. Wie in vorstehendem Beispiel veranschaulicht wird, müssen die Quellenspaltentypen der SQL-Abfrage nicht mit den entsprechenden Zielspaltentypen identisch sein. Kompatibilität zwischen den Spaltentypen ist jedoch erforderlich.

Einschränkungen

Beim Laden von Daten über einen Cursor, der mit der Option DATABASE definiert wurde (oder bei Verwendung des Datenträgereintrags 'sqlu_remotefetch_entry' über die API 'db2Load'), gelten die folgenden Einschränkungen:

1. Die Option SOURCEUSEREXIT kann nicht gleichzeitig angegeben werden.
2. Die Option METHOD N wird nicht unterstützt.
3. Der Änderungswert usedefaults für den Dateityp wird nicht unterstützt.

Weitergeben von abhängigen sofort gespeicherten Zwischenspeichertabellen

Wenn die geladene Tabelle die zugrunde liegende Tabelle einer Zwischenspeichertabelle mit dem Attribut IMMEDIATE PROPAGATE (sofortige Weitergabe) ist

und die Ladeoperation im Einfügemodus erfolgt, wird die nachfolgende Weitergabe in die abhängigen sofort gespeicherten Zwischenspeichertabellen inkrementell verarbeitet.

Während einer inkrementellen Weitergabe werden die Zeilen, die den angehängten Zeilen in den zugrunde liegenden Tabellen entsprechen, an die Zwischenspeichertabellen angehängt. Die inkrementelle Weitergabe beschleunigt sich, wenn große zugrunde liegende Tabellen kleine Mengen von angehängten Daten enthalten. Die Leistung wird ebenfalls verbessert, wenn die Zwischenspeichertabelle zur Aktualisierung ihrer abhängigen, verzögerten MQT verwendet wird. Es gibt Fälle, in denen eine inkrementelle Weitergabe nicht zulässig ist und die Zwischenspeichertabelle als unvollständig gekennzeichnet wird. Dies bedeutet, dass das Byte für die Zwischenspeicherung in der Spalte CONST_CHECKED den Wert F erhält. In diesem Status kann die Zwischenspeichertabelle nicht zur Aktualisierung ihrer abhängigen, verzögerten MQT verwendet werden. Stattdessen muss im Verwaltungsprozess für die MQT eine vollständige Aktualisierung erfolgen.

Falls eine Tabelle den Status "Unvollständig" aufweist und die Option `INCREMENTAL` angegeben wurde, eine inkrementelle Weitergabe der Tabelle jedoch nicht möglich ist, wird ein Fehler zurückgegeben. Wenn eine der folgenden Operationen stattgefunden hat, inaktiviert das System die sofortige Datenweitergabe und ändert den Tabellenstatus in "Unvollständig":

- Eine LOAD REPLACE-Operation wurde für eine zugrunde liegende Tabelle der Zwischenspeichertabelle ausgeführt, oder die Option NOT LOGGED INITIALLY WITH EMPTY TABLE wurde nach der letzten Überprüfung der Integrität für die zugrunde liegende Tabelle aktiviert.
- Die abhängige MQT der Zwischenspeichertabelle oder die Zwischenspeichertabelle selbst wurde im Ersetzungsmodus oder im Einfügemodus (`REPLACE` oder `INSERT`) geladen.
- Für eine zugrunde liegende Tabelle wurde der Status "Festlegen der Integrität anstehend" aufgehoben, bevor die Zwischenspeichertabelle weitergegeben wurde (durch Verwendung der Option `FULL ACCESS` während der Überprüfung der Integrität).
- Die Integrität einer zugrunde liegenden Tabelle der Zwischenspeichertabelle wurde nicht inkrementell überprüft.
- Der Tabellenbereich, der die Zwischenspeichertabelle oder ihre zugrunde liegende Tabelle enthält, wurde bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt, und die Zwischenspeichertabelle und ihre zugrunde liegende Tabelle befinden sich in unterschiedlichen Tabellenbereichen.

Wenn die Zwischenspeichertabelle einen Wert `W` in der Spalte CONST_CHECKED des Katalogs SYSCAT.TABLES enthält und die Option `NOT INCREMENTAL` nicht angegeben wird, findet eine inkrementelle Weitergabe der Zwischenspeichertabelle statt, und die Spalte CONST_CHECKED des Katalogs SYSCAT.TABLES wird mit `U` gekennzeichnet, um deutlich zu machen, dass nicht alle Daten vom System überprüft wurden.

Das folgende Beispiel veranschaulicht eine LOAD INSERT-Operation in die zugrunde liegende Tabelle UT1 der Zwischenspeichertabelle G1 und ihre abhängige, verzögerte MQT AST1. In diesem Szenario werden sowohl die Überprüfung der Integrität für UT1 als auch die Aktualisierung von AST1 inkrementell verarbeitet:

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1,G1 IMMEDIATE CHECKED;

REFRESH TABLE AST1 INCREMENTAL;
```

Aktualisieren von abhängigen IMQTs (Immediate Materialized Query Tables)

Wenn die zugrunde liegende Tabelle einer aktualisierten IMQT (Immediate Materialized Query Table, sofort gespeicherte Abfragetabelle) unter Verwendung der Option INSERT geladen wird, bewirkt die Ausführung der Anweisung SET INTEGRITY für die abhängigen MQTs, für die REFRESH IMMEDIATE definiert ist, dass die MQT inkrementell aktualisiert wird.

Während einer inkrementellen Aktualisierung werden die Zeilen, die den angehängten Zeilen in den zugrunde liegenden Tabellen entsprechen, aktualisiert und in die MQTs eingefügt. Die inkrementelle Aktualisierung beschleunigt sich, wenn große zugrunde liegende Tabellen kleine Mengen von angehängten Daten enthalten. Es gibt Fälle, in denen eine inkrementelle Aktualisierung nicht zulässig ist und stattdessen eine vollständige Aktualisierung (also die Neuberechnung der Definitionsabfrage für die MQT) verwendet wird.

Wenn die Option INCREMENTAL angegeben ist, eine inkrementelle Verarbeitung der MQT jedoch nicht möglich ist, wird in den folgenden Fällen ein Fehler zurückgegeben:

- Eine LOAD REPLACE-Operation wurde in einer zugrunde liegenden Tabelle der MQT ausgeführt, oder die Option NOT LOGGED INITIALLY WITH EMPTY TABLE wurde aktiviert, seitdem die letzte Überprüfung auf Integritätsbedingungen für die zugrunde liegende Tabelle stattgefunden hat.
- Die MQT wurde geladen (im Ersetzungsmodus oder im Einfügemodus).
- Für eine zugrunde liegende Tabelle wurde der Status "Festlegen der Integrität anstehend" aufgehoben, bevor die MQT aktualisiert wurde (durch Verwendung der Option FULL ACCESS während der Überprüfung der Integrität).
- Die Integrität einer zugrunde liegenden Tabelle der MQT wurde nicht inkrementell überprüft.
- Die MQT befand sich vor dem Upgrade im Status "Festlegen der Integrität anstehend".
- Der Tabellenbereich, der die MQT oder ihre zugrunde liegende Tabelle enthält, wurde bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt, und die MQT und ihre zugrunde liegende Tabelle befinden sich in unterschiedlichen Tabellenbereichen.

Wenn die MQT einen oder mehrere Werte W in der Spalte CONST_CHECKED des Katalogs SYSCAT.TABLES enthält und die Option NOT INCREMENTAL in der Anweisung SET INTEGRITY nicht angegeben ist, wird die Tabelle inkrementell aktualisiert, und die Spalte CONST_CHECKED des Katalogs SYSCAT.TABLES wird mit U gekennzeichnet, um deutlich zu machen, dass nicht alle Daten vom System überprüft wurden.

Das folgende Beispiel veranschaulicht eine Ladeoperation in die zugrunde liegende Tabelle UTI der MQT mit der Bezeichnung AST1. Die Tabelle UT1 wird auf Datenintegrität überprüft und in den Modus "Kein Versetzen von Daten" versetzt. Sobald die inkrementelle Aktualisierung der Tabelle AST1 abgeschlossen ist, wird die Ta-

belle UT1 wieder in den Status "Vollständiger Zugriff" versetzt. In diesem Szenario werden sowohl die Überprüfung der Integrität für UT1 als auch die Aktualisierung von AST1 inkrementell verarbeitet.

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1 IMMEDIATE CHECKED;
REFRESH TABLE AST1;
```

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.

Versetzen von Daten mithilfe einer angepassten Anwendung (Benutzerexit)

Die LOAD-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.

Die Aufgabe des Benutzerexits besteht darin, mindestens eine benannte Pipe mit Daten aufzufüllen, die gleichzeitig vom Dienstprogramm LOAD gelesen werden. In einer Mehrpartitionsdatenbank können mehrere Instanzen des Benutzerexits gleichzeitig aufgerufen werden, um eine parallele Verarbeitung der Eingabedaten zu erreichen.

Wie Abb. 5 auf Seite 72 zeigt, erstellt das Dienstprogramm LOAD mindestens eine benannte Pipe und startet einen Prozess zum Ausführen des angepassten ausführbaren Codes. Der Benutzerexit führt der bzw. den benannten Pipe(s) Daten zu, während das Dienstprogramm LOAD die Daten gleichzeitig liest.

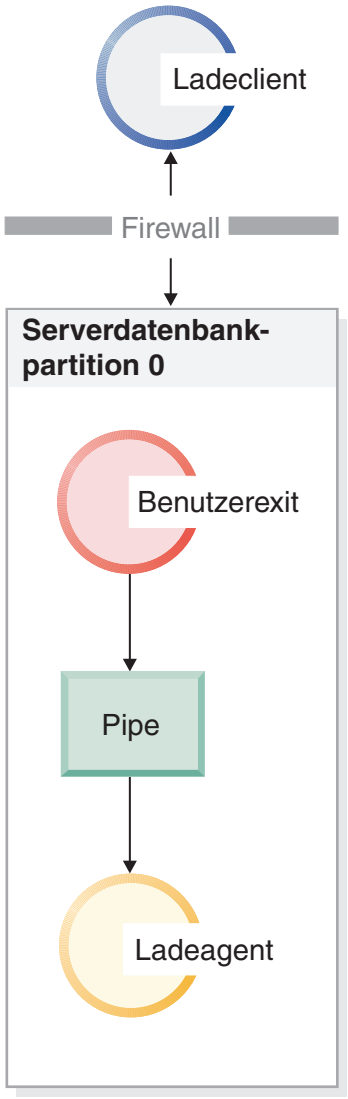


Abbildung 5. Das Dienstprogramm LOAD liest aus der Pipe und verarbeitet die ankommenden Daten.

Die Daten, die der Pipe zugeführt werden, müssen den angegebenen LOAD-Optionen entsprechen, einschließlich Dateityp und allen Änderungswerten für den Dateityp. Die angegebenen Datendateien werden vom Dienstprogramm LOAD nicht direkt gelesen. Stattdessen werden die angegebenen Datendateien als Argumente an den Benutzerexit übergeben, wenn dieser ausgeführt wird.

Aufrufen des Benutzerexits

Der Benutzerexit muss sich im Unterverzeichnis 'bin' des DB2-Installationsverzeichnisses (häufig 'sqllib') befinden. Das Dienstprogramm LOAD ruft den ausführbaren Code des Benutzerexits mit folgenden Befehlszeilenparametern auf:

```
<pipebasisdateiname> <anzahl_der_quellendatenträger>
<quellendatenträger_1> <quellendatenträger_2> ... <benutzerexit-id>
<anzahl_der_benutzerexits> <nummer_der_datenbankpartition>
```

Dabei gilt Folgendes:

<pipebasisdateiname>
Dies ist der Basisname für benannte Pipes, die das Dienstprogramm LOAD

erstellt und aus denen es Daten liest. Das Dienstprogramm erstellt für jede im Befehl LOAD angegebene Quellendatei jeweils eine Pipe, und an den Namen jeder dieser Pipes wird die Erweiterung .xxx angehängt, wobei xxx der Index der angegebenen Quellendatei ist. Werden beispielsweise 2 Quellendateien im Befehl LOAD angegeben und lautet der an den Benutzerexit übergebene Wert für das Argument <pipebasisdateiname> pipe123, werden den beiden benannten Pipes pipe123.000 und pipe123.001 Daten vom Benutzerexit zugeführt. In Umgebung mit partitionierten Datenbanken wird die Nummer der Datenbankpartition (DBPARTITION) .yyy vom Dienstprogramm LOAD an den Basisnamen der Pipes angehängt, sodass die Namen der Pipes das folgende Format aufweisen: pipe123.xxx.yyy..

<anzahl_der_quellendatenträger>
Dies ist die Anzahl der nachfolgenden Datenträgerargumente.

<quellendatenträger_1> <quellendatenträger_2> ...
Dies ist die Liste der Quellendateien (mindestens eine), die im Befehl LOAD angegeben sind. Jede Quellendatei ist in doppelte Anführungszeichen gesetzt.

<benutzerexit_id>
Dies ist ein Sonderwert, der bei aktivierter Option PARALLELIZE nützlich ist. Dieser ganzzahlige Wert (von 1 bis N, wobei N die Gesamtanzahl der gestarteten Benutzerexits ist) identifiziert eine bestimmte Instanz eines aktiven Benutzerexits. Bei inaktivierter Option PARALLELIZE ist 1 der Standardwert.

<anzahl_der_benutzerexits>
Dies ist ein Sonderwert, der bei aktivierter Option PARALLELIZE nützlich ist. Dieser Wert stellt die Gesamtanzahl der gleichzeitig aktiven Benutzerexits dar. Bei inaktivierter Option PARALLELIZE ist 1 der Standardwert.

<nummer_der_datenbankpartition>
Dies ist ein Sonderwert, der bei aktivierter Option PARALLELIZE nützlich ist. Hierbei handelt es sich um die Nummer der Datenbankpartition (DBPARTITION), auf der der Benutzerexit ausgeführt wird. Bei inaktivierter Option PARALLELIZE ist 0 der Standardwert.

Zusätzliche Optionen und Funktionen

Im folgenden Abschnitt werden die zusätzlichen Optionen der Funktion SOURCEUSEREXIT beschrieben:

REDIRECT
Mit dieser Option können Sie Daten an die Kennung STDIN übergeben oder Daten aus den Kennungen STDOUT und STDERR des Benutzerexitprozesses erfassen.

INPUT FROM BUFFER <puffer>
Mit dieser Option können Sie Informationen direkt in den STDIN-Eingabedatenstrom Ihres Benutzerexits übergeben. Nach dem Start des Prozesses, der den Benutzerexit ausführt, fordert das Dienstprogramm LOAD den Dateideskriptor für den STDIN-Eingabedatenstrom dieses neuen Prozesses an und übergibt den angegebenen Puffer. Der Benutzerexit liest die Informationen aus STDIN. Das Dienstprogramm LOAD sendet den Inhalt von <puffer> einfach über STDIN an den Benutzerexit, ohne den Pufferinhalt zu interpretieren oder zu ändern. Wenn ein Benutzerexit beispielsweise zwei Werte aus STDIN lesen soll (eine Benutzer-ID mit 8 Byte und ein

Kennwort mit 8 Byte), könnte ein in C geschriebener ausführbarer Benutzerexit-Code folgende Zeilen enthalten:

```
rc = read (stdin, pUserID, 8);
rc = read (stdin, pPasswd, 8);
```

Ein Benutzer könnte diese Informationen mithilfe der Option `INPUT FROM BUFFER` übergeben, wie in folgendem Befehl LOAD gezeigt:

```
LOAD FROM myfile1 OF DEL INSERT INTO table1
SOURCEUSEREXIT myuserexit1 REDIRECT INPUT FROM BUFFER myuseridmypasswd
```

Anmerkung: Das Dienstprogramm LOAD begrenzt die Größe von <puffer> auf die Maximalgröße eines LOB-Wertes. Innerhalb des Befehlszeilenprozessors (CLP) ist die Größe von <puffer> jedoch auf die Maximalgröße einer CLP-Anweisung begrenzt. Innerhalb von CLP wird außerdem empfohlen, für den Inhalt von <puffer> nur traditionelle ASCII-Zeichen zu verwenden. Diese Einschränkungen lassen sich umgehen, indem das Dienstprogramm LOAD mithilfe der API db2Load aufgerufen wird oder indem stattdessen die Option `INPUT FROM FILE` verwendet wird.

INPUT FROM FILE <dateiname>
Mit dieser Option können Sie den Inhalt einer Datei auf Clientseite direkt in den STDIN-Eingabedatenstrom des Benutzerexits übergeben. Diese Option ist beinahe mit der Option `INPUT FROM BUFFER` identisch, vermeidet jedoch die mögliche CLP-Einschränkung. Der Dateiname muss eine vollständig qualifizierte Datei auf der Clientseite sein und darf nicht größer sein als die Maximalgröße eines LOB-Wertes.

OUTPUT TO FILE <dateiname>
Mit dieser Option können Sie die STDOUT- und STDERR-Datenströme aus dem Benutzerexitprozess in einer Datei auf der Serverseite erfassen. Nach dem Start des Prozesses, der den ausführbaren Code des Benutzerexits ausführt, leitet das Dienstprogramm LOAD die Kennungen STDOUT und STDERR aus diesem neuen Prozess in die Datei um, deren Name angegeben wurde. Diese Option ist nützlich beim Debugging und Protokollieren von Fehlern und Aktivitäten innerhalb des Benutzerexits. Bei dem Dateinamen muss es sich um eine vollständig qualifizierte Datei auf der Serverseite handeln. Ist die Option `PARALLELIZE` aktiviert, ist pro Benutzerexit eine Datei vorhanden, und an jede Datei wird eine dreistellige numerische Kennung angehängt (*dateiname.000*).

PARALLELIZE
Mit dieser Option kann der Durchsatz der im Dienstprogramm LOAD eingehenden Daten erhöht werden, indem mehrere Benutzerexitprozesse gleichzeitig aufgerufen werden. Diese Option steht nur für Mehrpartitionsdatenbanken zur Verfügung. Die Anzahl der aufgerufenen Instanzen des Benutzerexits entspricht der Anzahl der Partitionierungsagenten, wenn die Daten während der Ladeoperation auf mehrere Datenbankpartitionen verteilt werden sollen. Ansonsten entspricht sie der Anzahl der Ladeagenten.

Die an die einzelnen Benutzerexits übergebenen Argumente <benutzerexit_id>, <anzahl_der_benutzerexits> und <nummer_der_datenbankpartition> geben jeweils die eindeutige Kennung (1 bis N), die Gesamtanzahl der Benutzerexits (N) und die Nummer der Datenbankpartition (DBPARTITION), unter der die Benutzerexitinstanz ausgeführt wird, wieder. Sie sollten sicherstellen, dass Daten, die von einem der Benutzerexitprozesse in die benannte Pipe geschrieben werden, nicht von den anderen gleichzeitig ablaufenden Prozessen dupliziert werden. Obwohl es mehrere Möglichkeiten für eine Benutzerexitanwendung gibt, dies zu erreichen, können die-

se Werte dabei helfen zu verhindern, dass Daten dupliziert werden. Beispiel: Wenn alle Datensätze jeweils einen eindeutigen ganzzahligen Spaltenwert enthalten, könnte eine Benutzerexitanwendung die Werte von <benutzerexit-ID> und <anzahl_der_benutzerexits> verwenden, um sicherzustellen, dass jede Benutzerexitinstanz eine eindeutige Ergebnismenge für die entsprechende benannte Pipe zurückgibt. Die Benutzerexitanwendung könnte die Eigenschaft **MODULUS** wie folgt verwenden:

```
i = <benutzerexit_id>
N = <anzahl_der_benutzerexits>

foreach record
{
      if ((unique-integer MOD N) == i)
   {
         write this record to my named-pipe
   }
}
```

Die Anzahl der gestarteten Benutzerexitprozesse hängt von dem Verteilungsmodus ab, der für die Datenbankpartitionierung angegeben wurde:

1. Wie Abb. 6 auf Seite 76 zeigt, wird für jeden Agenten für Partitionierungsvorbereitung jeweils ein Benutzerexitprozess gestartet, wenn `PARTITION_AND_LOAD` (Standardeinstellung) oder `PARTITION_ONLY` ohne PARALLEL angegeben ist.

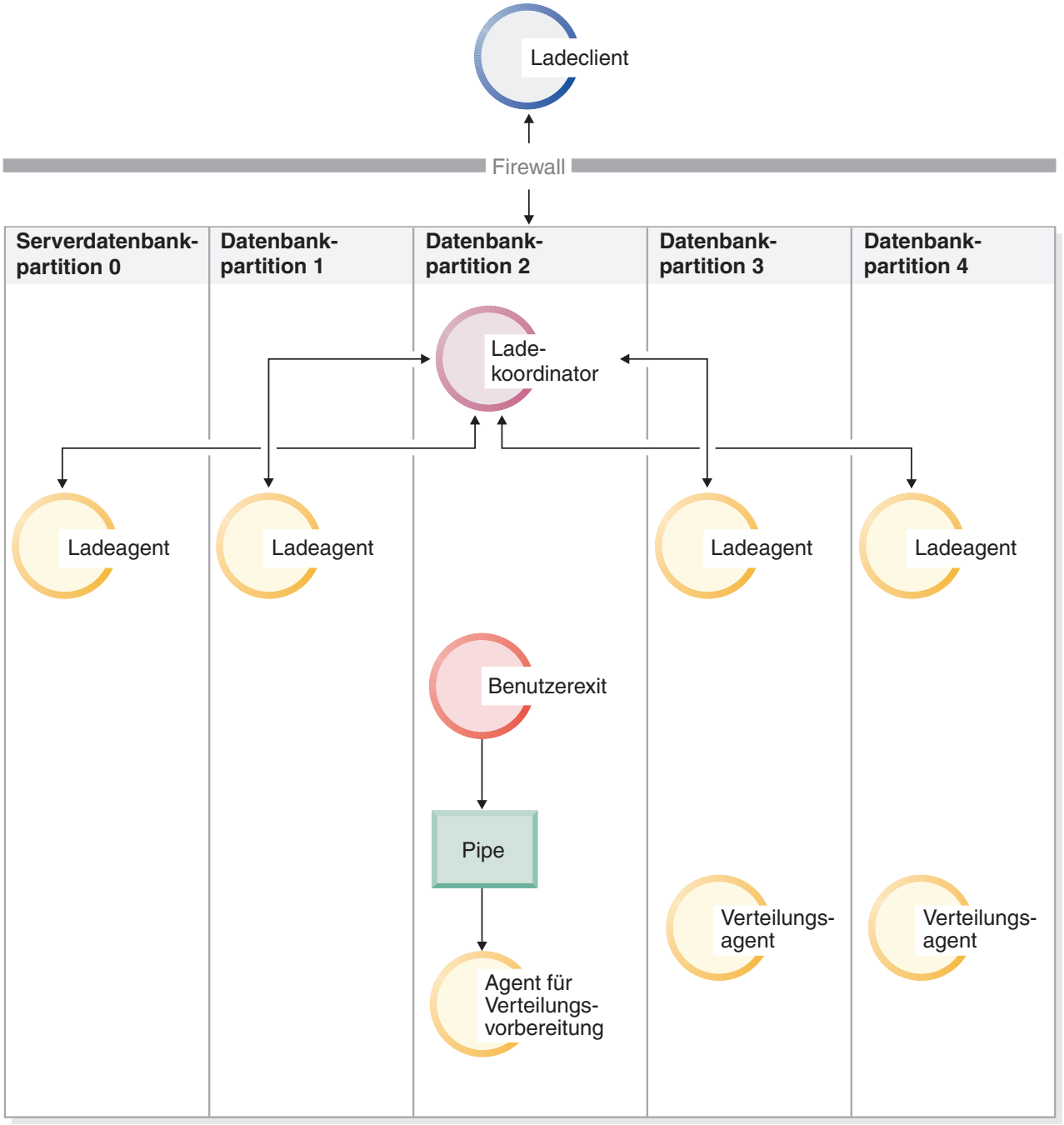


Abbildung 6. Verschiedene ausgeführte Tasks, wenn `PARTITION_AND_LOAD` (Standardeinstellung) oder `PARTITION_ONLY` ohne PARALLEL angegeben ist.

2. Wie Abb. 7 auf Seite 77 zeigt, wird für jeden Partitionierungsagenten jeweils ein Benutzerexitprozess gestartet, wenn `PARTITION_AND_LOAD` (Standardeinstellung) oder `PARTITION_ONLY` mit der Option `PARALLEL` angegeben ist.

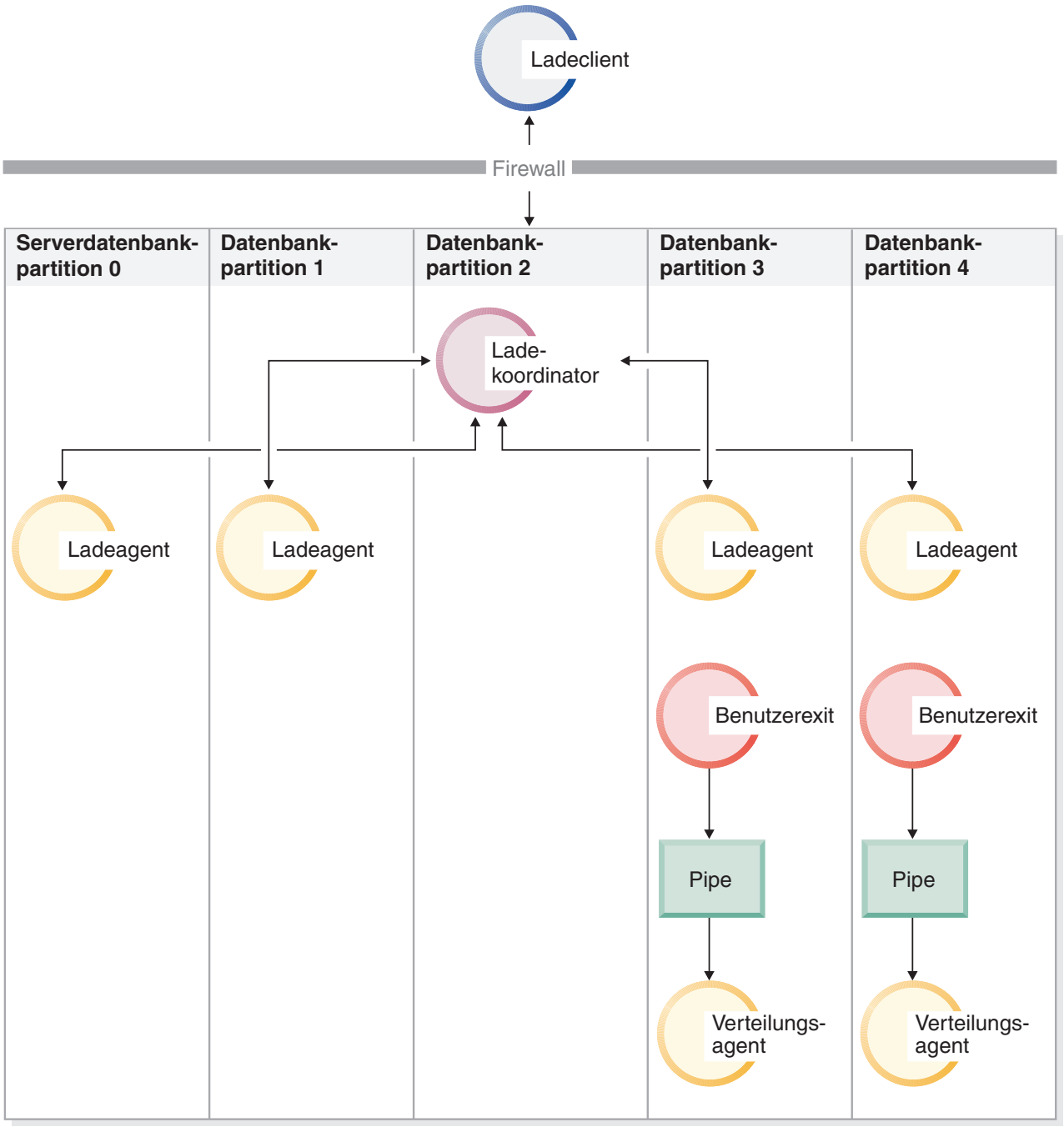


Abbildung 7. Verschiedene ausgeführte Tasks, wenn `PARTITION_AND_LOAD` (Standardeinstellung) oder `PARTITION_ONLY` mit PARALLEL angegeben ist.

3. Wie Abb. 8 auf Seite 78 zeigt, wird für jeden Ladeagenten jeweils ein Benutzerexitprozess gestartet, wenn `LOAD_ONLY` oder `LOAD_ONLY_VERIFY_PART` angegeben ist.

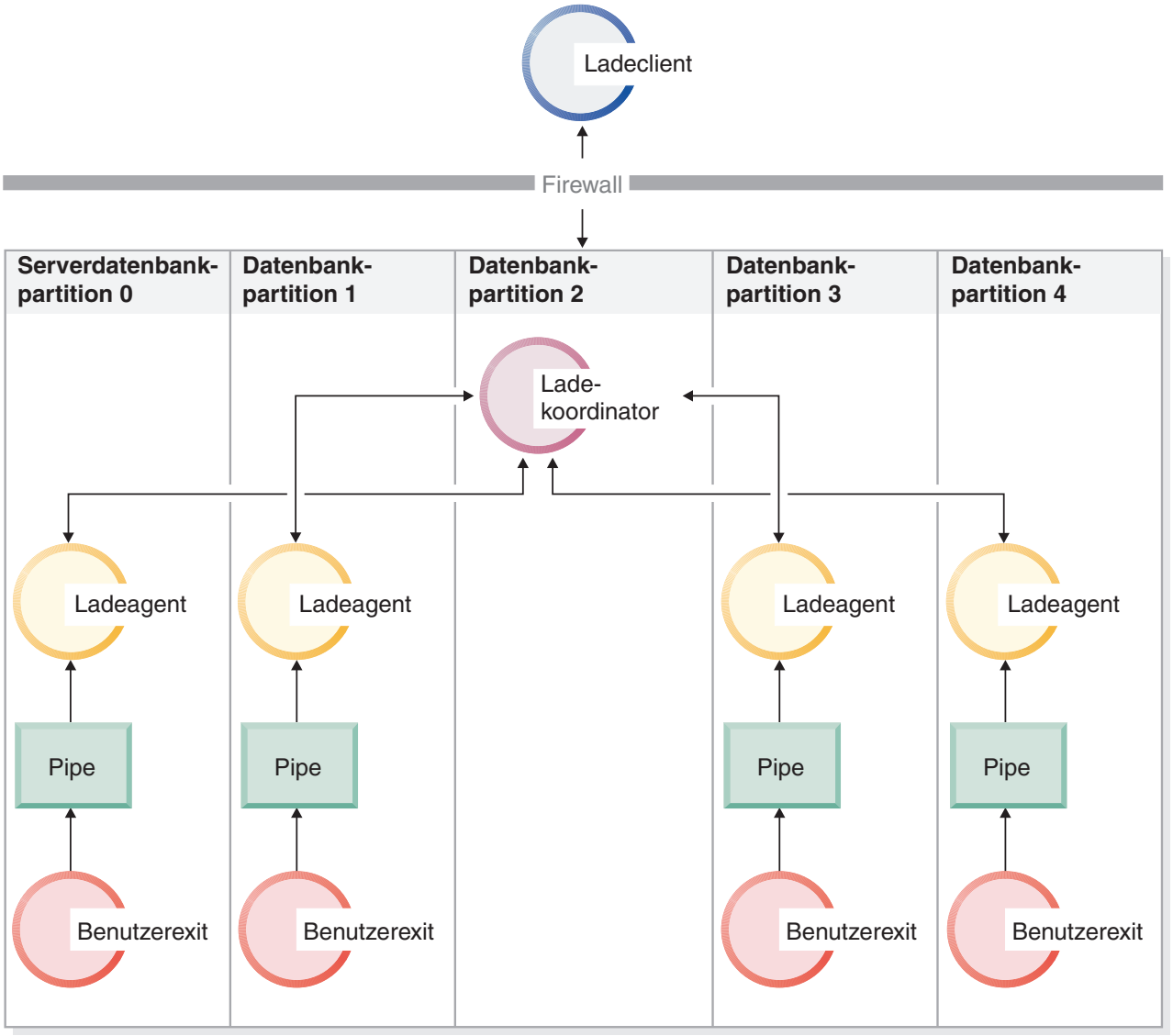


Abbildung 8. Verschiedene ausgeführte Tasks, wenn LOAD_ONLY oder LOAD_ONLY_VERIFY_PART angegeben ist.

Einschränkungen

- Die Optionen LOAD_ONLY und LOAD_ONLY_VERIFY_PART des Modus 'partitioned-db-cfg' werden nur unterstützt, wenn die SOURCEUSEREXIT-Option PARALLELIZE angegeben ist.

Beispiele

Beispiel 1: LOAD-Benutzerexitscript, das für alle Datensätze der Quellendatenträgerdatei alle Tabulatorzeichen '\t' durch Kommas ',' ersetzt. Verwenden Sie zum Aufrufen des Dienstprogramms LOAD mithilfe dieses Benutzerexitscripts einen Befehl ähnlich dem folgenden:

```
DB2 LOAD FROM /path/file1 OF DEL INSERT INTO schema1.table1
SOURCEUSEREXIT example1.pl REDIRECT OUTPUT TO FILE /path/ue_msgs.txt
```

Beachten Sie, dass der Benutzerexit im Ordner sqllib/bin/ gespeichert sein muss und Ausführungsberechtigungen erfordert.

example1.pl:

```
#!/bin/perl

# Dateiname: example1.pl
#
# Dieses Script ist ein einfaches Beispiel für einen Benutzerexit für die Funktion
# SOURCEUSEREXIT des Dienstprogramms LOAD. Es ersetzt für alle Datensätze der
# Quellendatenträgerdatei alle Tabulatorzeichen '\t' durch Kommas ','.
#
# Mit einen Befehl ähnlich dem folgenden wird LOAD über diesen Benutzerexit aufgerufen:
#
#  db2 LOAD FROM /path/file1 OF DEL INSERT INTO schema1.table1
#   SOURCEUSEREXIT example1.pl REDIRECT OUTPUT TO FILE /path/ue_msgs.txt
#
# Der Benutzerexit muss im Ordner sqllib/bin/ gespeichert sein und erfordert
# Ausführungsberechtigungen.
#------------------------------------------------------------------------
if ($#ARGV < 5)
{
  print "Ungültige Anzahl von Argumenten:\n@ARGV\n";
  print "LOAD muss den Benutzerexit mit mindestens 5 Argumenten aufrufen:\n";
  print "<pipebasisdateiname> <anzahl_der_quellendatenträger> ";
  print "<quellendatenträger_1> <quellendatenträger_2> ... <benutzerexit-id> ";
  print "<anzahl_der_benutzerexits> <nummer_der_datenbankpartition> ";
  print "<optional:_eingabe_umleiten> \n";
  die;
}

# FIFO-Ausgabedatei öffnen (LOAD liest Daten aus dieser Pipe)
#------------------------------------------------------------------------
$basePipeName = $ARGV[0];
$outputPipeName = sprintf("%s.000", $basePipeName);
open(PIPETOLOAD, '>', $outputPipeName) || die "$outputPipeName kann nicht geöffnet werden";

# Anzahl der Datenträgerdateien ermitteln
#-------------------------
$NumMediaFiles = $ARGV[1];

# Jede Datenträgerdatei öffnen, Inhalt lesen, '\t' durch ',' ersetzen an LOAD senden
#-----------------------------------------------------------------------------
for ($i=0; $i<$NumMediaFiles; $i++)
{
  # Multimediadatei öffnen
  #--------------------
  $mediaFileName = $ARGV[2+$i];
  open(MEDIAFILETOREAD, '<', $mediaFileName) || die "$mediaFileName kann nicht geöffnet werden";

  # Jeden Datensatz lesen
  #----------------------
  while ( $line = <MEDIAFILETOREAD> )
  {
    # '\t' durch ',' ersetzen
    #--------------------------------
    $line =~ s/\t/,/g;

    # Diesen Datensatz zur Verarbeitung an LOAD senden
    #-----------------------------------------
    print PIPETOLOAD $line;
  }
  # Datenträgerdatei schließen
  #--------------------
  close MEDIAFILETOREAD;
}

# FIFO schließen
```

```
#---------------
close PIPETOLOAD;

exit 0;
```

Überwachen einer Ladeoperation mit dem Befehl LIST UTILITIES

Sie können den Befehl **LIST UTILITIES** verwenden, um den Fortschritt von Ladeoperationen für eine Datenbank zu überwachen.

Vorgehensweise

Zur Verwendung des Befehls **LIST UTILITIES** gehen Sie wie folgt vor:
Setzen Sie den Befehl **LIST UTILITIES** ab und geben Sie dabei den Parameter **SHOW DETAIL** an:

```
list utilities show detail
```

Beispiel

Es folgt eine Beispielausgabe der Leistungsüberwachung einer Ladeoperation mit dem Befehl **LIST UTILITIES**:

```
ID                             = 10
Typ                            = LOAD
Datenbankname                  = TEST
Membernummer                   = 1
Beschreibung                   = OFFLINE LOAD DEL AUTOMATIC INDEXING REPLACE
COPY NO BEER   .TABLE1
Startzeit                      = 08/16/2011 08:52:53.861841
Status                         = Wird ausgeführt
Aufruftyp                      = Benutzer
Fortschrittsüberwachung:
   Phasennummer                = 1
      Beschreibung             = SETUP
      Gesamte Arbeit           = 0 Byte
      Abgeschlossene Arbeit    = 0 Byte
      Startzeit                = 08/16/2011 08:52:53.861865

   Phasennummer [Aktuell]      = 2
      Beschreibung             = LOAD
      Gesamte Arbeit           = 49900 Zeilen
      Abgeschlossene Arbeit    = 25313 Zeilen
      Startzeit                = 08/16/2011 08:52:54.277687

   Phasennummer                = 3
      Beschreibung             = BUILD
      Gesamte Arbeit           = 2 Indizes
      Abgeschlossene Arbeit    = 0 Indizes
      Startzeit                = Nicht gestartet
```

Weitere Aspekte des Ladens

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. 9). Eine weitere Möglichkeit ist der Einsatz mehrerer Prozessoren in einer SMP-Umgebung, der eine partitionsinterne Parallelität ermöglicht (siehe Abb. 10). Beide Möglichkeiten können kombiniert verwendet werden, um ein noch schnelleres Laden der Daten zu erreichen.

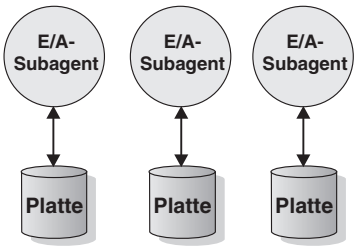


Abbildung 9. E/A-Parallelität beim Laden von Daten

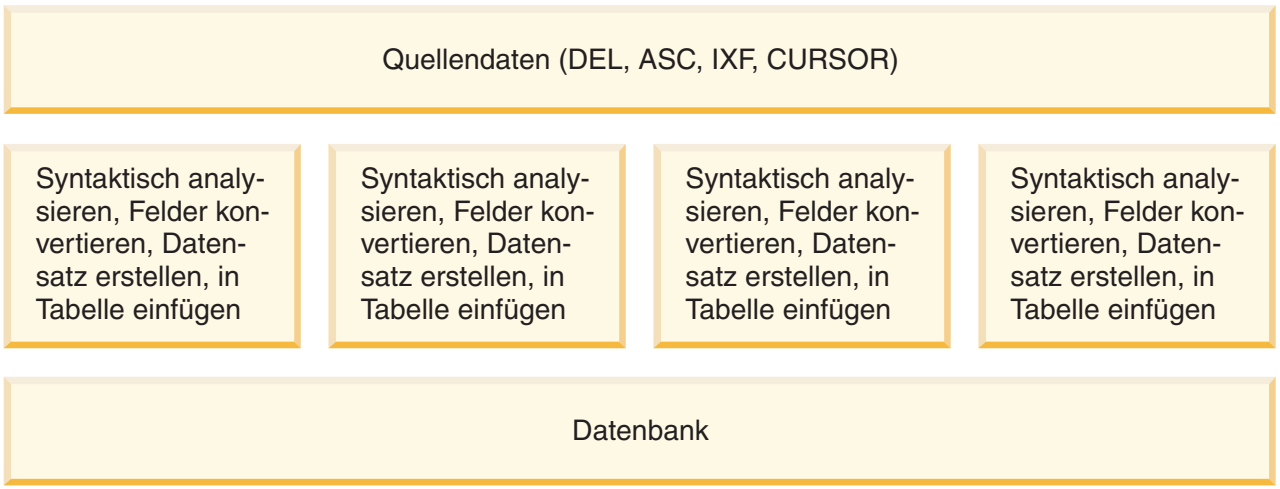


Abbildung 10. Partitionsinterne Parallelität beim Laden von Daten

Indexerstellung während einer Ladeoperation

Indizes werden im Rahmen der BUILD-Phase einer Ladeoperation erstellt. Es gibt vier Indexierungsmodi, die im Befehl **LOAD** angegeben werden können:

1. `REBUILD`. Alle Indizes werden erneut erstellt.
2. `INCREMENTAL`. Indizes werden mit neuen Daten erweitert.
3. `AUTOSELECT`. Das Dienstprogramm LOAD wählt automatisch den Modus `REBUILD` oder `INCREMENTAL` aus. `AUTOSELECT` ist der Standardwert. Wenn eine Operation **LOAD** oder **REPLACE** ausgeführt wird, verwendet das System den Indexierungsmodus `REBUILD`. Andernfalls basiert der ausgewählte Indexierungsmodus auf dem Verhältnis zwischen dem Volumen der in der Tabelle vorhandenen Daten und dem Volumen der neu geladenen Daten. Wenn dieses Verhältnis ausreicht, wird der Indexierungsmodus `INCREMENTAL` ausgewählt. Andernfalls verwendet das System den Indexierungsmodus `REBUILD`.
4. `DEFERRED`. Das Dienstprogramm LOAD versucht nicht, Indizes zu erstellen, wenn dieser Modus angegeben wird. Die Indizes werden als aktualisierungsbedürftig gekennzeichnet, und möglicherweise wird eine erneute Erstellung erzwungen, wenn zum ersten Mal auf die Indizes zugegriffen wird. Die Option `DEFERRED` ist in folgenden Situationen nicht zulässig:

- Wenn die Option **ALLOW READ ACCESS** angegeben wurde. (Bei dieser Option werden die Indizes nicht beibehalten, die Indexsuchfunktionen benötigen jedoch einen gültigen Index.)
- Wenn für die Tabelle eindeutige Indizes definiert sind.
- Wenn XML-Daten geladen werden (der XML-Pfadindex ist eindeutig und wird standardmäßig erstellt, wenn eine XML-Spalte zu einer Tabelle hinzugefügt wird).

Bei Ladeoperationen, für die die Option **ALLOW READ ACCESS** angegeben ist, müssen abhängig vom gewählten Indexierungsmodus besondere Gegebenheiten hinsichtlich der Speicherauslastung und Protokollierung berücksichtigt werden. Wenn die Option **ALLOW READ ACCESS** angegeben ist, hält das Dienstprogramm LOAD Indizes sogar während ihrer erneuten Erstellung für Abfragen zur Verfügung.

Gibt eine Ladeoperation im Modus **ALLOW READ ACCESS** die Option **INDEXING MODE INCREMENTAL** an, schreibt das Dienstprogramm LOAD einige Protokollsätze, die die Integrität der Indexbaumstruktur gewährleisten. Die Anzahl der geschriebenen Protokollsätze entspricht einem Bruchteil der Anzahl der eingefügten Schlüssel. Sie ist erheblich kleiner, als dies bei einer ähnlichen SQL-Operation INSERT erforderlich wäre. Eine Ladeoperation im Modus **ALLOW NO ACCESS** mit angegebener Option **INDEXING MODE INCREMENTAL** schreibt neben den üblichen Protokollen zur Speicherbereichszuordnung nur einen kleinen Protokollsatz.

Anmerkung: Dies gilt allerdings nur, wenn nicht **COPY YES** angegeben wurde und wenn für den Konfigurationsparameter **logindexrebuild** die Einstellung ON ausgewählt wurde.

Wenn eine Ladeoperation im Modus **ALLOW READ ACCESS** die Option **INDEXING MODE REBUILD** angibt, werden neue Indizes als *Schattenindizes* erstellt, und zwar entweder im selben Tabellenbereich wie der ursprüngliche Index oder in einem Tabellenbereich für temporäre Systemtabellen. Die ursprünglichen Indizes bleiben erhalten und sind während der Ladeoperation verfügbar. Sie werden erst am Ende der Ladeoperation durch die neuen Indizes ersetzt, während die Tabelle exklusiv gesperrt ist. Falls die Ladeoperation fehlschlägt und die Transaktion rückgängig gemacht wird, bleiben die ursprünglichen Indizes erhalten.

In der Standardeinstellung wird der Schattenindex im gleichen Tabellenbereich wie der ursprüngliche Index erstellt. Da der ursprüngliche Index und der neue Index gleichzeitig verwaltet werden, muss ausreichend Tabellenbereich für die gleichzeitige Aufnahme beider Indizes vorhanden sein. Wenn die Ladeoperation abgebrochen wird, wird der zusätzliche Bereich, der zur Erstellung des neuen Indexes verwendet wird, freigegeben. Beim Commit der Ladeoperation wird der Bereich, der für den ursprünglichen Index verwendet wurde, freigegeben, und der neue Index wird zum aktuellen Index. Wenn neue Indizes in demselben Tabellenbereich wie die ursprünglichen Indizes erstellt werden, findet das Ersetzen der ursprünglichen Indizes fast sofort statt.

Wenn die Indizes in einem SMS-Tabellenbereich erstellt werden, können Sie Indexdateien im Tabellenbereichsverzeichnis mit dem Suffix .IN1 und dem Suffix .INX erkennen. Diese Suffixe lassen nicht darauf schließen, welches der ursprüngliche Index ist und welches der Schattenindex ist. Werden die Indizes jedoch in einem DMS-Tabellenbereich erstellt, ist der neue Schattenindex nicht erkennbar.

Verbessern der Leistung bei der Indexerstellung

Erstellen von neuen Indizes in einem Tabellenbereich für temporäre Systemtabellen

Der neue Index kann in einem Tabellenbereich für temporäre Systemtabellen erstellt werden. Auf diese Weise kann verhindert werden, dass im eigentlichen Tabellenbereich nicht genügend Speicher zur Verfügung steht. Mit der Option **USE** *tabellenbereichsname* können die Indizes bei Verwendung der Optionen **INDEXING MODE REBUILD** und **ALLOW READ ACCESS** in einem Tabellenbereich für temporäre Systemtabellen erneut erstellt werden. Der Tabellenbereich für temporäre Systemtabellen kann ein SMS-Tabellenbereich oder ein DMS-Tabellenbereich sein. Die Seitengröße des Tabellenbereichs für temporäre Systemtabellen muss jedoch mit der Seitengröße des Tabellenbereichs für den ursprünglichen Index identisch sein.

Die Option **USE** *tabellenbereichsname* wird ignoriert, wenn die Ladeoperation nicht im Modus **ALLOW READ ACCESS** ausgeführt wird oder wenn der Indexierungsmodus nicht kompatibel ist. Die Option **USE** *tabellenbereichsname* wird nur bei der Option **INDEXING MODE REBUILD** oder **INDEXING MODE AUTOSELECT** unterstützt. Wenn die Option **INDEXING MODE AUTOSELECT** angegeben ist und das Dienstprogramm LOAD die inkrementelle Verwaltung der Indizes auswählt, wird die Option **USE** *tabellenbereichsname* ignoriert.

Eine LOAD RESTART-Operation kann einen alternativen Tabellenbereich zur Indexerstellung sogar dann einsetzen, wenn die ursprüngliche Ladeoperation keinen alternativen Tabellenbereich verwendete. Eine LOAD RESTART-Operation kann nicht im Modus **ALLOW READ ACCESS** abgesetzt werden, wenn die ursprüngliche Ladeoperation nicht im Modus **ALLOW READ ACCESS** abgesetzt wurde. LOAD TERMINATE-Operationen führen keinen Indexrebuild durch. Daher wird die Option **USE** *tabellenbereichsname* ignoriert.

Während der BUILD-Phase der Ladeoperation werden die Indizes im Tabellenbereich für temporäre Systemtabellen erstellt. Anschließend wird der Index während der INDEX COPY-Phase aus dem Tabellenbereich für temporäre Systemtabellen in den Tabellenbereich des ursprünglichen Indexes kopiert. Um zu gewährleisten, dass im Tabellenbereich des ursprünglichen Indexes genug Speicherbereich für den neuen Index vorhanden ist, wird während der BUILD-Phase im ursprünglichen Tabellenbereich Speicher zugeordnet. Wenn also bei einer Ladeoperation nicht genügend Speicherbereich für den Index vorhanden ist, dann erfolgt dies während der BUILD-Phase. In einem solchen Fall geht der ursprüngliche Index nicht verloren.

Die INDEX COPY-Phase schließt sich an die BUILD- und DELETE-Phase an. Bevor die INDEX COPY-Phase beginnt, wird die Tabelle exklusiv gesperrt. Dies bedeutet, dass sie während der gesamten INDEX COPY-Phase nicht im Lesezugriff verfügbar ist. Da die INDEX COPY-Phase eine physische Kopie erstellt, kann es sein, dass die Tabelle über einen beträchtlichen Zeitraum nicht verfügbar ist.

Anmerkung: Wenn es sich entweder bei dem Tabellenbereich für temporäre Systemtabellen oder beim Tabellenbereich für den Index um einen DMS-Tabellenbereich handelt, kann das Lesen aus dem Tabellenbereich für temporäre Systemtabellen eine wahlfreie Ein-/Ausgabe für den Tabellenbereich für temporäre Systemtabellen bewirken und zu einer Verzögerung führen. Das Schreiben in den Tabellenbereich für den Index wird weiterhin optimiert und die Werte **DISK_PARALLELISM** werden verwendet.

Aspekte zu großen Indizes

Zur Verbesserung der Leistung bei der Erstellung großer Indizes während einer Ladeoperation kann es sinnvoll sein, den Datenbankkonfigurationsparameter **sortheap** zu optimieren. **sortheap** ordnet die Speichermenge zu, die zum Sortieren der Indexschlüssel während einer Ladeoperation verwendet wird. Beispiel: Damit vom Dienstprogramm LOAD pro Index 4000 Hauptspeicherseiten zum Sortieren der Schlüssel verwendet werden, muss der Datenbankkonfigurationsparameter **sortheap** auf `4000` Seiten gesetzt werden. Dann müssen alle Anwendungen von der Datenbank getrennt und anschließend muss der Befehl **LOAD** aufgerufen werden.

Wenn ein Index so umfangreich ist, dass er nicht im Hauptspeicher sortiert werden kann, tritt ein *Sortierüberlauf* ein. Dies bedeutet, dass die Daten auf mehrere "Sortierläufe" aufgeteilt und in einem Tabellenbereich für temporäre Tabellen gespeichert werden, der später zusammengeführt wird. Verwenden Sie das Monitorelement **sort_overflows**, um festzustellen, ob ein Sortierüberlauf stattgefunden hat. Besteht keine Möglichkeit, einen Sortierüberlauf durch ein Heraufsetzen des Parameters **sortheap** zu verhindern, muss der Pufferpool für Tabellenbereiche für temporäre Tabellen unbedingt groß genug sein, um das Volumen der durch den Überlauf verursachten Platten-E/A zu minimieren. Es empfiehlt sich des Weiteren, Tabellenbereiche für temporäre Tabellen mit mehreren Containern zu deklarieren, die sich jeweils auf unterschiedlichen Platteneinheiten befinden, damit während der Zusammenführung der Sortierläufe eine E/A-Parallelität erzielt wird. Ist für eine Tabelle mehr als ein Index definiert, steigt die Speicherbelegung proportional an, da die Ladeoperation alle Schlüssel im Speicher hält.

Verzögern der Indexerstellung

Im Allgemeinen ist es effizienter, die Indexerstellung während der Ladeoperation zuzulassen, indem entweder der Modus `REBUILD` oder `INCREMENTAL` angegeben wird, anstatt die Indexerstellung zu verzögern. Wie in Abb. 11 auf Seite 85 dargestellt werden Tabellen normalerweise in drei Schritten erstellt: Laden von Daten, Erstellen von Indizes und Erfassen von Statistikdaten. Das führt zu mehrfacher Daten-E/A während der Ladeoperation, während der Indexerstellung (es sind mehrere Indizes pro Tabelle möglich) und während der Sammlung von Statistikdaten (die zu E/A bei den Tabellendaten und allen Indizes führt). Eine viel schnellere Methode besteht darin, alle diese Aufgaben vom Dienstprogramm LOAD in einem Datendurchlauf ausführen zu lassen. Es sollte hierbei allerdings beachtet werden, dass eindeutige Indizes die Leistung von LOAD verringern, wenn Indizes doppelt vorhanden sind.

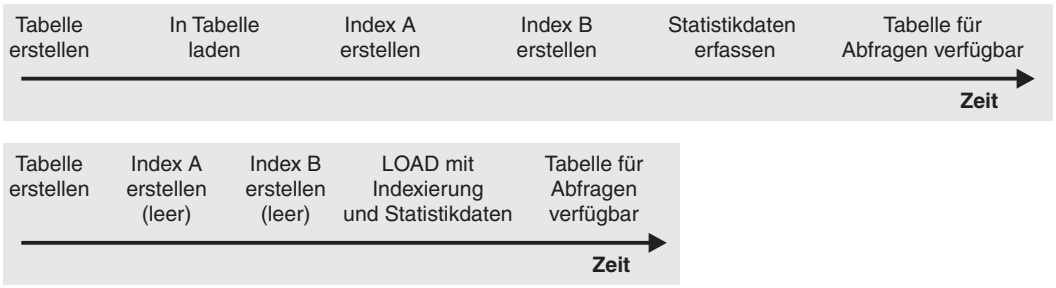


Abbildung 11. Erhöhen der Leistung von LOAD durch gleichzeitige Indexierung und Erfassung von Statistikdaten. Tabellen werden normalerweise in drei Schritten erstellt: Laden von Daten, Erstellen von Indizes und Erfassen Statistikdaten. Das führt zu mehrfacher Daten-E/A während der Ladeoperation, während der Indexerstellung (es sind mehrere Indizes pro Tabelle möglich) und während der Sammlung von Statistikdaten (die zu E/A bei den Tabellendaten und allen Indizes führt). Eine viel schnellere Methode ist, alle diese Aufgaben vom Dienstprogramm LOAD in einem Datendurchlauf ausführen zu lassen.

In bestimmten Fällen kann die Verzögerung der Indexerstellung und der Aufruf der Anweisung CREATE INDEX zur Leistungsverbesserung beitragen. Bei der Sortierung während eines Indexrebuilds werden maximal so viele Seiten verwendet, wie in **sortheap** angegeben wurden. Wird mehr Speicherplatz benötigt, wird der Pufferpool TEMP verwendet, und (schließlich) erfolgt ein Überlauf auf die Platte. Kommt es beim Laden zu einem Überlauf und somit zu einer Leistungsminderung, kann es ratsam sein, das Dienstprogramm **LOAD** mit der Option **INDEXING MODE DEFERRED** auszuführen und den Index später erneut zu erstellen. Mit CREATE INDEX wird jeweils immer nur ein Index erstellt. Dadurch wird beim mehrfachen Durchsuchen der Tabelle zum Erfassen von Schlüsseln die Speicherbelegung reduziert.

Ein weiterer Vorteil der Indexerstellung mit der Anweisung CREATE INDEX gegenüber der parallelen Erstellung mit einer Ladeoperation liegt darin, dass die Anweisung CREATE INDEX zum Sortieren von Schlüsseln mehrere Prozesse, so genannte Threads, verwenden kann. Das Erstellen des Indexes selbst wird nicht parallel ausgeführt.

Erstellen eines Komprimierungswörterverzeichnisses (Compression Dictionary) während einer Ladeoperation

Die Ausführung der Operationen **LOAD INSERT** und **LOAD REPLACE** für Tabellen, für die die Komprimierung aktiviert wurde, kann die Erstellung von Komprimierungswörterverzeichnissen auslösen. Abhängig vom Typ der Zeilenkomprimierung, die von einer Tabelle verwendet wird, wird die Wörterverzeichniserstellung auf unterschiedliche Arten durchgeführt.

Die *klassische Zeilenkomprimierung* verwendet zum Komprimieren von Daten ein einzelnes *Komprimierungswörterverzeichnis auf Tabellenebene*. Die *adaptive Komprimierung* verwendet mehrere *Komprimierungswörterverzeichnisse auf Seitenebene*, um einzelne Datenseiten zusammen mit den Komprimierungswörterverzeichnissen auf Tabellenebene, die in der klassischen Zeilenkomprimierung benutzt werden, zu komprimieren.

Komprimierungswörterverzeichnisse auf Seitenebene

Wörterverzeichnisse auf Seitenebene werden während der Ausführung der Operation **LOAD INSERT** oder **LOAD REPLACE** automatisch erstellt und aktualisiert. Die Optionen **KEEPDICTIONARY** und **RESETDICTIONARY** des Befehls **LOAD** haben keine Auswirkungen auf Wörterverzeichnisse auf Seitenebene.

Komprimierungswörterverzeichnisse auf Tabellenebene

Wörterverzeichnisse auf Tabellenebene werden sowohl für die Operation LOAD INSERT als auch für die Operation LOAD REPLACE automatisch erstellt, wenn kein Wörterverzeichnis vorhanden ist. Wenn jedoch ein Wörterverzeichnis auf Tabellenebene vorhanden ist, wird das Wörterverzeichnis standardmäßig nicht aktualisiert. Dies bedeutet, dass bei der Operation **LOAD REPLACE** standardmäßig die Option **KEEPDICTIONARY** angenommen wird. Sie können die Option **RESETDICTIONARY** angeben, um das vorhandene Wörterverzeichnis auf Tabellenebene zu entfernen und ein neues Wörterverzeichnis zu erstellen.

Die Wirkungsweise von LOAD INSERT richtet sich immer nach dem Verhalten, das von der Option **KEEPDICTIONARY** vorgegeben wird.

Bei der Erstellung von Wörterverzeichnissen auf Tabellenebene für Nicht-XML-Daten verwendet das Dienstprogramm LOAD die in der Zieltabelle vorhandenen Daten zum Erstellen der Wörterverzeichnisse. Dabei wird vorausgesetzt, dass diese bereits vorhandenen Daten repräsentativ für die Art der Daten sind, die in dieser Tabelle gespeichert werden sollen. In den Fällen, in denen in der Zieltabelle nicht genug Daten vorhanden sind, erstellt das Dienstprogramm LOAD die Wörterverzeichnisse, nachdem es ausreichend Eingabedaten geprüft hat. In dieser Situation verwendet das Dienstprogramm LOAD nur die Eingabedaten zur Erstellung des Wörterverzeichnisses.

Bei XML-Daten prüft das Dienstprogramm LOAD lediglich die ankommenden Daten.

Wenn Wörterverzeichnisse für bereichspartitionierte Tabellen erstellt werden, wird jede Partition wie eine einzelne Tabelle behandelt. Es werden keine partitionsübergreifenden Wörterverzeichnisse erstellt und die Wörterverzeichniserstellung erfolgt nicht in Partitionen, die bereits Wörterverzeichnisse enthalten. Bei Tabellendaten basiert das Wörterverzeichnis, das für jede Partition erstellt wird, ausschließlich auf den in der betreffenden Partition bereits vorhandenen Tabellendaten (und gegebenenfalls den geladenen Daten). Wenn die in einer Partition bereits vorhandenen Daten unter dem Mindestschwellenwert liegen, wird ab Version 9.7 Fixpack 1 das Wörterverzeichnis ausschließlich auf der Basis der geladenen Daten generiert. Bei XML-Daten basiert das für jede Partition erstellte Wörterverzeichnis auf den Daten, die in die betreffende Partition geladen werden.

LOAD REPLACE mit der Option KEEPDICTIONARY

Eine **LOAD REPLACE**-Operation, die die Option **KEEPDICTIONARY** verwendet, behält die vorhandenen Wörterverzeichnisse bei und verwendet sie zur Komprimierung der geladenen Daten, sofern für die Zieltabelle das Attribut COMPRESS aktiviert ist. Wenn keine Wörterverzeichnisse vorhanden sind, generiert das Dienstprogramm LOAD für Tabellen mit aktiviertem Attribut COMPRESS neue Wörterverzeichnisse (unter der Voraussetzung, dass die in die Tabelle geladenen Daten einen zuvor festgelegten Schwellenwert für Tabellenzeilen oder für XML-Dokumente im standardmäßigen XML-Speicherobjekt überschreiten). Da die Daten der Zieltabelle ersetzt werden, verwendet das Dienstprogramm LOAD zum Erstellen der Wörterverzeichnisse lediglich die Eingabedaten. Nach Erstellung eines Wörterverzeichnisses wird dieses in die Tabelle eingefügt, und die Ladeoperation wird fortgesetzt.

LOAD REPLACE mit der Option RESETDICTIONARY

Es sind zwei Schlüsselauswirkungen für die Verwendung der Option **RESETDICTIONARY** zu erwarten, wenn Daten in eine Tabelle geladen werden, deren COMPRESS-Attribut aktiviert ist. Erstens findet die Erstellung von Datenverzeichnissen statt, solange ein beliebiges Datenvolumen in der Zieltabelle vorhanden ist, nachdem die **LOAD REPLACE**-Operation abgeschlossen ist. Mit anderen Worten: Die neuen Komprimierungswörterverzeichnisse können auch auf einer einzigen Zeile von Daten oder einem einzigen XML-Dokument beruhen. Zweitens werden die vorhandenen Wörterverzeichnisse gelöscht aber nicht ersetzt (die Zieltabelle verfügt also über keine Komprimierungswörterverzeichnisse mehr), falls eine der folgenden Situationen zutrifft:

- Die Operation wird für eine Tabelle mit inaktiviertem Attribut COMPRESS ausgeführt
- Es wurde nichts geladen (Nullzeilen), was bedeutet, dass ADM5591W im Benachrichtigungsprotokoll erfasst wird

Anmerkung: Wenn Sie eine **LOAD TERMINATE**-Operation nach einer **LOAD REPLACE**-Operation mit der Option **RESETDICTIONARY** durchführen, werden eventuell vorhandene Komprimierungswörterverzeichnisse gelöscht und nicht ersetzt.

Leistungseinfluss

Das Erstellen von Wörterverzeichnissen wirkt sich auf folgende Weise auf die Leistung einer Ladeoperation aus:

- Bei **LOAD INSERT**-Operationen werden sämtliche bereits vorhandenen Tabellendaten (und nicht nur das im Mindestschwellenwert für die Wörterverzeichniserstellung angegebene Volumen) vor der Erstellung des Komprimierungswörterverzeichnisses auf Tabellenebene überprüft. Daher steigt die für diese Überprüfung benötigte Zeit mit der Tabellengröße an.
- Zur Erstellung der Komprimierungswörterverzeichnisse fällt zusätzlicher Verarbeitungsaufwand an, obwohl der Zeitaufwand für die Wörterverzeichniserstellung minimal ist.

Obwohl bestimmte Operationen in Bezug auf die Wörterverzeichniserstellung Auswirkungen auf die CPU-Auslastung durch den Befehl **LOAD** haben können, sind Ladeoperationen normalerweise E/A-orientiert. Dies bedeutet, dass ein hoher Anteil der Zeitdauer, die für das Warten auf den Abschluss des Ladevorgangs benötigt wird, darauf zurückzuführen ist, dass auf das Schreiben von Daten auf die Platte gewartet werden muss. Die erhöhte CPU-Auslastung, die durch die Wörterverzeichniserstellung verursacht wird, führt normalerweise nicht zu einer Erhöhung der Zeitdauer, die für den Ladevorgang benötigt wird. Vielmehr ist es so, dass aufgrund der Tatsache, dass die Daten in einem komprimierten Format geschrieben werden, die E/A-Zeiten stattdessen im Vergleich zum Laden von Daten in nicht komprimierte Tabellen reduziert werden können.

Optionen zur Verbesserung der Ladeleistung

Es gibt verschiedene Befehlsparameter, die Sie zum Optimieren der Ladeleistung verwenden können. Darüber hinaus steht eine Reihe von Änderungswerten für Dateitypen zur Verfügung, die speziell für das Dienstprogramm LOAD konzipiert sind und in bestimmten Fällen die Leistung dieses Dienstprogramms erheblich steigern können.

Befehlsparameter

Das Dienstprogramm LOAD liefert die bestmögliche Leistung, indem optimale Werte für DISK_PARALLELISM, CPU_PARALLELISM und DATA BUFFER ermittelt werden, wenn diese Parameter nicht vom Benutzer angegeben wurden. Die Optimierung erfolgt auf der Grundlage der Größe und des freien Speicherbereichs, der im Zwischenspeicher der Dienstprogramme verfügbar ist. Bevor Sie versuchen, die Werte dieser Parameter für Ihre Anforderungen zu optimieren, sollten Sie prüfen, ob nicht die autonomen Einstellungen von DISK_PARALLELISM und CPU_PARALLELISM für Ihr System gut geeignet sind.

Die folgenden Informationen beziehen sich auf die Auswirkungen der verschiedenen, beim Dienstprogramm LOAD verfügbaren Optionen auf die Leistung:

ALLOW READ ACCESS
Mithilfe dieser Option können Sie eine Tabelle abfragen, während eine Ladeoperation ausgeführt wird. Sie können nur die Daten anzeigen, die vor der Ladeoperation bereits in der Tabelle vorhanden waren. Wenn die Option INDEXING MODE INCREMENTAL ebenfalls angegeben wird und die Ladeoperation fehlschlägt, muss die nachfolgende LOAD TERMINATE-Operation möglicherweise Inkonsistenzen im Index korrigieren. Hierzu ist eine Indexsuche erforderlich, die mit einem erheblichen Aufwand an E/A-Operationen verbunden ist. Wird die Option ALLOW READ ACCESS auch für die LOAD TERMINATE-Operation angegeben, wird für die Ein-/Ausgabe der Pufferpool verwendet.

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 der Veröffentlichung *Neuerungen in DB2 Version 10.1*.

COPY YES oder NO
Verwenden Sie diesen Parameter, um anzugeben, ob während einer Ladeoperation eine Kopie der Eingabedaten erstellt werden soll. COPY YES kann nur verwendet werden, wenn die aktualisierende Recovery aktiviert wurde. Diese Option verringert die Leistung von LOAD, weil alle Ladedaten während der Ladeoperation kopiert werden. Der Anstieg der E/A-Aktivität kann unter Umständen zu einer längeren Ladezeit auf einem E/A-abhängigen System führen. Das Angeben mehrerer Einheiten oder Verzeichnisse (auf verschiedenen Platten) kann die durch diese Operation verursachte Leistungseinbuße ausgleichen. COPY NO kann nur verwendet werden, wenn die aktualisierende Recovery aktiviert wurde. Diese Option hat keine Auswirkungen auf die Leistung von LOAD. Allerdings werden alle Tabellenbereiche, die der geladenen Tabelle zugeordnet sind, in den Status "Backup anstehend" versetzt und müssen gesichert werden, bevor auf die Tabelle zugegriffen werden kann.

CPU_PARALLELISM
Mit diesem Parameter können Sie die Anzahl der pro Datenbankpartition ausgeführten Prozesse ausschöpfen (wenn dies von Ihrer Maschine unterstützt wird) und die Leistung von LOAD signifikant verbessern. Mit diesem Parameter wird die Anzahl von Prozessen bzw. Threads angegeben, die vom Dienstprogramm LOAD zur Syntaxanalyse, Konvertierung und Formatierung von Datensätzen verwendet werden. Der maximal zulässige Wert ist 30. Wenn für den angegebenen Wert zu wenig Speicher verfügbar ist, wird der Wert vom Dienstprogramm angepasst. Wenn dieser Parameter

nicht angegeben wird, wählt das Dienstprogramm LOAD einen Standardwert auf Grundlage der Anzahl CPUs auf dem System aus.

Die Reihenfolge der Datensätze in den Quellendaten (siehe Abb. 12) bleibt unabhängig vom Wert dieses Parameters erhalten, wenn Folgendes gilt:

- Der Änderungswert `anyorder` für den Datentyp ist nicht angegeben.
- Die Option PARTITIONING_DBPARTNUMS ist nicht angegeben (und für die Partitionierung werden mehrere Partitionen verwendet).

Wenn Tabellen Daten des Typs LOB oder LONG VARCHAR enthalten, wird `CPU_PARALLELISM` auf den Wert 1 gesetzt. In diesem Fall wird die Parallelität nicht unterstützt.

Die Verwendung dieses Parameters ist zwar nicht auf SMP-Hardware begrenzt, doch wird bei Verwendung in Nicht-SMP-Umgebungen möglicherweise keine erkennbare Leistungsverbesserung erzielt.

Abbildung 12. Beibehaltung der Satzreihenfolge der Quellendaten bei Ausnutzung der Anzahl von pro Datenbankpartition ausgeführten Prozessen bei einer Ladeoperation

DATA BUFFER

Mit dem Parameter `DATA BUFFER` wird die Gesamtmenge des Speichers (in Einheiten von jeweils 4 KB) angegeben, die dem Dienstprogramm LOAD als Puffer zugeordnet wird. Es wird empfohlen, die Größe dieses Puffers auf mehrere *Speicherbereiche* (Extents) einzustellen. Der Datenpuffer wird aus dem Zwischenspeicher der Dienstprogramme zugeordnet; er kann jedoch die Einstellung für den Datenbankkonfigurationsparameter *util_heap_sz* überschreiten, solange im System ausreichend Speicher verfügbar ist.

DISK_PARALLELISM

Mit dem Parameter `DISK_PARALLELISM` wird die Anzahl von Prozessen bzw. Threads angegeben, die vom Dienstprogramm LOAD zum Schreiben von Datensätzen auf die Platte verwendet werden. Mit diesem Parameter können Sie verfügbare Container beim Laden von Daten ausschöpfen und die Leistung von LOAD signifikant verbessern. Der maximal zulässige Wert entspricht entweder dem Vierfachen des Wertes von `CPU_PARALLELISM` (der vom Dienstprogramm LOAD tatsächlich verwendet wird) oder beträgt 50. Der höhere dieser beiden Werte wird verwendet. `DISK_PARALLELISM` entspricht standardmäßig der Summe der Tabellenbereichscontainer in allen Tabellenbereichen, die Objekte für die zu ladende Tabelle enthalten, außer wenn dieser Wert den maximal zulässigen Wert übersteigt.

NONRECOVERABLE

Wenn die aktualisierende Recovery aktiviert wurde, verwenden Sie diesen Parameter, wenn LOAD-Transaktionen für eine Tabelle nicht wiederherstellbar sein müssen, wenn eine aktualisierende Recovery durchgeführt wird. Eine Ladeoperation mit `NONRECOVERABLE` und mit `COPY NO` weisen die gleiche Leistung auf. Allerdings besteht ein erheblicher Unterschied in Bezug auf das Risiko eines möglichen Datenverlustes. Eine Ladeoperation mit `NONRECOVERABLE` markiert eine Tabelle als mit einer aktualisierenden Recovery nicht wiederherstellbar, der Zugriff auf die Tabelle wird jedoch nicht eingeschränkt. Dies kann zu Problemen führen, wenn eine aktualisierende Recovery der Ladeoperation ausgeführt werden muss. In diesem Fall ge-

hen sowohl die geladenen Daten als auch alle nachfolgend durchgeführten Aktualisierungen an der Tabelle verloren. Eine mit COPY NO durchgeführte Ladeoperation versetzt alle abhängigen Tabellenbereiche in den Status "Backup anstehend", durch den nicht mehr auf die Tabelle zugegriffen werden kann, bis der Backup durchgeführt wurde. Da nach Ausführung dieses Ladeoperationstyps ein Backup erstellt werden muss, riskieren Sie nicht den Verlust der geladenen Daten oder der nachfolgenden Aktualisierungen an der Tabelle. Dies bedeutet also, dass eine Ladeoperation mit COPY NO vollständig wiederherstellbar ist.

Anmerkung: Wenn diese LOAD-Transaktionen bei nachfolgenden Restores und aktualisierenden Recoverys vorgefunden werden, wird die Tabelle nicht aktualisiert und als ungültig (invalid) gekennzeichnet. Weitere Aktionen für diese Tabelle werden ignoriert. Nach Fertigstellung der aktualisierenden Recovery kann die Tabelle nur gelöscht werden.

SAVECOUNT

Verwenden Sie diesen Parameter, um ein Intervall für die Festlegung von Konsistenzpunkten **während der LOAD-Phase** einer Ladeoperation einzustellen. Die Synchronisation der Vorgänge zum Definieren eines Konsistenzpunkts dauert einige Zeit. Wenn dies zu häufig geschieht, sinkt die Leistung von LOAD merklich. Wenn eine sehr große Zahl von Zeilen geladen werden muss, sollte ein hoher Wert für SAVECOUNT angegeben werden (z. B. bei einer Ladeoperation mit 100 Millionen Sätzen ein Wert von 10 Millionen).

Eine LOAD RESTART-Operation wird automatisch vom letzten Konsistenzpunkt aus fortgesetzt. Dies gilt allerdings nur dann, wenn die LOAD RESTART-Operation bei der LOAD-Phase wieder aufgenommen wird.

STATISTICS USE PROFILE

Erfasst Statistiken, die im Tabellenstatistikprofil angegeben sind. Mit diesem Parameter können Sie Datenverteilungs- und Indexstatistikdaten effizienter erfassen als durch den Aufruf des Dienstprogramms RUNSTATS nach Fertigstellung der Ladeoperation. Die Leistung der Ladeoperation selbst wird bei Verwendung dieses Parameters allerdings verringert. (Dies gilt insbesondere dann, wenn DETAILED INDEXES ALL angegeben ist.)

Die Leistungsstärke der Anwendungen ist am höchsten, wenn sie die bestmöglichen Verteilungs- und Indexstatistikdaten verwenden. Nach der Aktualisierung der Statistikdaten können die Anwendungen die neuen Zugriffspfade zu den Tabellendaten auf der Grundlage der letzten Statistikdaten verwenden. Neue Zugriffspfade zu einer Tabelle können erstellt werden, indem Sie die Anwendungspakete mit dem Befehl **BIND** erneut binden. Das Tabellenstatistikprofil wird erstellt, indem der Befehl **RUNSTATS** mit den SET PROFILE-Optionen ausgeführt wird.

Beim Laden von Daten in große Tabellen empfiehlt es sich, für den Datenbankkonfigurationsparameter *stat_heap_sz* (Größe des Statistikzwischenspeichers) einen höheren Wert anzugeben.

USE <tabellenbereichsname>

Bei Ausführung einer Ladeoperation mit ALLOW READ ACCESS, für die der Indexierungsmodus REBUILD verwendet wird, erlaubt dieser Parameter die erneute Erstellung eines Indexes in einem Tabellenbereich für temporäre Systemtabellen und das Zurückkopieren dieses Indexes in den Indextabellenbereich während der INDEX COPY-Phase einer Ladeoperation.

Standardmäßig wird der vollständig neu erstellte Index (der auch als *Schattenindex* bezeichnet wird) im selben Tabellenbereich wie der ursprüngliche Index erstellt. Dies kann unter Umständen zu Ressourcenproblemen führen, da sich sowohl der ursprüngliche Index als auch der Schattenindex gleichzeitig in demselben Tabellenbereich befinden. Falls der Schattenindex in demselben Tabellenbereich wie der ursprüngliche Index erstellt wird, wird der ursprüngliche Index sofort durch den Schattenindex ersetzt. Wird der Schattenindex jedoch in einem Tabellenbereich für temporäre Systemtabellen erstellt, ist bei der Ladeoperation eine INDEX COPY-Phase erforderlich, die den Index aus einem Tabellenbereich für temporäre Systemtabellen in den Tabellenbereich für den Index kopiert. Der Kopiervorgang umfasst umfangreiche E/A-Aktionen. Wenn es sich bei einem der Tabellenbereiche um einen DMS-Tabellenbereich handelt, kann die Ein-/Ausgabe im Tabellenbereich für temporäre Systemtabellen möglicherweise nicht sequenziell erfolgen. Während der INDEX COPY-Phase werden die in der Option `DISK_PARALLELISM` angegebenen Werte berücksichtigt.

WARNINGCOUNT

Verwenden Sie diesen Parameter, um die Anzahl von Warnungen anzugeben, die vom Dienstprogramm zurückgegeben werden kann, bevor eine Ladeoperation beendet wird. Geben Sie für den Parameter `WARNINGCOUNT` einen relativ niedrigen numerischen Wert an, wenn Sie lediglich einige wenige oder gar keine Warnungen erwarten. Die Ladeoperation wird nach Erreichen des Wertes für `WARNINGCOUNT` gestoppt. Damit haben Sie die Möglichkeit, die Daten zu korrigieren, bevor Sie versuchen, die Ladeoperation abzuschließen.

Änderungswerte für Dateityp

ANYORDER

Standardmäßig behält das Dienstprogramm LOAD die Satzreihenfolge der Quellendaten bei. Wenn das Dienstprogramm LOAD in einer SMP-Umgebung eingesetzt wird, dann müssen die Parallelverarbeitungsoperationen synchronisiert werden, um die Beibehaltung der korrekten Reihenfolge zu gewährleisten.

In einer SMP-Umgebung wird das Dienstprogramm LOAD mit dem Änderungswert `anyorder` für den Dateityp angewiesen, die Reihenfolge nicht beizubehalten. Hierdurch lässt sich die Effizienz verbessern, da die zur Beibehaltung der Reihenfolge erforderliche Synchronisation vermieden werden kann. Wenn die zu ladenden Daten jedoch bereits vorsortiert sind, beschädigt die Angabe von `anyorder` möglicherweise die Vorsortierung, und die Vorteile der Vorsortierung gehen für nachfolgende Abfragen verloren.

Anmerkung: Der Änderungswert `anyorder` für den Dateityp ist ohne Wirkung, wenn für `CPU_PARALLELISM` der Wert 1 definiert ist. Außerdem besteht keine Kompatibilität mit der Option `SAVECOUNT`.

BINARYNUMERICS, ZONEDDECIMAL und PACKEDDECIMAL

Für ASCII-Quellendaten mit fester Länge und mit universellen Zeilenbegrenzern (ASC) kann es bei der Darstellung numerischer Daten im Binärformat bei der Ausführung von Ladeoperationen zu einer Verbesserung der Leistung kommen. Wenn der Änderungswert `packeddecimal` für den Dateityp angegeben wird, werden Dezimaldaten vom Dienstprogramm LOAD als im gepackten Dezimalformat (zwei Stellen pro Byte) vorliegend interpretiert. Wenn der Änderungswert `zoneddecimal` für den Dateityp angegeben ist, werden Dezimaldaten vom Dienstprogramm LOAD als im ge-

zonten Dezimalformat (eine Stellen pro Byte) vorliegend interpretiert. Bei allen anderen numerischen Typen werden die Daten bei Angabe des Änderungswertes `binarynumerics` für den Dateityp vom Dienstprogramm LOAD als im Binärformat vorliegend interpretiert.

Anmerkung:

- Wenn der Änderungswert `binarynumerics`, `packeddecimal` oder `zoneddecimal` für den Dateityp angegeben ist, werden numerische Daten unabhängig von der verwendeten Plattform als im Big Endian-Format (höherwertiges Byte zuerst) vorliegend interpretiert.
- Der Änderungswert `packeddecimal` für den Dateityp und der Änderungswert `zoneddecimal` für den Dateityp schließen sich gegenseitig aus.
- Der Änderungswert `packeddecimal` für den Dateityp und der Änderungswert `zoneddecimal` für den Dateityp gelten nur für die dezimalen Zielspalten. Die binären Daten müssen hierbei mit den Definitionen der Zielspalten übereinstimmen.
- Der Änderungswert `reclen` für den Dateityp muss angegeben werden, wenn der Änderungswert `binarynumerics`, `packeddecimal` oder `zoneddecimal` für den Dateityp angegeben ist.

FASTPARSE

Diese Werte sollten mit Bedacht verwendet werden. In bestimmten Situationen, in denen bekannt ist, dass die zu ladenden Daten gültig sind, ist es eventuell nicht erforderlich, dass das Dienstprogramm LOAD die Syntaxprüfung im selben Umfang durchführt wie bei Daten, die als fehlerverdächtiger eingestuft werden. Durch eine Reduzierung des Geltungsbereichs dieses Schrittes kann die Leistung des Dienstprogramms LOAD um ca. 10 - 20 Prozent gesteigert werden. Hierzu kann der Änderungswert `fastparse` für den Dateityp angegeben werden, mit dem der Aufwand für die Datenprüfung in vom Benutzer angegebenen Spaltenwerten aus ASC- und DEL-Dateien reduziert werden kann.

NOROWWARNINGS

Während der Ausführung einer Ladeoperation werden Warnungen zu zurückgewiesenen Zeilen in eine angegebene Datei geschrieben. Wenn vom Dienstprogramm LOAD jedoch eine große Anzahl zurückgewiesener, ungültiger oder abgeschnittener Datensätze verarbeitet werden muss, kann sich dies negativ auf die Leistung von LOAD auswirken. In Situationen, in denen mit zahlreichen Warnungen zu rechnen ist, können Sie den Änderungswert `norowwarnings` für den Dateityp verwenden, um die Aufzeichnung dieser Warnungen zu unterdrücken.

PAGEFREESPACE, INDEXFREESPACE und TOTALFREESPACE

Durch Dateneinfügungs- und Datenaktualisierungsoperationen in Tabellen steigt mit der Zeit die Notwendigkeit zur Tabellen- und Indexreorganisation. Eine Lösung für dieses Problem besteht in der Erhöhung des freien Speicherbereichs für Tabellen und Indizes anhand von `pagefreespace`, `indexfreespace` und `totalfreespace`. Die ersten beiden Änderungswerte, die Priorität gegenüber dem Wert für PCTFREE besitzen, geben den Prozentsatz der Daten- und Indexseiten an, der als freier Speicherbereich verbleiben soll. Demgegenüber gibt `totalfreespace` den Prozentsatz der Gesamtseiten an, der als freier Speicherbereich am Ende der Tabelle angefügt werden soll.

Ladefunktionen zur Verwaltung der referenziellen Integrität

Zwar ist das Dienstprogramm LOAD normalerweise effizienter als das Dienstprogramm IMPORT, es erfordert jedoch eine Anzahl von Funktionen, um die referenzielle Integrität der geladenen Informationen sicherzustellen:

- **Tabellensperren** für die Steuerung des gemeinsamen Zugriffs und zur Verhinderung unkontrollierten Datenzugriffs während einer Ladeoperation
- **Tabellenstatus** und **Tabellenbereichsstatus**, über die entweder der Zugriff auf Daten gesteuert oder besondere Benutzeraktionen sondiert werden können
- **Ladeausnahmetabellen**, durch die sichergestellt wird, dass Zeilen mit ungültigen Daten nicht einfach ohne Ihr Wissen gelöscht werden

Überprüfen auf ungültige Integritätsbedingungen nach einer Ladeoperation

Nach einer Ladeoperation kann sich die geladene Tabelle unter Umständen im Status "Festlegen der Integrität anstehend" (im Modus READ für "Lesezugriff" oder NO ACCESS für "Kein Zugriff") befinden, wenn eine der folgenden Bedingungen zutrifft:

- Für die Tabelle sind Prüfungen auf Integritätsbedingungen oder referenzielle Integritätsbedingungen definiert.
- Die Tabelle enthält generierte Spalten, und die Ladeoperation wurde mit einem Client der Version 7 oder früher eingeleitet.
- Der Tabelle sind IMQTs (Immediate Materialized Query Tables - sofort gespeicherte Abfragetabellen) oder sofort gespeicherte Zwischenspeichertabellen untergeordnet, die auf sie verweisen.
- Die Tabelle ist eine Zwischenspeichertabelle oder eine MQT (Materialized Query Table, gespeicherte Abfragetabelle).

Die Markierung `STATUS` des Eintrags für die geladene Tabelle in `SYSCAT.TABLES` gibt den Status "Festlegen der Integrität anstehend" der Tabelle an. Damit die geladene Tabelle vollumfänglich verwendbar ist, muss für `STATUS` der Wert `N` und für `ACCESS MODE` der Wert `F` eingetragen sein. Diese Werte geben an, dass der Zugriff auf die Tabelle vollständig möglich ist und dass die Tabelle einen normalen Status aufweist.

Falls der geladenen Tabelle andere Tabellen untergeordnet sind, kann mit dem Parameter SET INTEGRITY PENDING CASCADE angegeben werden, ob der Status "Festlegen der Integrität anstehend" der geladenen Tabelle direkt an die untergeordneten Tabellen weitergegeben werden soll oder nicht.

Wenn für die Tabelle sowohl Integritätsbedingungen als auch untergeordnete Fremdschlüsseltabellen, abhängige MQTs sowie abhängige Zwischenspeichertabellen definiert sind und alle Tabellen vor der Ladeoperation einen normalen Status aufweisen, entstehen - je nach angegebenen Ladeparametern - die folgenden Ergebnisse:

INSERT, ALLOW READ ACCESS und SET INTEGRITY PENDING CASCADE IMMEDIATE

Die geladene Tabelle, ihre abhängigen MQTs und ihre abhängigen Zwischenspeichertabellen werden in den Status "Festlegen der Integrität anstehend" mit Lesezugriff versetzt.

INSERT, ALLOW READ ACCESS und SET INTEGRITY PENDING CASCADE DEFERRED

Nur die geladene Tabelle wird in den Status "Festlegen der Integrität anste-

hend" mit Lesezugriff versetzt. Untergeordnete Fremdschlüsseltabellen, untergeordnete MQTs und untergeordnete Zwischenspeichertabellen behalten ihren ursprünglichen Status bei.

INSERT, ALLOW NO ACCESS und SET INTEGRITY PENDING CASCADE IMMEDIATE

Die geladene Tabelle, ihre abhängigen MQTs und ihre abhängigen Zwischenspeichertabellen werden in den Status "Festlegen der Integrität anstehend, kein Zugriff" versetzt.

INSERT oder REPLACE, ALLOW NO ACCESS und SET INTEGRITY PENDING CASCADE DEFERRED

Nur die geladene Tabelle wird in den Status "Festlegen der Integrität anstehend, kein Zugriff" versetzt. Untergeordnete Fremdschlüsseltabellen, untergeordnete IMQTs und untergeordnete sofort gespeicherte Zwischenspeichertabellen behalten ihren ursprünglichen Status bei.

REPLACE, ALLOW NO ACCESS und SET INTEGRITY PENDING CASCADE IMMEDIATE

Die Tabelle und alle ihre untergeordneten Fremdschlüsseltabellen, untergeordneten IMQTs und untergeordneten sofort gespeicherten Zwischenspeichertabellen werden in den Status "Festlegen der Integrität anstehend, kein Zugriff" versetzt.

Anmerkung: Die Angabe der Option ALLOW READ ACCESS in einer LOAD REPLACE-Operation führt zu einem Fehler.

Verwenden Sie die Anweisung SET INTEGRITY, um den Status "Festlegen der Integrität anstehend" zu entfernen. Mit der Anweisung SET INTEGRITY wird eine Tabelle auf ungültige Integritätsbedingungen überprüft, und die Tabelle wird aus dem Status "Festlegen der Integrität anstehend" herausgenommen. Wenn alle Ladeoperationen im Einfügemodus (INSERT) ausgeführt werden, können die Integritätsbedingungen mit der Anweisung SET INTEGRITY inkrementell verarbeitet werden (das heißt, es wird nur der angefügte Teil der Tabelle auf ungültige Integritätsbedingungen überprüft). Beispiel:

```
db2 load from infile1.ixf of ixf insert into table1
db2 set integrity for table1 immediate checked
```

Nur der angefügte Teil von TABLE1 wird auf ungültige Integritätsbedingungen überprüft. Die Überprüfung nur des angefügten Teils auf ungültige Integritätsbedingungen ist schneller als die Überprüfung der vollständigen Tabelle, vor allem bei einer umfangreichen Tabelle, an die kleine Datenmengen angefügt werden.

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Festlegen der Integrität. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Wenn beim Laden einer Tabelle die Option SET INTEGRITY PENDING CASCADE DEFERRED angegeben wird und die Anweisung SET INTEGRITY zur Überprüfung auf ungültige Integritätsbedingungen eingesetzt wird, werden die untergeordneten Tabellen in den Status "Festlegen der Integrität anstehend, kein Zugriff" versetzt. Sie müssen eine explizite Anforderung absetzen, um diesen Status für die Tabellen aufzuheben.

Wenn beim Laden einer Tabelle mit abhängigen MQTs oder abhängigen Zwischenspeichertabellen die Option INSERT angegeben wird und die Anweisung SET INTEGRITY zur Überprüfung auf ungültige Integritätsbedingungen verwendet wird, wird der Status "Festlegen der Integrität anstehend" für die Tabelle aufgehoben, und die Tabelle wird in den Status "Kein Versetzen von Daten" versetzt. Dies geschieht, um die nachfolgenden inkrementellen Aktualisierungen der abhängigen MQTs und die inkrementelle Weitergabe der abhängigen Zwischenspeichertabellen zu vereinfachen. Im Status "Kein Versetzen von Daten" sind Operationen, die das Versetzen von Zeilen innerhalb der Tabelle bewirken könnten, nicht zulässig.

Sie können den Status "Kein Versetzen von Daten" außer Kraft setzen, indem Sie beim Absetzen der Anweisung SET INTEGRITY die Option FULL ACCESS angeben. Der Zugriff auf die Tabelle ist vollständig möglich. Bei nachfolgenden Anweisungen REFRESH TABLE findet jedoch eine vollständig Neuberechnung der abhängigen MQTs statt, und die abhängigen Zwischenspeichertabellen werden in einen unvollständigen Status versetzt.

Falls die Option ALLOW READ ACCESS für eine Ladeoperation angegeben wird, verbleibt die Tabelle im Status "Lesezugriff", bis die Anweisung SET INTEGRITY zur Überprüfung auf ungültige Integritätsbedingungen verwendet wird. Sobald die Tabelle festgeschrieben wurde, können Anwendungen sie zwar nach Daten abfragen, die vor der Ladeoperation vorhanden waren. Die Anwendungen können die neu geladenen Daten jedoch erst dann erkennen, nachdem die Anweisung SET INTEGRITY abgesetzt wurde.

Vor einer Überprüfung auf ungültige Integritätsbedingungen können mehrere Ladeoperationen für eine Tabelle stattfinden. Wenn alle Ladeoperationen im Modus "Lesezugriff zulassen" (ALLOW READ ACCESS) abgeschlossen werden, stehen für Abfragen nur die Daten zur Verfügung, die vor der ersten Ladeoperation in der Tabelle vorhanden waren.

Es können eine oder mehrere Tabellen in einem einzigen Aufruf dieser Anweisung überprüft werden. Wenn eine abhängige Tabelle separat überprüft werden soll, kann sich die übergeordnete Tabelle nicht im Status "Festlegen der Integrität anstehend" befinden. Andernfalls müssen die übergeordnete Tabelle und die abhängige Tabelle gleichzeitig überprüft werden. In Falle eines Zyklus für referenzielle Integrität müssen alle von diesem Zyklus betroffenen Tabellen in einen einzigen Aufruf der Anweisung SET INTEGRITY aufgenommen werden. Es kann unter Umständen von Vorteil sein, die übergeordnete Tabelle auf ungültige Integritätsbedingungen zu überprüfen, während eine abhängige Tabelle geladen wird. Dies kann nur geschehen, wenn sich die beiden Tabellen nicht im selben Tabellenbereich befinden.

Beim Absetzen der Anweisung SET INTEGRITY können Sie die Option INCREMENTAL angeben, um die inkrementelle Verarbeitung explizit anzufordern. In den meisten Fällen wird diese Option nicht benötigt, weil die DB2-Datenbank die inkrementelle Verarbeitung auswählt. Wenn die inkrementelle Verarbeitung nicht möglich ist, wird automatisch die volle Verarbeitung verwendet. Wenn die Option INCREMENTAL angegeben ist, die inkrementelle Verarbeitung jedoch nicht möglich ist, wird in folgenden Fällen ein Fehler gemeldet:

- Der Tabelle werden neue Integritätsbedingungen hinzugefügt, während sie sich im Status "Festlegen der Integrität anstehend" befindet.
- Eine LOAD REPLACE-Operation wird ausgeführt, oder die Option NOT LOGGED INITIALLY WITH EMPTY TABLE wird aktiviert, nachdem die letzte Überprüfung auf Integritätsbedingungen für die Tabelle stattgefunden hat.

- Für eine übergeordnete Tabelle wird eine LOAD REPLACE-Operation ausgeführt, oder eine solche Tabelle wird nicht inkrementell auf Integrität überprüft.
- Vor einem Upgrade befindet sich die Tabelle im Status "Festlegen der Integrität anstehend". Wenn für die Tabelle zum ersten Mal nach dem Upgrade eine Überprüfung auf Integritätsbedingungen vorgenommen wird, ist eine vollständige Verarbeitung erforderlich.
- Der Tabellenbereich, der die Tabelle oder ihre übergeordnete Tabelle enthält, wird bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt, und die Tabelle sowie ihre übergeordnete Tabelle befinden sich in unterschiedlichen Tabellenbereichen.

Wenn eine Tabelle mindestens einen Wert `W` in der Spalte CONST_CHECKED des Katalogs SYSCAT.TABLES enthält und die Option NOT INCREMENTAL in der Anweisung SET INTEGRITY nicht angegeben ist, wird die Tabelle inkrementell verarbeitet, und die Spalte CONST_CHECKED des Katalogs SYSCAT.TABLES wird mit `U` gekennzeichnet, um anzugeben, dass nicht alle Daten vom System überprüft wurden.

Die Anweisung SET INTEGRITY aktiviert keine DELETE-Trigger als Ergebnis des Löschens von Zeilen, die gegen Integritätsbedingungen verstoßen. Die Trigger sind jedoch aktiv, sobald für die Tabelle der Status "Festlegen der Integrität anstehend" aufgehoben wird. Wenn Sie Daten korrigieren und Zeilen aus der Ausnahmetabelle in die geladene Tabelle einfügen, werden demzufolge auch alle INSERT-Trigger aktiviert, die möglicherweise für die Tabelle definiert sind. Die Implikationen davon sollten berücksichtigt werden. Eine Möglichkeit wäre z. B., den INSERT-Trigger zu löschen, Zeilen aus der Ausnahmetabelle einzufügen und anschließend den INSERT-Trigger erneut zu erstellen.

Sperren von Tabellen während einer Ladeoperation

In den meisten Fällen setzt das Dienstprogramm LOAD Sperren auf Tabellenebene ein, um den Zugriff auf Tabellen einzuschränken. Die Stufe der Sperren richtet sich nach der Phase, in der sich die Ladeoperation befindet und danach, ob bei der Ladeoperation ein Lesezugriff zulässig ist oder nicht.

Eine Ladeoperation im Modus `ALLOW NO ACCESS` verwendet für die Dauer der Ladeoperation eine exklusive Sperre des Typs "Z" (Super Exclusive Lock) für die Tabelle.

Bevor eine Ladeoperation im Modus `ALLOW READ ACCESS` gestartet wird, wartet das Dienstprogramm LOAD, bis alle Anwendungen, die vor der Ladeoperation gestartet wurden, ihre Sperren für die Zieltabelle freigeben. Am Anfang einer Ladeoperation fordert das Dienstprogramm LOAD eine Aktualisierungssperre vom Typ "U-lock" (Update) für die Tabelle an. Diese Sperre wird solange beibehalten, bis die Daten festgeschrieben werden. Sobald das Dienstprogramm LOAD eine Aktualisierungssperre für die Tabelle anfordert, wartet es, bis alle Anwendungen, die vor dem Beginn der Ladeoperation Sperren für die Tabelle gehalten hatten, diese Sperren freigeben. Dies gilt auch dann, wenn es sich hierbei um kompatible Sperren handelt. Dies wird durch ein temporäres Upgrade der Aktualisierungssperre auf eine Sperre von Typ "Z" erreicht, die nicht mit neuen Anforderungen für Tabellensperren für die Zieltabelle in Konflikt steht, solange die angeforderten Sperren mit der Aktualisierungssperre der Ladeoperation kompatibel sind. Das Dienstprogramm LOAD führt beim Commit der Daten ein Upgrade der Sperre auf eine Sperre vom Typ "Z" durch, sodass sich das Commit etwas verzögern kann, während das Dienstprogramm LOAD darauf wartet, dass Anwendungen mit Konflikte verursachenden Sperren beendet werden.

Anmerkung: Während die Ladeoperation darauf wartet, dass die Anwendungen ihre Sperren für die Tabelle vor dem Laden freigeben, kann eine Zeitlimitüberschreitung eintreten. Für die Ladeoperation tritt jedoch keine Zeitlimitüberschreitung ein, während sie auf die Sperre vom Typ "Z" wartet, die für das Commit der Daten erforderlich ist.

Anwendungen mit einen Konflikt verursachenden Sperren
Verwenden Sie die Option LOCK WITH FORCE des Befehls **LOAD**, um eine Beendigung von Anwendungen, die Konflikte verursachende Sperren für die Zieltabelle besitzen, zu erzwingen, damit die Ladeoperation fortgesetzt werden kann. Bevor eine Ladeoperation im Modus ALLOW READ ACCESS fortgesetzt werden kann, wird die Beendigung von Anwendungen, die die folgenden Sperren halten, erzwungen:

- Tabellensperren, die Konflikte mit einer Aktualisierungssperre des Typs "U" (Update Lock) für eine Tabelle verursachen (beispielsweise IMPORT oder INSERT).
- Alle Tabellensperren, die in der COMMIT-Phase der Ladeoperation vorhanden sind.

Anwendungen, die Konflikte verursachende Sperren für die Systemkatalogtabellen besitzen, werden durch das Dienstprogramm LOAD nicht zwingend beendet. Wenn die Beendigung einer Anwendung auf dem System durch das Dienstprogramm LOAD erzwungen wird, geht ihre Datenbankverbindung verloren, und das System gibt einen Fehler (SQL1224N) zurück.

Wird die Option COPY NO bei einer Ladeoperation für eine wiederherstellbare Datenbank angegeben, werden alle Objekte im Zieltabellenbereich im Modus für die gemeinsame Benutzung (SHARE) gesperrt, bevor der Tabellenbereich in den Status "Backup anstehend" versetzt wird. Dieser Vorgang erfolgt unabhängig vom Zugriffsmodus. Wenn die Option LOCK WITH FORCE angegeben wird, wird die Beendigung aller Anwendungen erzwungen, die Sperren für Objekte im Tabellenbereich besitzen, die Konflikte mit einer gemeinsam benutzten Sperre des Typs "S" (Share Lock) verursachen.

Ladeoperationen mit Lesezugriff

Das Dienstprogramm LOAD bietet zwei Optionen, mit denen der Umfang des Zugriffs gesteuert werden kann, der anderen Anwendungen auf eine geladene Tabelle eingeräumt wird. Die Option ALLOW NO ACCESS sperrt die Tabelle exklusiv und ermöglicht keinen Zugriff auf die Tabellendaten, während die Tabelle geladen wird.

ALLOW NO ACCESS ist das Standardverhalten. Die Option ALLOW READ ACCESS hingegen verhindert jeglichen Schreibzugriff auf die Tabelle durch andere Anwendungen, lässt jedoch einen Lesezugriff auf bereits vorhandene Daten zu. Der folgende Abschnitt behandelt die Option ALLOW READ ACCESS.

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 der Veröffentlichung *Neuerungen in DB2 Version 10.1*.

Tabellendaten und Indexdaten, die vor dem Beginn einer Ladeoperation vorhanden waren, stehen für Abfragen zur Verfügung, während die Ladeoperation läuft. Betrachten Sie das folgende Beispiel:

1. Erstellen Sie eine Spalte mit einer Spalte des Typs INTEGER:

```
create table ED (ed int)
```

2. Laden Sie drei Zeilen:

```
   load from File1 of del insert into ED
...
   Anzahl gelesener Zeilen         = 3
   Anzahl übersprungener Zeilen     = 0
   Anzahl geladener Zeilen         = 3
   Anzahl zurückgewiesener Zeilen   = 0
   Anzahl gelöschter Zeilen         = 0
   Anzahl festgeschriebener Zeilen = 3
```

3. Führen Sie eine Abfrage der Tabelle aus:

```
   select * from ED

   ED
-----------
          1

          2
          3


  3 Satz/Sätze ausgewählt.
```

4. Führen Sie eine Ladeoperation mit der Option ALLOW READ ACCESS aus, und laden Sie zwei weitere Datenzeilen:

```
load from File2 of del insert into ED allow read access
```

5. Gleichzeitig fragen Sie die Tabelle über eine andere Verbindung ab, während die Ladeoperation läuft:

```
   select * from ED

   ED
-----------
          1

          2
          3


  3 Satz/Sätze ausgewählt.
```

6. Warten Sie, bis die Ladeoperation abgeschlossen ist, und fragen Sie anschließend die Tabelle ab:

```
   select * from ED

   ED
-----------
          1

          2
          3

          4
          5

      5 Satz/Sätze ausgewählt.
```

Die Option ALLOW READ ACCESS ist sehr nützlich, wenn große Datenmengen geladen werden, weil sie den Benutzern jederzeit - sogar während laufender oder nach fehlgeschlagenen Ladeoperationen - den Zugriff auf die Tabellendaten ermöglicht. Das Verhalten einer Ladeoperation im Modus ALLOW READ ACCESS ist unabhängig von der Isolationsstufe der Anwendung. Dies bedeutet, dass Eingabeprogramme mit einer beliebigen Isolationsstufe die bereits vorhandenen Daten jederzeit lesen können, jedoch nicht in der Lage sind, die neu geladenen Daten zu lesen, bevor die Ladeoperation abgeschlossen ist.

Der Lesezugriff ist während der gesamten Ladeoperation möglich. Eine Ausnahme bilden hier lediglich der Anfang und das Ende der Operation.

Erster Fall: Die Ladeoperation fordert eine spezielle exklusive Sperre vom Typ "Z" für eine kurze Dauer gegen Ende ihrer SETUP-Phase an. Wenn eine Anwendung eine inkompatible Sperre für die Tabelle hält, bevor die Ladeoperation diese spezielle Z-Sperre anfordert, wartet die Ladeoperation eine bestimmte Zeit darauf, dass diese inkompatible Sperre freigegeben wird, bevor sie aufgrund einer Zeitlimitüberschreitung fehlschlägt. Die entsprechende Zeitdauer wird mit dem Datenbankkonfigurationsparameter *locktimeout* festgelegt. Bei Angabe der Option `LOCK WITH FORCE` erzwingt die Ladeoperation die Beendigung anderer Anwendungen, um eine Zeitlimitüberschreitung zu vermeiden. Die Ladeoperation fordert die spezielle Z-Sperre an, schreibt die Phase fest, gibt die Sperre frei und fährt anschließend mit der Ladephase (LOAD-Phase) fort. Allen Anwendungen, die für die Tabelle eine Lesesperre anfordern, nachdem die Ladeoperation im Modus `ALLOW READ ACCESS` gestartet wurde, wird die Sperre mit Grant erteilt. Hierbei kommt es zu keinen Konflikten mit dieser speziellen Z-Sperre. Neue Anwendungen, die versuchen, vorhandene Daten aus der Zieltabelle zu lesen, können dies tun.

Zweiter Fall: Vor dem Commit der Daten am Ende der Ladeoperation fordert das Dienstprogramm LOAD eine exklusive Sperre des Typs "Z" für die Tabelle an. Das Dienstprogramm LOAD wartet, bis alle Anwendungen, die Sperren für die Tabelle halten, diese freigeben. Hierdurch kann sich vor dem Commit der Daten eine Verzögerung ergeben. Durch Verwendung der Option `LOCK WITH FORCE` wird die Beendigung von Anwendungen, die Konflikte verursachen, erzwungen und so eine Fortsetzung der Ladeoperation ohne Verzögerung ermöglicht. Normalerweise fordert eine Ladeoperation im Modus `ALLOW READ ACCESS` die exklusive Sperre für einen kurzen Zeitraum an. Wenn jedoch die Option `USE <tabellenbereichsname>` angegeben wird, bleibt die exklusive Sperre für die gesamte Dauer der INDEX COPY-Phase bestehen.

Wird das Dienstprogramm LOAD für eine Tabelle ausgeführt, die für mehrere Datenbankpartitionen definiert ist, wird das Ladeprozessmodell für jede einzelne Datenbankpartition ausgeführt. Dies bedeutet, dass Sperren unabhängig von den übrigen Datenbankpartitionen erteilt und freigegeben werden. Dies führt dazu, dass Deadlocks entstehen können, wenn eine Abfrage oder eine andere Operation zum gleichen Zeitpunkt ausgeführt wird und versucht, dieselben Sperren anzufordern. Angenommen Operation A hat eine Tabellensperre für die Datenbankpartition 0 erhalten und die Ladeoperation hat eine Tabellensperre für die Datenbankpartition 1 erhalten. Dadurch kann sich ein Deadlock ergeben, da Operation A auf die Erteilung einer Tabellensperre für die Datenbankpartition 1 wartet, während die Ladeoperation auf eine Tabellensperre für die Datenbankpartition 0 wartet. In diesem Fall führt der Deadlock-Detektor willkürlich für eine der Operationen einen Rollback durch.

Anmerkung:

1. Wenn eine Ladeoperation abgebrochen wird oder fehlschlägt, behält sie dieselbe Zugriffsebene bei, die beim Absetzen der Ladeoperation angegeben wurde. Wenn also eine Ladeoperation im Modus `ALLOW NO ACCESS` fehlschlägt, ist ein Zugriff auf die Tabellendaten erst dann möglich, wenn der Befehl LOAD TERMINATE oder LOAD RESTART abgesetzt wurde. Wird eine Ladeoperation im Modus `ALLOW READ ACCESS` abgebrochen, sind die zuvor vorhandenen Tabellendaten weiterhin für den Lesezugriff verfügbar.
2. Falls die Option `ALLOW READ ACCESS` für eine abgebrochene oder fehlgeschlagene Ladeoperation angegeben war, kann sie auch für die LOAD RESTART- oder

LOAD TERMINATE-Operation angegeben werden. War für die abgebrochene oder fehlgeschlagene Ladeoperation jedoch die Option ALLOW NO ACCESS angegeben, kann die Option ALLOW READ ACCESS weder für die LOAD RESTART- noch für die LOAD TERMINATE-Operation angegeben werden.

In den folgenden Fällen wird die Option ALLOW READ ACCESS nicht unterstützt:

- Die Option REPLACE ist angegeben. Da eine LOAD REPLACE-Operation die vorhandenen Tabellendaten abschneidet, bevor die neuen Daten geladen werden, können vorab vorhandene Daten erst dann abgefragt werden, wenn die Ladeoperation abgeschlossen ist.
- Die Indizes wurden als ungültig gekennzeichnet, und ihre Wiederherstellung steht an. Indizes können in bestimmten Szenarios für eine aktualisierende Recovery oder über die Verwendung des Befehls **db2dart** als ungültig gekennzeichnet werden.
- Die Option INDEXING MODE DEFERRED ist angegeben. Dieser Modus gibt an, dass eine Wiederherstellung der Indizes erforderlich ist.
- Eine Ladeoperation mit der Option ALLOW NO ACCESS wird erneut gestartet oder beendet. Eine Ladeoperation im Modus ALLOW READ ACCESS kann erst dann für die Tabelle stattfinden, nachdem diese vollständig in den Onlinestatus versetzt wurde.
- Eine Ladeoperation findet für eine Tabelle statt, die sich im Status Set Integrity Pending No Access (Festlegen der Integrität anstehend, kein Zugriff) befindet. Dies gilt auch bei mehreren Ladeoperationen für Tabellen mit Integritätsbedingungen. Eine Tabelle wird erst dann in den Onlinestatus versetzt, nachdem die Anweisung SET INTEGRITY abgesetzt wurde.

Wenn sich Tabellendaten im Offlinestatus befinden, ist ein Lesezugriff während einer Ladeoperation generell erst dann möglich, wenn die Tabelle wieder im Onlinestatus steht.

Tabellenbereichsstatus während und nach Ladeoperationen

Das Dienstprogramm LOAD setzt Angaben zum Tabellenbereichsstatus ein, um die Datenbankkonsistenz während einer Ladeoperation zu gewährleisten. Über diese Angaben zum Status ist es möglich, den Zugriff auf Daten zu steuern oder Benutzeraktionen zu sondieren.

Das Dienstprogramm LOAD versetzt die Tabellenbereiche, die von der Ladeoperation betroffen sind, nicht in den Quiescemodus (d. h. sperrt sie nicht permanent) und verwendet Statuswerte für Tabellenbereiche nur bei Ladeoperationen, bei denen Sie den Parameter **COPY NO** angegeben haben.

Mit dem Befehl **LIST TABLESPACES** können Sie die Statuswerte von Tabellenbereichen prüfen. Tabellenbereiche können sich in mehreren Status gleichzeitig befinden. Der Befehl **LIST TABLESPACES** gibt die folgenden Statuswerte zurück:

Normal

Der normale Status ist der Anfangsstatus eines Tabellenbereichs nach seiner Erstellung und gibt an, dass derzeit keine (abnormalen) Status in dem Tabellenbereich vorhanden sind.

Laden läuft

Der Status "Laden läuft" gibt an, dass für den Tabellenbereich eine Ladeoperation in Bearbeitung ist. Dieser Status verhindert, dass abhängige Tabellen während einer Ladeoperation gesichert werden. Dieser Tabellenbereichsstatus unterscheidet sich dadurch vom Tabellenstatus "Laden läuft"

(der in allen Ladeoperationen verwendet wird), da das Dienstprogramm LOAD Tabellenbereiche nur dann in den Status "Laden läuft" versetzt, wenn Sie den Parameter **COPY NO** bei einer wiederherstellbaren Datenbank angegeben haben. Die Tabellenbereiche behalten diesen Status für die Dauer der Ladeoperation bei.

Backup anstehend

Wenn Sie eine Ladeoperation für eine wiederherstellbare Datenbank ausführen und den Parameter **COPY NO** angeben, werden Tabellenbereiche in den Tabellenbereichsstatus "Backup anstehend" nach der ersten Festschreibung versetzt. Ein Tabellenbereich mit dem Status "Backup anstehend" kann nicht aktualisiert werden. Sie können den Status "Backup anstehend" für den Tabellenbereich nur durch die Sicherung des Tabellenbereichs entfernen. Der Tabellenbereich verbleibt auch bei einem Abbruch der Ladeoperation im Status "Backup anstehend", weil der Status des Tabellenbereichs am Beginn der Ladeoperation geändert wird und nicht zurückgesetzt werden kann.

Restore anstehend

Wenn Sie erfolgreich eine Ladeoperation mit der Option **COPY NO** ausführen, die Datenbank wiederherstellen und bei dieser Operation eine aktualisierende Recovery ausführen, werden die zugeordneten Tabellenbereiche in den Status "Restore anstehend" versetzt. Um den Status "Restore anstehend" für Tabellenbereiche aufzuheben, muss ein Restore ausgeführt werden.

Anmerkung: DB2 LOAD versetzt Tabellenbereiche nicht in den Status **Laden anstehend** oder **Löschen anstehend**.

Beispiel für einen Tabellenbereichsstatus

Wenn Sie eine Eingabedatei (`staffdata.del`) wie folgt in die Tabelle NEWSTAFF laden:

```
update db cfg for sample using logarchmeth1 logretain;
backup db sample;
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff copy no;
connect reset;
```

und Sie eine weitere Sitzung öffnen und die folgenden Befehle absetzen:

```
connect to sample;
list tablespaces;
connect reset;
```

befindet sich USERSPACE1 (der Standardtabellenbereich für die Beispieldatenbank) im Status "Laden läuft" und nach der ersten Festschreibung außerdem im Status "Backup anstehend". Nachdem die Ladeoperation beendet ist, zeigt der Befehl **LIST TABLESPACES**, dass der Tabellenbereich USERSPACE1 sich nun im Status "Backup anstehend" befindet:

```
  Tabellenbereichs-ID                       = 2
Name                                 = USERSPACE1
Typ                                  = Von der Datenbank verwalteter Bereich
Inhalt                               = Alle permanenten Daten. Großer Tabellenbereich.
Status                               = 0x0020
   Detaillierte Erläuterung:
Backup anstehend
```

Tabellenstatus während und nach Ladeoperationen

Das Dienstprogramm LOAD setzt Angaben zum Tabellenstatus ein, um die Datenbankkonsistenz während einer Ladeoperation zu gewährleisten. Über diese Angaben zum Status ist es möglich, den Zugriff auf Daten zu steuern oder Benutzeraktionen zu sondieren.

Setzen Sie zum Festlegen des Status einer Tabelle den Befehl **LOAD QUERY** ab, der außerdem den Status einer Ladeoperation prüft. Tabellen können sich in mehreren Status gleichzeitig befinden. Der Befehl **LOAD QUERY** gibt die folgenden Statuswerte zurück:

Normal
Der normale Status ist der Anfangsstatus einer Tabelle nach ihrer Erstellung und gibt an, dass derzeit keine (abnormalen) Status in der Tabelle vorhanden sind.

Lesezugriff
Wenn Sie die Option **ALLOW READ ACCESS** angeben, gilt der Status "Lesezugriff" für die Tabelle. Die Daten, die vor dem Aufruf des Befehls LOAD in der Tabelle vorhanden waren, sind während der Ladeoperation nur im Lesezugriff verfügbar. Wenn Sie die Option **ALLOW READ ACCESS** angeben und die Ladeoperation fehlschlägt, sind die Daten, die vor der Ladeoperation in der Tabelle vorhanden waren, nach dem Fehler weiterhin im Lesezugriff verfügbar.

Laden läuft
Der Tabellenstatus "Laden läuft" gibt an, dass für die Tabelle eine Ladeoperation in Bearbeitung ist. Das Dienstprogramm LOAD entfernt diesen Übergangsstatus, nachdem das Laden erfolgreich abgeschlossen wurde. Wenn die Ladeoperation jedoch fehlschlägt oder unterbrochen wird, ändert sich der Tabellenstatus in "Laden anstehend".

Umverteilung läuft
Der Tabellenstatus "Umverteilung läuft" gibt an, dass für die Tabelle eine Umverteilung in Bearbeitung ist. Das Dienstprogramm REDISTRIBUTE entfernt diesen Übergangsstatus, nachdem die Verarbeitung der Tabelle erfolgreich abgeschlossen wurde. Wenn die Umverteilungsoperation jedoch fehlschlägt oder unterbrochen wird, ändert sich der Tabellenstatus in "Umverteilung anstehend".

Laden anstehend
Der Tabellenstatus "Laden anstehend" gibt an, dass eine Ladeoperation fehlgeschlagen ist oder unterbrochen wurde. Mit einem der folgenden Schritte können Sie den Status "Laden anstehend" entfernen:

- Ermitteln Sie zunächst die Fehlerursache. Wenn das Dienstprogramm LOAD beispielsweise nicht über genügend Plattenspeicherplatz verfügte, fügen Sie dem Tabellenbereich Container hinzu. Starten Sie anschließend die Ladeoperation erneut.
- Beenden Sie die Ladeoperation.
- Führen Sie eine **LOAD REPLACE**-Operation für die Tabelle aus, bei der eine Ladeoperation fehlgeschlagen ist.
- Stellen Sie Tabellenbereiche für die Ladetabelle wieder her. Verwenden Sie dazu den Befehl **RESTORE DATABASE** mit dem letzten Backup eines Tabellenbereichs oder einer Datenbank und führen Sie danach weitere Recoveryaktionen aus.

Umverteilung anstehend
Der Tabellenstatus "Umverteilung anstehend" gibt an, dass eine Umvertei-

lungsoperation fehlgeschlagen ist oder unterbrochen wurde. Sie können die Operation **REDISTRIBUTE CONTINUE** oder **REDISTRIBUTE ABORT** ausführen, um den Status "Umverteilung anstehend" zu entfernen.

Für Laden nicht neu startbar
Im Status "Für Laden nicht neu startbar" wird eine Tabelle teilweise geladen und lässt ein erneutes Starten der Ladeoperation nicht zu. Es gibt zwei Situationen, in denen eine Tabelle in den Status "Für Laden nicht neu startbar" versetzt wird:

- Wenn Sie eine aktualisierende Recovery nach einer fehlgeschlagenen Ladeoperation ausführen, die Sie nicht erfolgreich neu starten oder beenden konnten
- Wenn Sie eine Restoreoperation auf der Basis eines Online-Backups ausführen, der erstellt wurde, während die Tabelle sich im Tabellenstatus "Laden läuft" oder "Laden anstehend" befand.

Für die Tabelle gilt außerdem der Status "Laden anstehend". Wenn Sie für die Tabelle den Status "Für Laden nicht neu startbar" entfernen möchten, setzen Sie den Befehl **LOAD TERMINATE** oder **LOAD REPLACE** ab.

Festlegen der Integrität anstehend
Der Status "Festlegen der Integrität anstehend" gibt an, dass für die geladene Tabelle Integritätsbedingungen gelten, die noch nicht überprüft wurden. Das Dienstprogramm LOAD versetzt eine Tabelle in diesen Status, sobald eine Ladeoperation für eine Tabelle mit Integritätsbedingungen beginnt. Mit der Anweisung SET INTEGRITY können Sie den Status "Festlegen der Integrität anstehend" für die Tabelle aufheben.

Indizes des Typs 1
Der Status "Indizes des Typs 1" gibt an, dass die Tabelle derzeit Indizes des Typs 1 verwendet. Indizes des Typs 1 werden seit Version 9.7 nicht mehr unterstützt. Diese Indizes müssen vor einem Upgrade auf Version 10 in Indizes des Typs 2 konvertiert werden. Andernfalls werden die Indizes des Typs 1 beim ersten Zugriff auf eine Tabelle automatisch als Indizes des Typs 2 erneut erstellt.

Details zum Konvertieren von Indizes des Typs 1 vor dem Upgrade von Datenbanken finden Sie im Thema „Konvertieren von Indizes des Typs 1 in Indizes des Typs 2".

Nicht verfügbar
Eine Tabelle wird durch eine aktualisierende Recovery, die über eine nicht wiederherstellbare Ladeoperation erfolgt, in den Status "Nicht verfügbar" versetzt. In diesem Status ist die Tabelle nicht verfügbar, Sie müssen Sie freigeben oder aus einem Backup wiederherstellen.

Beispiel für eine Tabelle mit mehreren Status

Wenn Sie eine Eingabedatei (`staffdata.del`) mit einem großen Datenvolumen wie folgt in die Tabelle NEWSTAFF laden:

```
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff allow read access;
connect reset;
```

und Sie eine weitere Sitzung öffnen und die folgenden Befehle absetzen:

```
connect to sample;
load query table newstaff;
connect reset;
```

zeigt der Befehl **LOAD QUERY**, dass die Tabelle NEWSTAFF sich in den beiden Tabellenstatus "Lesezugriff" und "Laden läuft" befindet:

```
Tabellenstatus:
     Laden läuft
     Lesezugriff
```

LOAD-Ausnahmetabellen

Eine LOAD-Ausnahmetabelle stellt einen konsolidierten Bericht aller Zeilen zur Verfügung, die gegen eindeutige Indexregeln, Bereichsvorgaben und Sicherheitsrichtlinien verstoßen haben, während eine Ladeoperation ausgeführt wurde. Sie können eine LOAD-Ausnahmetabelle angeben, indem Sie die Klausel FOR EXCEPTION des Befehls **LOAD** verwenden.

Einschränkung: Eine Ausnahmetabelle darf keine Identitätsspalte oder generierte Spalten anderen Typs enthalten. Ist in der Primärtabelle eine Identitätsspalte vorhanden, sollte die entsprechende Spalte in der Ausnahmetabelle nur den Spaltentyp, die Spaltenlänge und die Attribute enthalten, die angeben, ob die Spalte Nullwerte enthalten kann oder nicht. Die Ausnahmetabelle kann außerdem auch nicht partitioniert werden und darf keinen eindeutigen Index haben.Darüber hinaus können Sie keine Ausnahmetabelle angeben, wenn Folgendes zutrifft:

- Wenn die Zieltabelle Sicherheit mit kennsatzbasierter Zugriffssteuerung (LBAC) verwendet und mindestens eine XML-Spalte enthält.
- Wenn die Zieltabelle in Bereiche partitioniert ist und mindestens eine XML-Spalte enthält.

Die Ausnahmetabelle, die mit dem Dienstprogramm LOAD verwendet wird, ist identisch mit den Ausnahmetabellen, die von der Anweisung SET INTEGRITY verwendet werden. Es handelt sich dabei um eine vom Benutzer erstellte Tabelle, die die Definition der zu ladenden Tabelle wiedergibt und einige zusätzliche Spalten beinhaltet.

Eine LOAD-Ausnahmetabelle kann dem Tabellenbereich, in dem sich die zu ladende Tabelle befindet, oder einem anderen Tabellenbereich zugeordnet werden. In beiden Fällen sollten die LOAD-Ausnahmetabelle und die zu ladende Tabelle derselben Datenbankpartitionsgruppe zugeordnet werden und denselben Verteilungsschlüssel aufweisen. Stellen Sie zusätzlich sicher, dass die Ausnahmetabelle und die zu ladende Tabelle dieselbe Partitionszuordnungs-ID (SYSIBM.SYSTABLES.PMAP_ID) aufweisen, die sich während der Umverteilungsoperation (Operation zum Hinzufügen/Löschen von Datenbankpartitionen) unterscheiden kann.

Einsatzmöglichkeiten einer Ausnahmetabelle

Eine Ausnahmetabelle können Sie verwenden, um Daten zu laden, die über einen eindeutigen Index verfügen und die doppelte Datensätze enthalten können. Wenn Sie keine Ausnahmetabelle angeben und doppelte Datensätze gefunden werden, wird die Ladeoperation fortgesetzt. In diesem Fall wird lediglich eine Warnung über die gelöschten doppelten Datensätze angezeigt. Die doppelten Datensätze werden nicht protokolliert.

Nach Fertigstellung der Ladeoperation können Sie die Informationen in der Ausnahmetabelle verwenden, um fehlerhafte Daten zu korrigieren. Anschließend können die korrigierten Daten in die Tabelle eingefügt werden.

Zeilen werden an die in der Ausnahmetabelle vorhandenen Informationen angefügt. Da nicht überprüft wird, ob die Tabelle leer ist, werden neue Informationen einfach an die ungültigen Zeilen aus vorherigen Ladeoperationen angefügt. Wenn

die Ausnahmetabelle nur die ungültigen Zeilen der aktuellen Ladeoperation enthalten soll, müssen Sie die bereits vorhandenen Zeilen löschen, bevor Sie das Dienstprogramm aufrufen. Bei der Definition einer Ladeoperation können Sie festlegen, dass der Zeitpunkt, zu dem eine Verletzung festgestellt wird, sowie der Name der verletzten Integritätsbedingung in der Ausnahmetabelle aufgezeichnet werden.

Da alle Löschereignisse protokolliert werden, kann das Protokoll während der DELETE-Phase der Ladeoperation stark an Umfang zunehmen, wenn sehr viele Datensätze gegen die Eindeutigkeitsbedingung verstoßen.

Zeilen, die vor der Indexerstellung aufgrund ungültiger Daten zurückgewiesen wurden, werden nicht in die Ausnahmetabelle aufgenommen.

Fehlgeschlagener oder unvollständiger Ladevorgang

Erneutes Starten einer unterbrochenen Ladeoperation

Wenn während der Ausführung einer Ladeoperation ein Fehler auftritt oder wenn diese unterbrochen wird, können Sie diese mit dem Dienstprogramm LOAD beenden, die Tabelle erneut laden oder die Ladeoperation nochmals starten.

Wenn das Dienstprogramm LOAD aufgrund eines Benutzerfehlers nicht gestartet werden kann, etwa weil eine Datendatei fehlt oder weil ungültige Spaltennamen festgestellt wurden, wird es beendet, und die Zieltabelle verbleibt in einem normalen Status.

Wenn die Ladeoperation beginnt, wird die Zieltabelle in den Tabellenstatus "Laden läuft" versetzt. Falls ein Fehler auftritt, ändert sich der Tabellenstatus in "Laden anstehend". Um die Tabelle wieder in einen anderen Status zu versetzen, können Sie **LOAD TERMINATE** eingeben, um die Operation mit einem Rollback zurückzusetzen, **LOAD REPLACE** eingeben, um die gesamte Tabelle erneut zu laden, oder **LOAD RESTART** eingeben.

Normalerweise ist es in dieser Situation zu empfehlen, dass Sie die Ladeoperation erneut starten. Dies spart Zeit, weil das Dienstprogramm LOAD die Ladeoperation ab dem zuletzt erreichten fehlerfreien Punkt innerhalb des Verarbeitungsablaufs und nicht vom Beginn der Operation an erneut startet. Wo genau der Neustart der Operation beginnt, hängt von den Parametern ab, die im ursprünglichen Befehl angegeben wurden. Wenn die Option `SAVECOUNT` angegeben wurde und bei der zuvor ausgeführten Ladeoperation in der LOAD-Phase ein Fehler aufgetreten ist, dann startet die Ladeoperation beim letzten erreichten Konsistenzpunkt. Andernfalls startet die Ladeoperation am Beginn der letzten Phase, die erfolgreich ausgeführt wurde (also bei der LOAD-, BUILD- oder DELETE-Phase).

Beim Laden von XML-Dokumenten ist die Vorgehensweise etwas anders. Da die Option `SAVECOUNT` für das Laden von XML-Daten nicht unterstützt wird, werden Ladeoperationen, bei denen in der LOAD-Phase ein Fehler auftritt, vom Beginn der Operation an erneut gestartet. Wie bei anderen Datentypen werden Indizes im Modus REBUILD erstellt, wenn bei der Ladeoperation in der BUILD-Phase ein Fehler auftritt; also wird die Tabelle durchsucht, um alle Indexschlüssel aus jeder Zeile zu erfassen. Zur Erfassung der Indexschlüssel müssen jedoch auch alle XML-Dokumente durchsucht werden. Zum Durchsuchen von XML-Dokumenten nach Schlüsseln muss eine erneute Syntaxanalyse für diese Dokumente erfolgen; diese Operation ist mit einem hohen Systemaufwand verbunden. Darüber hinaus müssen zuerst die internen XML-Indizes, wie z. B. Indizes für Regionen oder Pfade, erneut erstellt werden; dazu muss außerdem das XDA-Objekt durchsucht werden.

Nachdem Sie den Fehler behoben haben, der zu dem Fehler in der Ladeoperation führte, können Sie den Ladebefehl erneut eingeben. Vergewissern Sie sich, dass Sie dabei genau die gleichen Parameter eingeben wie beim ursprünglichen Befehl, sodass das Dienstprogramm LOAD die erforderlichen temporären Dateien lokalisieren kann. Eine Ausnahme bildet hierbei der Fall, dass Sie den Lesezugriff nicht zulassen wollen. Eine Ladeoperation, in der die Option ALLOW READ ACCESS angegeben wurde, kann auch mit der Option ALLOW NO ACCESS erneut gestartet werden.

Anmerkung: Versuchen Sie nicht, temporäre Dateien, die vom Dienstprogramm LOAD erstellt wurden, zu löschen oder zu ändern.

Wenn die mit dem folgenden Befehl eingeleitete Ladeoperation fehlschlägt:

```
LOAD FROM dateiname OF dateityp
SAVECOUNT n
MESSAGES nachrichtendatei
lademethode
INTO zieltabellenname
```

dann müssen Sie sie erneut starten, indem Sie die angegebene Lademethode (lademethode) durch die Methode RESTART ersetzen:

```
LOAD FROM dateiname OF dateityp
SAVECOUNT n
MESSAGES nachrichtendatei
RESTART
INTO zieltabellenname
```

Fehlgeschlagene Ladeoperationen, die nicht erneut gestartet werden können

Fehlgeschlagene oder unterbrochene Ladeoperationen können nicht erneut gestartet werden, wenn die dabei verwendete Tabelle sich im Status "Für Laden nicht neu startbar" befindet. Tabellen werden aus den folgenden Gründen in diesen Status versetzt:

- Nach einer fehlgeschlagenen Ladeoperation, die nicht erfolgreich erneut gestartet oder beendet werden konnte, wird eine aktualisierende Recovery ausgeführt.
- Eine Restoreoperation wird auf der Basis eines Online-Backups ausgeführt, der erstellt wurde, während die Tabelle sich im Tabellenstatus "Laden läuft" oder "Laden anstehend" befand.

Geben Sie in diesem Fall entweder den Befehl **LOAD TERMINATE** oder **LOAD REPLACE** ein.

Einschränkungen bei fehlgeschlagenen Ladeoperationen

Der Befehl **BACKUP DATABASE** gibt möglicherweise einen E/A-Fehler zurück, wenn der Befehl **LOAD** für eine Tabelle im SMS-Tabellenbereich fehlschlägt und die Tabelle im Status "Laden anstehend" verbleibt.

Die Tabellendaten werden möglicherweise nicht konsistent angezeigt, wenn sich eine Tabelle im Status "Laden anstehend" befindet. Inkonsistente Tabellendaten führen zum Fehlschlagen des Befehls **BACKUP DATABASE**. Die Tabelle bleibt inkonsistent, bis ein nachfolgender Befehl **LOAD TERMINATE**, **LOAD RESTART** oder **LOAD REPLACE** abgeschlossen ist.

Sie müssen den Status "Laden anstehend" für die Tabelle aufheben, bevor Sie ein Backup für die Datenbank ausführen.

Erneutes Starten oder Beenden einer Ladeoperation mit ALLOW READ ACCESS

Eine unterbrochene oder abgebrochene Ladeoperation, in der der Parameter **ALLOW READ ACCESS** angegeben wurde, kann auch mit dem Parameter **ALLOW READ ACCESS** erneut gestartet oder beendet werden. Durch die Verwendung des Parameters **ALLOW READ ACCESS** können andere Anwendungen die Tabellendaten abfragen, während die Operation zum Beenden oder erneutem Starten läuft. Wie auch bei einer Ladeoperation im Modus **ALLOW READ ACCESS** ("Lesezugriff zulassen") wird die Tabelle vor dem Commit der Daten exklusiv gesperrt.

Informationen zu diesem Vorgang

Falls das Indexobjekt nicht verfügbar ist oder als ungültig gekennzeichnet wurde, ist eine LOAD RESTART- oder LOAD TERMINATE-Operation im Modus **ALLOW READ ACCESS** nicht zulässig.

Falls die ursprüngliche Ladeoperation in der INDEX COPY-Phase unterbrochen oder abgebrochen wurde, ist eine LOAD RESTART-Operation im Modus **ALLOW READ ACCESS** nicht zulässig, da der Index beschädigt sein könnte.

Falls eine Ladeoperation im Modus **ALLOW READ ACCESS** während der LOAD-Phase unterbrochen oder abgebrochen wurde, wird sie in der LOAD-Phase erneut gestartet. Wurde sie während einer der anderen Phasen unterbrochen oder abgebrochen, findet der Neustart in der BUILD-Phase statt. Sofern die ursprüngliche Ladeoperation im Modus **ALLOW NO ACCESS** ("Keinen Zugriff zulassen") ausgeführt wurde, setzt die LOAD RESTART-Operation in der DELETE-Phase ein, wenn die ursprüngliche Ladeoperation diesen Punkt erreicht hat und der Index gültig ist. Wurde der Index als ungültig gekennzeichnet, startet das Dienstprogramm LOAD die Ladeoperation in der BUILD-Phase neu.

Anmerkung: Alle LOAD RESTART-Operationen wählen den Indexierungsmodus `REBUILD` selbst dann aus, wenn der Parameter **INDEXING MODE INCREMENTAL** angegeben wird.

Durch Absetzen eines Befehls **LOAD TERMINATE** wird die abgebrochene oder unterbrochene Ladeoperation generell mit einer minimalen Verzögerung zurückgesetzt. Wird jedoch ein Befehl **LOAD TERMINATE** für eine Ladeoperation abgesetzt, bei der **ALLOW READ ACCESS** und **INDEXING MODE INCREMENTAL** angegeben wurde, kann es unter Umständen zu einer Verzögerung kommen, während das Dienstprogramm LOAD die Indizes durchsucht und eventuelle Inkonsistenzen korrigiert. Die Länge dieser Verzögerung ist von der Größe der Indizes abhängig. Sie tritt unabhängig davon auf, ob der Parameter **ALLOW READ ACCESS** für die LOAD TERMINATE-Operation angegeben wurde. Eine Verzögerung findet nicht statt, wenn die ursprüngliche Ladeoperation vor der BUILD-Phase fehlschlug.

Anmerkung: Eine Verzögerung, die aus Korrekturen an Inkonsistenzen im Index entsteht, ist wesentlich geringer als die Verzögerung, die dadurch verursacht wird, dass die Indizes als ungültig gekennzeichnet und wiederhergestellt werden.

Eine LOAD RESTART-Operation kann nicht für eine Tabelle ausgeführt werden, deren Status "Für Laden nicht neu startbar" (Not Load Restartable) lautet. Die Tabelle kann während einer aktualisierenden Recovery in den Tabellenstatus "Für Laden nicht neu startbar" versetzt werden. Dies ist beispielsweise möglich, wenn die aktualisierende Recovery bis zu einem Zeitpunkt ausgeführt wird, der vor dem Ende einer Ladeoperation liegt, oder wenn eine aktualisierende Recovery über eine

abgebrochene Ladeoperation hinweg, jedoch nicht bis zum Ende der LOAD TERMINATE- oder LOAD RESTART-Operation ausgeführt wird.

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 der Veröffentlichung *Neuerungen in DB2 Version 10.1*.

Recovery von Daten mithilfe der Datei mit den Angaben zur Speicherposition der Ladekopie

Anhand der Registrierdatenbankvariablen **DB2LOADREC** wird die Datei mit den Angaben zur Speicherposition der Ladekopie identifiziert. Diese Datei wird während der aktualisierenden Recovery zum Lokalisieren der Ladekopie verwendet.

DB2LOADREC enthält die folgenden Informationen:

- Datenträgertyp
- Anzahl der zu verwendenden Datenträgereinheiten
- Speicherposition der Ladekopie, die bei einer Ladeoperation für eine Tabelle generiert wurde
- Dateiname der Ladekopie (sofern zutreffend)

Falls die Speicherpositionsdatei nicht existiert oder darin kein übereinstimmender Eintrag gefunden wurde, werden die Informationen aus dem Protokollsatz verwendet.

Die Informationen in der Datei könnten überschrieben werden, bevor die aktualisierende Recovery stattfindet.

Anmerkung:

1. In einer Mehrpartitionsdatenbank muss die Registrierdatenbankvariable **DB2LOADREC** unter Verwendung des Befehls **db2set** für alle Datenbankpartitionsserver festgelegt werden.
2. In einer Mehrpartitionsdatenbank muss die Datei der Ladekopie auf jedem Datenbankpartitionsserver vorhanden sein, und der Dateiname (einschließlich des Pfads) muss derselbe sein.
3. Falls ein Eintrag in der Datei, die durch die Registrierdatenbankvariable **DB2LOADREC** angegeben ist, ungültig ist, wird der ungültige Eintrag durch die Informationen aus der alten Datei mit den Angaben zur Speicherposition der Ladekopie ersetzt.

Die Datei mit den Angaben zur Speicherposition enthält die folgenden Informationen. Die ersten fünf Parameter müssen gültige Werte aufweisen und werden zum Identifizieren der Ladekopie verwendet. Die gesamte Struktur wird für jede aufgezeichnete Ladekopie wiederholt. Beispiel:

```
TIMestamp      19950725182542         * Time stamp generated at load time
DBPartition    0                      * DB Partition number (OPTIONAL)
SCHema         PAYROLL                * Schema of table loaded
TABlename      EMPLOYEES              * Table name
DATabasename   DBT                    * Database name
DB2instance    toronto                * DB2INSTANCE
BUFfernumber   NULL                   * Number of buffers to be used for
                                       Recovery SESsionnumber  NULL
                                     * Number of sessions to be used for
                                       Recovery TYPeofmedia    L
                                     * Type of media - L for local device
```

```
                                                    A for TSM
                                                    O for other vendors

LOCationnumber 3
* Number of locations
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
DBP            18
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            O
SHRlib         /@sys/lib/backup_vendor.a
```

Anmerkung:

1. Die ersten drei Zeichen in jedem Schlüsselwort sind wichtig. Alle Schlüsselwörter sind in der angegebenen Reihenfolge erforderlich. Leerzeilen werden nicht akzeptiert.
2. Die Zeitmarke hat das Format *jjjjmmttssmmss* (Jahr Monat Tag Stunde Minute Sekunde).
3. Alle Felder sind verbindlich, mit Ausnahme der Felder BUF und SES (die den Wert NULL haben können) sowie des Feldes DBP (das nicht in der Liste aufgeführt zu sein braucht). Falls für SES NULL definiert ist, wird der durch den Konfigurationsparameter *dft_loadrec_ses* angegebene Wert verwendet. Ist BUF NULL, lautet der Standardwert SES+2.
4. Falls auch nur einer der der Einträge in der Datei mit den Angaben zur Speicherposition ungültig ist, werden die entsprechenden Werte aus der früheren Datei mit den Angaben zur Speicherposition der Ladekopie verwendet.
5. Es gibt folgende Datenträgertypen: lokale Einheit (`L` für Band, Platte oder Diskette), TSM (`A`) oder Datenträger anderer Lieferanten (`O`). Lautet der Typ `L`, ist die Anzahl der Speicherpositionen, gefolgt von den Einträgen zur Speicherposition, erforderlich. Lautet der Typ `A`, ist keine weitere Eingabe erforderlich. Lautet der Typ `O`, ist der Name der gemeinsam benutzten Bibliothek erforderlich.
6. Der Parameter `SHRlib` zeigt auf eine Bibliothek, die über eine Funktion zum Speichern der Ladekopiedaten verfügt.
7. Wenn Sie eine Ladeoperation aufrufen und hierbei die Option `COPY NO` oder `NONRECOVERABLE` angeben und nach Abschluss der Operation keine Backup-Kopie der Datenbank oder der betroffenen Tabellenbereiche erstellen, können Sie die Datenbank oder die Tabellenbereiche nicht in dem Zustand wiederherstellen, in dem sich diese nach der Ladeoperation befinden werden. Das heißt, dass Sie keine aktualisierende Recovery ausführen können, um die Datenbank oder die Tabellenbereiche auf dem Stand wiederherzustellen, der nach der Ladeoperation vorlag. Sie können die Datenbank oder die Tabellenbereiche lediglich für einen Zeitpunkt vor der Ladeoperation wiederherstellen.

Für den Fall, dass Sie eine bestimmte Ladekopie verwenden wollen, können Sie in der Datei des Recoveryprotokolls für die Datenbank die Zeitmarke für diese spezifische Ladeoperation ermitteln. In einer Mehrpartitionsdatenbank ist die Datei des Recoveryprotokolls für jede Datenbankpartition lokal.

LOAD-Speicherauszugsdatei

Der Änderungswert `dumpfile` für den Dateityp teilt dem Dienstprogramm LOAD den Namen und die Speicherposition der Ausnahmedatei mit, in die die zurückgewiesenen Zeilen geschrieben werden.

Bei Ausführung in einer Umgebung mit partitionierten Datenbanken können Zeilen entweder von den Subagenten für Partitionierung oder den Subagenten für Laden zurückgewiesen werden. Aus diesem Grund erhält der Name der Speicherauszugsdatei eine Erweiterung, die den Typ des Subagenten angibt sowie die Datenbankpartitionsnummer, in der die Ausnahmen generiert wurden. Beispiel: Angenommen, der folgende Wert wurde für die Speicherauszugsdatei (dumpfile) angegeben:

```
dumpfile = "/u/usrname/dumpit"
```

In diesem Fall werden die Zeilen, die vom Subagenten für Laden in Datenbankpartition fünf zurückgewiesen wurden, in einer Datei namens `/u/usrname/dumpit.load.005` gespeichert. Zeilen, die vom Subagenten für Laden in Datenbankpartition zwei zurückgewiesen wurden, werden in einer Datei namens `/u/usrname/dumpit.load.002` gespeichert, und Zeilen, die vom Subagenten für Partitionierung auf Datenbankpartition zwei zurückgewiesen wurden, werden in einer Datei namens `/u/usrname/dumpit.part.002` gespeichert usw.

Für Zeilen, die vom Subagenten für Laden zurückgewiesen werden, gilt Folgendes: Ist die Zeile kürzer als 32 768 Byte, wird der Datensatz vollständig in die Speicherauszugsdatei kopiert. Ist die Zeile länger, wird ein Zeilenfragment (das auch die Endbyte des Datensatzes enthält) in die Datei geschrieben.

Zeilen, die vom Subagenten für Partitionierung zurückgewiesen werden, werden unabhängig von der Länge des Datensatzes vollständig in die Speicherauszugsdatei kopiert.

Temporäre Dateien für das Dienstprogramm LOAD

Das DB2-Datenbanksystem erstellt während der Ladeverarbeitung temporäre Binärdateien. Diese Dateien werden zur Recovery einer Ladeoperation nach einem Systemabsturz, bei LOAD TERMINATE-Operationen, für Warnungen und Fehlernachrichten sowie die Laufzeitsteuerdaten verwendet.

Temporäre Dateien für das Dienstprogramm LOAD werden entfernt, nachdem die Ladeoperation ohne Fehler beendet wurde. Die temporären Dateien werden in einen Pfad geschrieben, der mit dem Parameter *temp-pfadname* des Befehls **LOAD** oder mit dem Parameter *piTempFilesPath* der API db2Load angegeben werden kann. Der Standardpfad ist ein Unterverzeichnis des Datenbankverzeichnisses.

Der Pfad der temporären Dateien befindet sich auf der Servermaschine. Der Zugriff auf diesen Pfad erfolgt ausschließlich durch die DB2-Instanz. Daher muss jede Angabe eines Pfadnamens im Parameter *temp-pfadname* unbedingt die Verzeichnisstruktur auf dem Server und nicht die auf dem Client berücksichtigen. Darüber hinaus muss der Eigner der DB2-Instanz unbedingt über Lese- und Schreibberechtigung für den Pfad verfügen.

Anmerkung: In einer DB2 pureScale-Umgebung sollten die temporären Dateien für das Dienstprogramm LOAD in einem Pfad vorhanden sein, auf den alle Member zugreifen können (z. B. auf einer gemeinsam genutzten Platte). Die temporären Dateien müssen auf einer gemeinsam genutzten Platte vorhanden sein. Andernfalls können die Recovery nach dem Absturz eines Members und **LOAD TERMINATE**-Operationen, die von einem anderen Member ausgeführt werden, zu Problemen führen.

Dies ist ein Unterschied zu einer Umgebung mit partitionierten Datenbanken, in denen die temporären Dateien für das Dienstprogramm LOAD auf einer lokalen Platte vorhanden sein sollten. Sie sollten keinen NFS-basierten (NFS - Network File System) Pfad auswählen. Andernfalls treten während der Ladeoperation erhebliche Leistungseinbußen auf.

Achtung: Die in diesen Pfad geschriebenen temporären Dateien dürfen unter keinen Umständen manipuliert werden. Andernfalls treten bei der Ladeoperation Störungen auf, und die Integrität der Datenbank wird gefährdet.

Protokollsätze für das Dienstprogramm LOAD

Der Dienstprogramm-Manager erzeugt Protokollsätze für eine Reihe von DB2-Dienstprogrammen. Dazu zählt auch das Dienstprogramm LOAD.

Die folgenden Protokollsätze kennzeichnen den Anfang oder das Ende eines bestimmten Vorgangs während einer Ladeoperation:

- **SETUP-Phase**
  - LOAD START. Dieser Protokollsatz markiert den Anfang der SETUP-Phase einer Ladeoperation.
  - Commitprotokollsatz. Dieser Protokollsatz weist auf einen erfolgreichen Abschluss der SETUP-Phase hin.
  - Abbruchsprotokollsatz (Abort). Dieser Protokollsatz weist darauf hin, dass die SETUP-Phase fehlgeschlagen ist. Bei Einzelpartitionsdatenbanken wird stattdessen ein Commitprotokollsatz für "Local Pending" generiert, wenn in der SETUP-Phase des Ladevorgangs ein Fehler auftritt, bevor die Tabelle physisch geändert wurde.
- **LOAD-Phase**
  - LOAD START. Dieser Protokollsatz markiert den Anfang der LOAD-Phase einer Ladeoperation.
  - Commitprotokollsatz für "Local Pending". Dieser Protokollsatz weist auf einen erfolgreichen Abschluss der LOAD-Phase hin.
  - Abbruchsprotokollsatz (Abort). Dieser Protokollsatz weist darauf hin, dass die LOAD-Phase fehlgeschlagen ist.
- **DELETE-Phase**
  - LOAD DELETE START. Dieser Protokollsatz ist dem Anfang der DELETE-Phase einer Ladeoperation zugeordnet. Die DELETE-Phase wird nur gestartet, wenn doppelte Primärschlüsselwerte vorliegen. Während der DELETE-Phase wird jede Löschoperation für einen Tabellensatz oder einen Indexschlüssel protokolliert.
  - LOAD DELETE END. Dieser Protokollsatz ist dem Ende der DELETE-Phase einer Ladeoperation zugeordnet. Diese DELETE-Phase wird während der aktualisierenden Recovery einer erfolgreichen Ladeoperation wiederholt.

Die folgende Liste vermittelt einen Überblick über die Protokollsätze, die das Dienstprogramm LOAD - abhängig vom Umfang der Eingabedaten - erstellt:

- Für jeden Speicherbereich, der durch das Dienstprogramm in einem DMS-Tabellenbereich für einen Tabellenbereich zugeordnet oder gelöscht wurde, werden zwei Protokollsätze erstellt.
- Für jeden belegten Block von Identitätswerten wird ein Protokollsatz erstellt.
- Für alle Datenzeilen oder Indexschlüssel, die während der DELETE-Phase einer Ladeoperation gelöscht werden, werden Protokollsätze erstellt.
- Wenn eine Ladeoperation unter Angabe der Optionen `ALLOW READ ACCESS` und `INDEXING MODE INCREMENTAL` ausgeführt wird, werden Protokollsätze erstellt, die die Integrität der Indexbaumstruktur erhalten. Die Anzahl der protokollierten Sätze ist wesentlich geringer als bei einer vollständig protokollierten Einfügung in den Index.

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 Datenbankpartitionen 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.

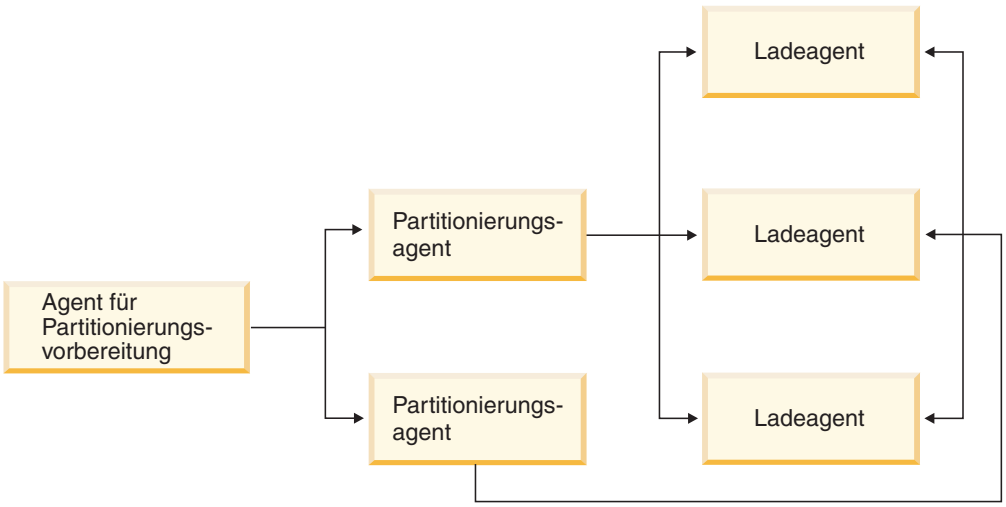


Abbildung 13. Übersicht über das Laden in partitionierte Datenbanken. Die Quellendaten werden durch den Agenten für Partitionierungsvorbereitung gelesen. Ungefähr die Hälfte der Daten wird jeweils an zwei Partitionierungsagenten gesendet, die die Daten verteilen und an eine der drei Datenbankpartitionen senden. Auf den einzelnen Datenbankpartitionen werden die Daten durch den jeweiligen Ladeagenten geladen.

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 Datenbankpartition. 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)
```

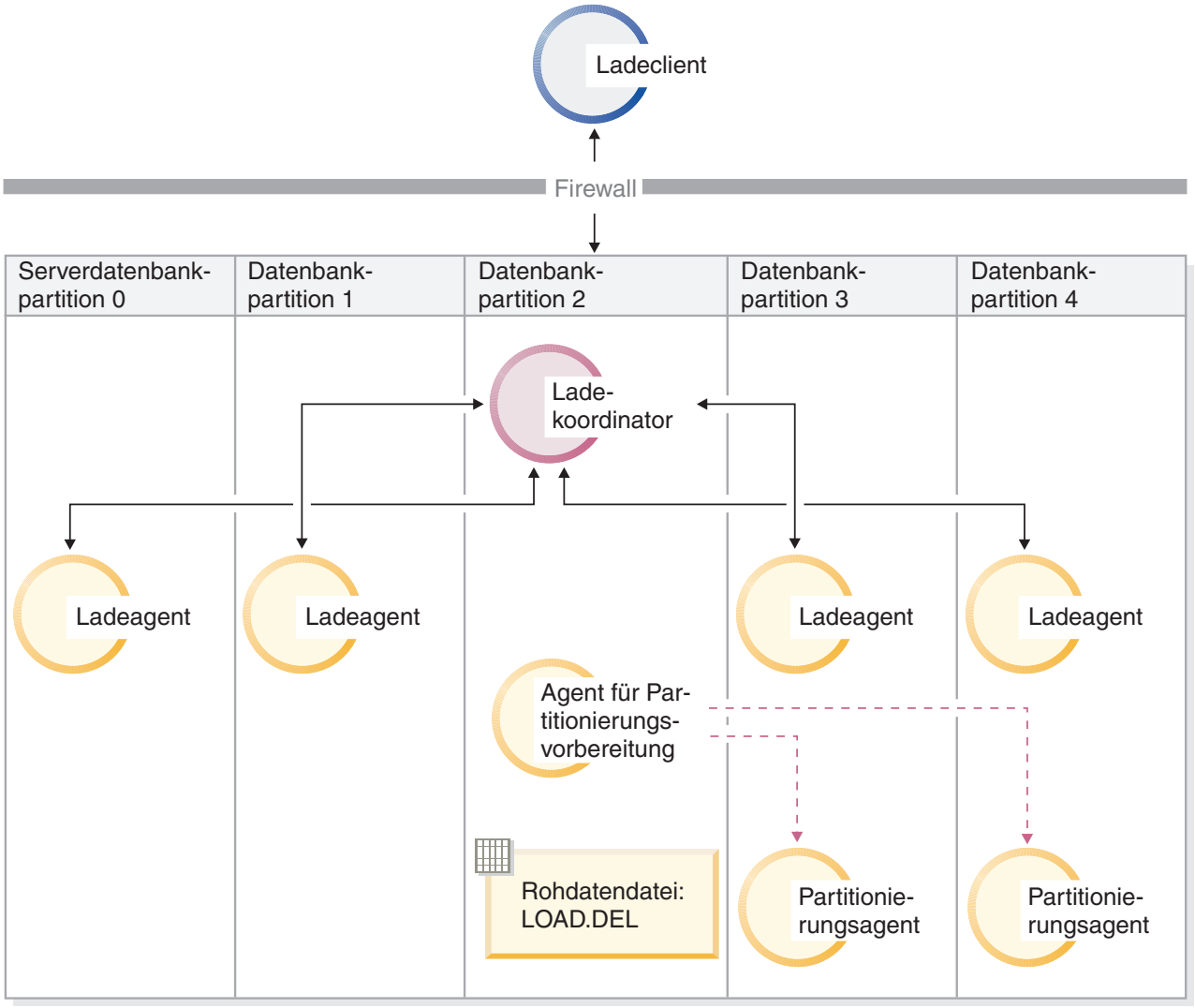


Abbildung 14. Laden von Daten in die Datenbankpartitionen 3 und 4. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die Daten werden in die Datenbankpartitionen 3 und 4 geladen.

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)
```

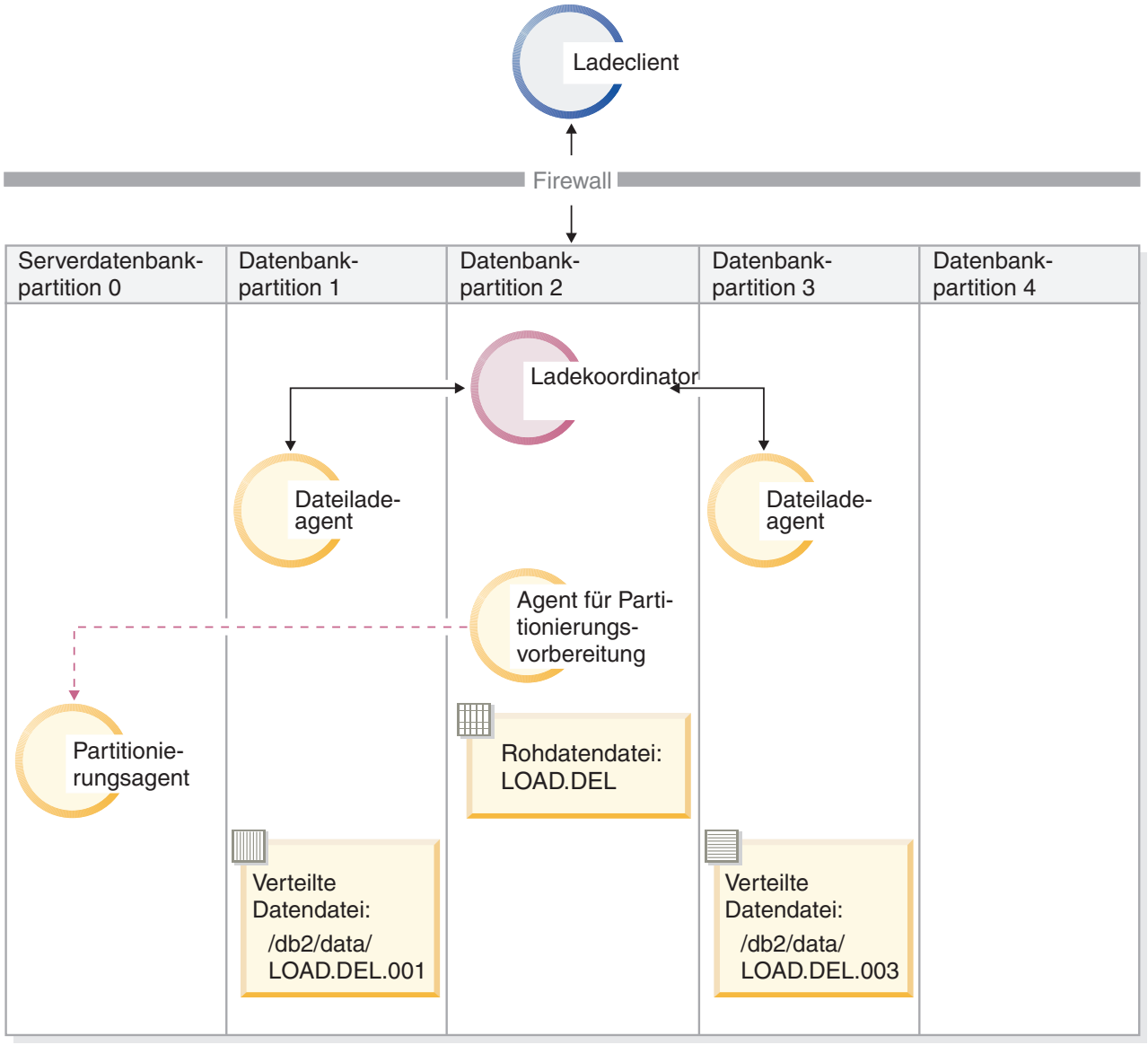


Abbildung 15. Laden von Daten in die Datenbankpartitionen 1 und 3 mithilfe eines Partitionierungsagenten. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die Daten werden auf die Datenbankpartitionen 1 und 3 geladen, wobei ein Partitionierungsagent 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 Lade-

datenbankpartition 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
```

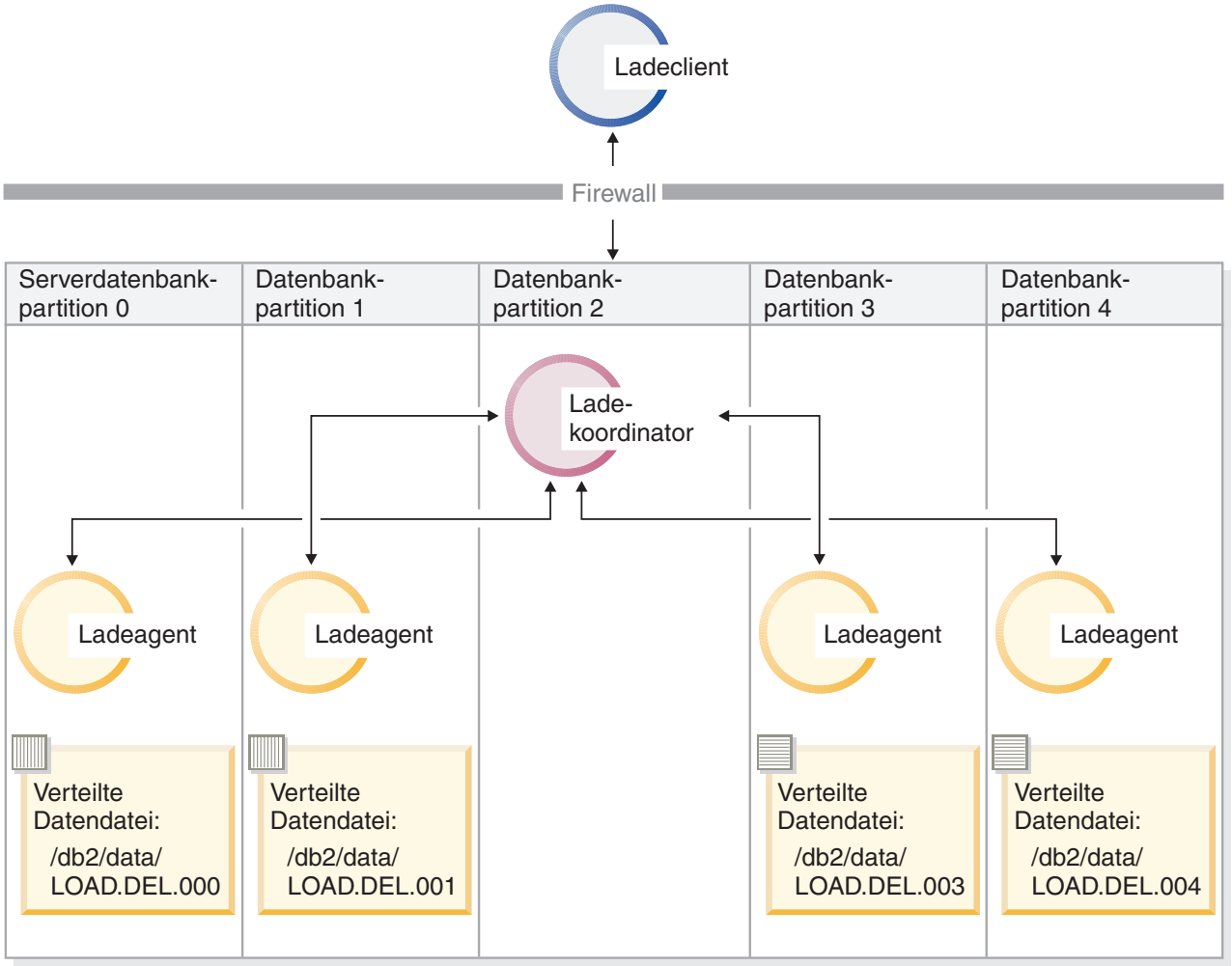


Abbildung 16. Laden von Daten in alle Datenbankpartitionen mit einer bestimmten definierten Tabelle. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die verteilten Daten werden auf alle Datenbankpartitionen geladen, auf denen TABLE1 definiert ist.

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
```

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 verwenden), 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.

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 aller 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 während 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.

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.

Referenz - Laden in einer partitionierten Umgebung

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
```

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 Speicher-

position 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 können. 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 Ab-

schnitt 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 Datenbankpartition 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 gleichmäßi-

gen 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.

Dienstprogramm INGEST

Das Dienstprogramm INGEST (das manchmal auch als fortlaufende Datenaufnahme oder CDI (Continuous Data Ingest) bezeichnet wird), ist ein clientseitiges DB2-Hochgeschwindigkeitsdienstprogramm, mit dem Datenströme aus Dateien und Pipes in DB2-Zieltabellen geleitet werden. Da das Dienstprogramm INGEST große Mengen von Echtzeitdaten ohne das Sperren der Zieltabelle verschieben kann, müssen Sie sich nicht zwischen Datenaktualität und Datenverfügbarkeit entscheiden.

Das Dienstprogramm INGEST nimmt mithilfe von ETL-Tools oder auf andere Weise vorverarbeitete Daten direkt oder aus der Dateienausgabe auf. Es kann kontinuierlich ausgeführt werden und kann daher einen fortlaufenden Datenstrom über Pipes verarbeiten. Die Daten werden mit einer hohen Geschwindigkeit eingepflegt, die sogar ausreicht, um große Datenbanken in Umgebungen mit partitionierten Datenbanken zu füllen.

Mit dem Befehl **INGEST** wird die Zieltabelle mit geringen Latenzzeiten in einem einzigen Schritt aktualisiert. Das Dienstprogramm INGEST verwendet Zeilensperren, wodurch es nur minimal zu Interferenzen mit anderen Benutzeraktivitäten in derselben Tabelle kommt.

Mit diesem Dienstprogramm können Sie mithilfe einer SQL-ähnlichen Schnittstelle DML-Operationen für eine Tabelle ausführen, ohne die Zieltabelle zu sperren. Diese INGEST-Operationen unterstützen die folgenden SQL-Anweisungen: INSERT, UPDATE, MERGE, REPLACE und DELETE. Das Dienstprogramm INGEST unterstützt auch die Verwendung von SQL-Ausdrücken zum Erstellen individueller Spaltenwerte aus mehr als einem Datenfeld.

Folgendes sind weitere Funktionen des Dienstprogramms INGEST:

- **Commit nach Zeit oder Zeilenanzahl** Sie können den INGEST-Konfigurationsparameter **commit_count** verwenden, um die Häufigkeit der Commitoperationen anhand der Anzahl der geschriebenen Zeilen festzulegen. Oder Sie können den INGEST-Standardkonfigurationsparameter **commit_period** verwenden, um die Häufigkeit der Commitoperationen anhand einer Zeitangabe festzulegen.
- **Unterstützung für das Kopieren zurückgewiesener Datensätze in eine Datei oder Tabelle oder das Löschen von diesen.** Sie können angeben, wie der Befehl **INGEST** mit Zeilen verfährt, die vom Dienstprogramm INGEST (mit dem Parameter **DUMPFILE**) oder von DB2 (mit dem Parameter **EXCEPTION TABLE**) zurückgewiesen wurden.
- **Unterstützung für Neustart und Recovery.** Standardmäßig können alle **INGEST**-Befehle vom letzten Commitpunkt aus erneut gestartet werden. Außerdem versucht das Dienstprogramm INGEST nach bestimmten Fehlern eine Recovery durchzuführen, wenn Sie den INGEST-Konfigurationsparameter **retry_count** angegeben haben.

Der Befehl **INGEST** unterstützt folgende Eingabedatenformate:

- Text mit Trennzeichen
- Positionsgebundener Text und Binärformat
- Spalten in verschiedenen Reihenfolgen und Formaten

Zusätzlich zu regulären Tabellen und Kurznamen unterstützt der Befehl **INGEST** folgende Tabellentypen:

- MDC-Tabellen (mehrdimensionales Clustering) und ITC-Tabellen (Clustering anhand der Einfügungszeit)
- Bereichspartitionierte Tabellen
- Bereichsclustertabellen (RCT-Tabellen)
- MQTs (Materialized Query Table, gespeicherte Abfragetabelle), für die MAINTAINED BY USER angegeben wurde, einschließlich der Übersichtstabellen
- temporale Tabellen
- aktualisierbare Sichten (außer typisierten Sichten)

Ein einzelner Befehl **INGEST** durchläuft drei wichtige Phasen:

1. Transport

Die Transporter lesen aus der Datenquelle und reihen Datensätze in die Warteschlangen des Formatierungsprogramms ein. Bei INSERT- und MERGE-Operationen gibt es pro Eingabequelle einen Transporter-Thread (beispielsweise einen Thread für jeweils eine Eingabedatei). Bei UPDATE- und DELETE-Operationen gibt es nur einen Transporter-Thread.

2. Format

Die Formatierungsprogramme führen für jeden Datensatz eine Syntaxanalyse durch, konvertieren die Daten in das für DB2-Datenbanksysteme erforderliche Format und reihen die einzeln formatierten Datensätze in die der Partition dieses Datensatzes entsprechende Warteschlange für Flusher ein. Die Anzahl der Formatierungsprogrammthreads wird durch den Konfigurationsparameter **num_formatters** angegeben. Der Standardwert lautet (Anzahl von logischen CPUs)/2.

3. Ausführen von Flushoperationen

Die Flusher setzen die SQL-Anweisungen ab, um die Operationen für die DB2-Tabellen auszuführen. Die Anzahl der Flusher pro Partition wird durch den Konfigurationsparameter **num_flushers_per_partition** angegeben. Der Standardwert lautet max(1, ((Anzahl logischer CPUs)/2)/(Anzahl Partitionen)).

Verwandte Tasks für INGEST - Übersicht

Dieser Abschnitt bietet eine allgemeine Übersicht der wichtigsten Konfigurations- und Betriebstasks, die in Verbindung mit dem Dienstprogramm INGEST stehen.

Einrichten von INGEST-Middleware

1. Entscheidung über den Einsatzort des Dienstprogramms INGEST

   Sie können INGEST-Jobs auf einem vorhandenen System oder auf einem eigenständigen System ausführen. Weitere Informationen finden Sie in „Entscheidung über den Einsatzort des Dienstprogramms INGEST" auf Seite 137.
2. Installieren Sie das Dienstprogramm INGEST (Teil von DB2 Data Server Runtime Client und von DB2 Data Server Client).

   Wenn Sie das Dienstprogramm INGEST auf einem neuen, eigenständigen System installieren wollen, führen Sie das Installationsprogramm für das Image der DB2-Clients aus. Weitere Informationen finden Sie unter „IBM Data Server-Clients-Installation (Linux, UNIX)" in *IBM Data Server-Clients - Installation*.

Entwickeln eines Prozesses zum Füllen einer Tabelle

1. (Falls erforderlich) Ermitteln von Codepagefehlern

   In Abhängigkeit davon, ob von den Eingabedaten, dem DB2-Client und dem DB2-Server dieselbe Codepage verwendet wird, müssen möglicherweise gewisse Benutzeraktionen vorgenommen werden, bevor der Befehl **INGEST** ausgeführt werden kann. Weitere Informationen hierzu finden Sie in „Codepageaspekte zum Dienstprogramm INGEST" auf Seite 153.
2. Durchführen eines Neustarts für fehlgeschlagene **INGEST**-Befehle - Konfiguration

   Damit eine Operation INGEST erneut startbar ist, müssen Sie eine Protokolltabelle für den Neustart erstellen, bevor Sie den Befehl **INGEST** ausgeben. Weitere Informationen hierzu finden Sie in „Erstellen einer Neustarttabelle" auf Seite 138.

3. Schreiben eines Befehls **INGEST**

   Setzen Sie den Befehl **INGEST** mit den obligatorischen Parametern wie z. B. der Eingabequelle und dem Datentyp sowie verschiedenen optionalen Parametern ab. Eine detaillierte Beschreibung der Befehlssyntax und -verwendung finden Sie zusammen mit Beispielen in „Daten einpflegen" auf Seite 139 und im Abschnitt „INGEST" des Handbuchs *Command Reference*.

4. Verarbeiten eines kontinuierlichen Datenstroms von INGEST-Jobs - Konfiguration

   Wenn Sie ohne großen Aufwand einen bereits geschriebenen Befehl **INGEST** aufrufen wollen, erstellen Sie ein Script für den Befehl und rufen Sie es bei Erfordernis auf. Weitere Informationen finden Sie in „Szenario: Verarbeiten eines Dateienstroms mit dem Dienstprogramm INGEST" auf Seite 158.

Ausführen von Betriebstasks

- (Falls erforderlich) Beheben eines fehlgeschlagenen Befehls **INGEST**

  Wenn ein Job INGEST fehlschlägt, können Sie zwischen der Option, den Befehl erneut zu starten, und der Option, den Befehl zu beenden, wählen. Weitere Informationen finden Sie in „Erneutes Starten einer fehlgeschlagenen INGEST-Operation" auf Seite 146 oder in „Beenden einer fehlgeschlagenen Operation INGEST" auf Seite 149.

- Überwachen eines Befehls **INGEST**

  Weitere Informationen hierzu finden Sie in „Überwachen von INGEST-Operationen" auf Seite 150.

(Optional) Optimieren der Leistung

- Überprüfen optimierbarer Konfigurationsparameter für den Befehl **INGEST**
- Ändern des Befehls **INGEST** zur Erfüllung von Hochleistungsanforderungen

  Weitere Informationen finden Sie in „Leistungsaspekte von INGEST-Operationen" auf Seite 153.

Entscheidung über den Einsatzort des Dienstprogramms INGEST

Das Dienstprogramm INGEST ist ein Teil der DB2-Clientinstallation. Sie können es auf dem Client oder dem Server ausführen.

Informationen zu diesem Vorgang

Es gibt zwei Auswahlmöglichkeiten für den Einsatzort des Dienstprogramms INGEST:

Auf einem vorhandenen Server in der Data-Warehouse-Umgebung
Bei dieser Konfigurationsart gibt es zwei Auswahlmöglichkeiten für das Ausführen von INGEST-Jobs:

- Auf der DB2-Koordinatorpartition (Datenbankpartitionsserver, zu dem Anwendungen eine Verbindung herstellen und auf dem sich der koordinierende Agent befindet).
- Auf einem vorhandenen ETL-Server (ETL = Extract, Transform, and Load).

Auf einem neuen Server
Bei dieser Konfigurationsart gibt es zwei Auswahlmöglichkeiten für das Ausführen von INGEST-Jobs:

- Auf einem Server, der ausschließlich das Dienstprogramm INGEST ausführt.
- Auf einem Server, der außerdem Host für eine zusätzliche DB2-Koordinatorpartition ist, die dem Dienstprogramm INGEST zugeordnet ist.

Eine Reihe von Faktoren beeinflussen die Entscheidung, wo das Dienstprogramm INGEST installiert werden sollte:

- Leistung: Die Installation des Dienstprogramms INGEST auf einem eigenen Server bietet deutliche Vorteile hinsichtlich der Leistung, sodass diese Möglichkeit für Umgebungen mit großen Datenmengen geeignet ist.
- Kosten: Wenn das Dienstprogramm INGEST auf einem vorhandenen Server installiert wird, bedeutet dies, dass durch das Ausführen des Dienstprogramms INGEST keine Mehrkosten entstehen.
- Verwaltungskomfort

Erstellen einer Neustarttabelle

Standardmäßig sind fehlgeschlagene **INGEST**-Befehle ab dem letzten Commitpunkt neu startbar. Sie müssen jedoch zuerst eine Neustarttabelle erstellen, in der die Informationen gespeichert sind, die zur Wiederaufnahme des Befehls **INGEST** benötigt werden.

Informationen zu diesem Vorgang

Sie müssen die Neustarttabelle nur einmal erstellen. Danach wird sie von allen **INGEST**-Befehlen in der Datenbank verwendet.

Das Dienstprogramm INGEST verwendet diese Tabelle zum Speichern von Informationen, die erforderlich sind, um einen nicht ordnungsgemäß beendeten Befehl INGEST am letzten Commitpunkt fortzusetzen.

Anmerkung: Die Neustarttabelle enthält keine Kopien der Eingabezeilen, sondern nur einige Zähler, die angeben, welche Zeilen festgeschrieben wurden.

Einschränkungen

- Es empfiehlt sich, die Neustarttabelle in demselben Tabellenbereich zu speichern wie die Zieltabellen, die vom Dienstprogramm INGEST aktualisiert werden. Falls dies nicht möglich ist, müssen Sie sicherstellen, dass der Tabellenbereich, in dem sich die Neustarttabelle befindet, auf demselben Stand ist wie der Tabellenbereich mit der Zieltabelle. Wenn Sie beispielsweise ein Restore oder eine aktualisierende Wiederherstellung für einen der Tabellenbereiche durchführen, müssen Sie für den anderen ebenfalls ein Restore oder eine aktualisierende Wiederherstellung auf denselben Stand durchführen. Haben die Tabellenbereiche nicht denselben Stand und Sie führen einen Befehl **INGEST** mit der Option **RESTART CONTINUE** aus, schlägt das Dienstprogramm INGEST möglicherweise fehl oder es pflegt falsche Daten ein.
- Wenn Sie im Rahmen Ihrer Strategie für die Disaster-Recovery die Zieltabellen von INGEST-Operationen replizieren, dann müssen Sie auch die Neustarttabelle replizieren, damit die Synchronisation mit den Zieltabellen erhalten bleibt.

Vorgehensweise

Gehen Sie wie folgt vor, um die Neustarttabelle zu erstellen:

- Wenn Sie einen Server der Version 10.1 verwenden, dann rufen Sie die gespeicherte Prozedur SYSPROC.SYSINSTALLOBJECTS auf:

```
db2 "CALL SYSPROC.SYSINSTALLOBJECTS('INGEST', 'C', tablespace-name, NULL)"
```

- Wenn Sie einen Server der Version 9.5, Version 9.7 oder Version 9.8 verwenden, dann setzen Sie die folgenden SQL-Anweisungen ab:

```
CREATE TABLE SYSTOOLS.INGESTRESTART (
  JOBID             VARCHAR(256) NOT NULL,
  APPLICATIONID     VARCHAR(256) NOT NULL,
  FLUSHERID         INT          NOT NULL,
  FLUSHERDISTID     INT          NOT NULL,
  TRANSPORTERID     INT          NOT NULL,
  BUFFERID          BIGINT       NOT NULL,
  BYTEPOS           BIGINT       NOT NULL,
  ROWSPROCESSED     INT          NOT NULL,
    PRIMARY KEY (JOBID, FLUSHERID, TRANSPORTERID, FLUSHERDISTID))
  IN <tabellenbereichsname>
  DISTRIBUTE BY (FLUSHERDISTID);

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE SYSTOOLS.INGESTRESTART TO PUBLIC;
```

Ergebnisse

Die Neustarttabelle SYSTOOLS.INGESTRESTART wird jetzt im angegebenen Tabellenbereich erstellt und Sie können nun neustartfähige **INGEST**-Befehle ausführen.

Beispiel

Ein Datenbankadministrator möchte alle **INGEST**-Befehle als neustartfähig ausführen und muss deshalb zuerst eine Neustarttabelle erstellen:

1. Der Datenbankadministrator stellt eine Verbindung zur Datenbank her:
   ```
   db2 CONNECT TO sample
   ```
2. Der Datenbankadministrator ruft die gespeicherte Prozedur auf:
   ```
   db2 "CALL SYSPROC.SYSINSTALLOBJECTS('INGEST', 'C', NULL, NULL)"
   ```

Nächste Schritte

Vergewissern Sie sich, dass alle Benutzer, die die Neustarttabelle ändern, über die entsprechende Berechtigung verfügen:

- Wird im Befehl **INGEST** die Option `RESTART NEW` angegeben, dann muss der Benutzer über die Berechtigungen SELECT, INSERT, UPDATE und DELETE für die Neustarttabelle verfügen.
- Wenn im Befehl **INGEST** die Option `RESTART TERMINATE` angegeben ist, dann muss der Benutzer über die Berechtigungen SELECT und DELETE für die Neustarttabelle verfügen.

Daten einpflegen

Mit dem Dienstprogramm INGEST können Sie mithilfe der SQL-ARRAY-Operationen zum Einfügen, Aktualisieren und Löschen kontinuierlich Daten in DB2-Tabellen weiterleiten bis keine Daten mehr in den Quellen vorhanden sind.

Vorbereitende Schritte

Bevor Sie das Dienstprogramm INGEST ausführen, müssen Sie mit der Datenbank, in die die Daten importiert werden, verbunden sein.

Standardmäßig sind fehlgeschlagene **INGEST**-Befehle ab dem letzten Commitpunkt neu startbar. Sie müssen jedoch zuerst eine Neustarttabelle erstellen, andernfalls erhalten Sie eine Fehlernachricht, in der Sie darauf hingewiesen werden, dass der ausgegebene Befehl nicht neu startbar ist. Das Dienstprogramm INGEST verwendet

diese Tabelle, um Informationen zu speichern, die erforderlich sind, um einen nicht ordnungsgemäß beendeten Befehl **INGEST** am letzten Commitpunkt fortzusetzen. Weitere Informationen hierzu finden Sie in „Erstellen einer Neustarttabelle" auf Seite 138.

Informationen zu diesem Vorgang

Eine Liste der erforderlichen Zugriffsrechte und Berechtigungen finden Sie im Abschnitt zu den Berechtigungen für den Befehl **INGEST**.

Einschränkungen

Eine umfassende Liste mit Einschränkungen für das Dienstprogramm INGEST finden Sie in „Dienstprogramm INGEST - Einschränkungen und Begrenzungen" auf Seite 151.

Vorgehensweise

Setzen Sie den Befehl **INGEST** ab und geben Sie dabei mindestens eine Quelle, das Format und die Zieltabelle an, wie im folgenden Beispiel dargestellt:

```
INGEST FROM FILE my_file.txt
    FORMAT DELIMITED
    INSERT INTO my_table;
```

Es empfiehlt sich, auch eine Zeichenfolge mit dem Parameter **RESTART NEW** für den Befehl **INGEST** anzugeben:

```
INGEST FROM FILE my_file.txt
   FORMAT DELIMITED
   RESTART NEW 'CDIjob001'
   INSERT INTO my_table;
```

Die Zeichenfolge, die Sie angeben, kann maximal 128 Byte lang sein. Da die Zeichenfolge den Befehl **INGEST** eindeutig identifiziert, muss sie in der aktuellen Datenbank für alle **INGEST**-Befehle, für die die Option **RESTART NEW** angegeben ist und die noch nicht beendet wurden, eindeutig sein.

Beispiel

Basisbeispiele für INGEST

In dem folgenden Beispiel werden Daten aus einer Textdatei mit Trennzeichen eingefügt:

```
INGEST FROM FILE my_file.txt
    FORMAT DELIMITED
    INSERT INTO my_table;
```

In dem folgenden Beispiel werden Daten aus einer Textdatei mit Trennzeichen mit Feldern eingefügt, die durch Komma voneinander getrennt sind (dies ist der Standard). Die Felder in der Datei entsprechen den Tabellenspalten.

```
INGEST FROM FILE my_file.txt
   FORMAT DELIMITED
   (
      $field1 INTEGER EXTERNAL,
      $field2 DATE 'mm/dd/yyyy',
      $field3 CHAR(32)
   )
   INSERT INTO my_table
      VALUES($field1, $field2, $field3);
```

Außerkraftsetzen des Begrenzers - Beispiel

Im folgenden Beispiel werden wie im vorherigen Beispiel Daten eingefügt, aber die Felder sind durch einen vertikalen Balken voneinander getrennt.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED by '|'
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
 )
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Übergehen der Felddefinitionsliste und der Liste VALUES - Beispiel

Im folgenden Beispiel wird die Tabelle wie folgt definiert:

```
CREATE TABLE my_table (
    c1 VARCHAR(32),
    c2 INTEGER GENERATED BY DEFAULT AS IDENTITY,
    c3 INTEGER GENERATED ALWAYS AS (c2 + 1),
   );
```

Der Benutzer setzt den folgenden Befehl **INGEST** ab:

```
INGEST FROM FILE my_file.txt
    FORMAT DELIMITED
    INSERT INTO mytable;
```

- Die Liste mit den Standardfelddefinitionen sieht folgendermaßen aus:

  ```
  (
      $C1 CHARACTER(32),
      $C2 INTEGER EXTERNAL,
      $C3 INTEGER EXTERNAL
  )
  ```

- Die Standardliste VALUES für die Anweisung INSERT hat folgendes Format:

  ```
  VALUES($C1, $C2, DEFAULT)
  ```

  Beachten Sie, dass der dritte Wert DEFAULT lautet, da die Feld $C3 entsprechende Spalte als GENERATED ALWAYS definiert ist. Der vierte Wert wird übergangen, da er über kein Feld verfügt.

Zusätzliche Felder, die zum Berechnen von Spaltenwerten verwendet werden - Beispiel

Das folgende Beispiel ist dasselbe wie das Beispiel für das Außerkraftsetzen des Begrenzers, jedoch entsprechen nur die ersten beiden Felder den ersten beiden Tabellenspalten (PROD_ID und DESCRIPTION), während der Wert für die dritte Tabellenspalte (TOTAL_PRICE) aus den verbleibenden drei Feldern berechnet wird.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED BY '|'
 (
  $prod_ID     CHAR(8),
  $description CHAR(32),
  $price       DECIMAL(5,2) EXTERNAL,
  $sales_tax   DECIMAL(4,2) EXTERNAL,
  $shipping    DECIMAL(3,2) EXTERNAL
 )
 INSERT INTO my_table(prod_ID, description, total_price)
  VALUES($prod_id, $description, $price + $sales_tax + $shipping);
```

Füllfelder - Beispiel

In dem folgenden Beispiel werden Daten aus einer Textdatei mit Trennzeichen eingefügt, deren Felder durch Komma voneinander getrennt sind

(dies ist der Standard). Die Felder in der Datei entsprechen den Tabellenspalten, außer dass zwischen den Feldern für die Spalten 2 und 3 bzw. die Spalten 3 und 4 zusätzliche Felder vorhanden sind.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $field1  INTEGER,
  $field2  CHAR(8),
  $filler1 CHAR,
  $field3  CHAR(32),
  $filler2 CHAR,
  $field4  DATE
 )
 INSERT INTO my_table VALUES($field1, $field2, $field3, $field4);
```

Änderungswerte für Format - Beispiel

In dem folgenden Beispiel werden Daten aus einer Textdatei mit Trennzeichen in Codepage 850 eingefügt. Die Datumsfelder sind in US-amerikanischem Format und die Zeichenfelder sind in Gleichheitszeichen eingeschlossen.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 INPUT CODEPAGE 850
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32) ENCLOSED BY '='
 )
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Positionsgebunden - Beispiel

In dem folgenden Beispiel werden Daten aus einer Datei mit Feldern an den angegebenen Positionen eingefügt. Die Felder in der Datei entsprechen den Tabellenspalten.

```
INGEST FROM FILE my_file.txt
    FORMAT POSITIONAL
 (
  $field1 POSITION(1:8)   INTEGER EXTERNAL,
  $field2 POSITION(10:19) DATE 'yyyy-mm-dd',
  $field3 POSITION(25:34) CHAR(10)
 )
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

DEFAULTIF - Beispiele

Dieses Beispiel ähnelt dem vorherigen Beispiel, außer dass das Dienstprogramm INGEST den Standardwert einfügt, wenn das zweite Feld mit einem Leerzeichen beginnt.

```
INGEST FROM FILE my_file.txt
 FORMAT POSITIONAL
 (
  $field1 POSITION(1:8) INTEGER EXTERNAL,
  $field2 POSITION(10:19) DATE 'yyyy-mm-dd' DEFAULTIF = ' ',
    $field3 POSITION(25:34) CHAR(10)
 )
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Dieses Beispiel ist dasselbe wie das vorherige Beispiel, außer dass sich der Standardanzeiger in der Spalte nach den Datenspalten befindet.

```
INGEST FROM FILE my_file.txt
 FORMAT POSITIONAL
 (
  $field1 POSITION(1:8)   INTEGER EXTERNAL,
  $field2 POSITION(10:19) DATE 'yyyy-mm-dd' DEFAULTIF(35) = ' ',
    $field3 POSITION(25:34) CHAR(10)
 )
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Mehrere Eingabequellen - Beispiel

In diesem Beispiel werden Daten aus drei Texten mit Trennzeichen eingefügt:

```
INGEST FROM FILE my_file.txt, my_file2.txt, my_file3.txt
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
 )
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Pipe - Beispiel

In diesem Beispiel werden Daten aus einer Pipe eingefügt:

```
INGEST FROM PIPE my_pipe
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
 )
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Optionen - Beispiel

In diesem Beispiel werden Daten aus einer Textdatei mit Trennzeichen eingefügt, deren Felder durch Komma voneinander getrennt sind (dies ist der Standard). Die Felder in der Datei entsprechen den Tabellenspalten. Der Befehl gibt an, dass zu schreibende Zeilen, die von DB2 zurückgewiesen werden (z. B. aufgrund von ungültigen Integritätsbedingungen), in die Tabelle EXCP_TABLE zu schreiben sind, dass Zeilen, die aufgrund anderer Fehler zurückgewiesen werden, zu verwerfen sind und dass Nachrichten in die Datei `messages.txt` zu schreiben sind.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
 )
 EXCEPTION TABLE excp_table
    MESSAGES messages.txt
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Neustart - Beispiel

In diesem Beispiel wird ein Befehl **INGEST** (der standardmäßig erneut startbar ist) abgesetzt, für den eine ID für den INGEST-Job angegeben ist:

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
```

```
 $field3 CHAR(32)
)
RESTART NEW 'ingestcommand001'
INSERT INTO my_table
 VALUES($field1, $field2, $field3);
```

Wenn der Befehl vor dem Beenden fehlschlägt, können Sie ihn mit dem folgenden Befehl erneut starten:

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
 )
 RESTART CONTINUE 'ingestcommand001'
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Neustart beenden - Beispiel

In diesem Beispiel wird derselbe Befehl **INGEST** ausgegeben wie im oben stehenden Beispiel "Neustart - Beispiel":

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
 )
 RESTART NEW 'ingestcommand001'
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Wenn der Befehl vor dem Beenden fehlschlägt und Sie nicht planen, ihn erneut zu starten, können Sie die Datensätze für den Neustart mit dem folgenden Befehl bereinigen:

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
 )
 RESTART TERMINATE 'ingestcommand001'
 INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

Nach Absetzen dieses Befehls können Sie den Befehl **INGEST** nicht mehr mit der Job-ID "ingestcommand001" erneut starten, aber Sie können diese Zeichenfolge für den Parameter RESTART NEW eines neuen Befehls **INGEST** wiederverwenden.

Spalten neu ordnen - Beispiel

In diesem Beispiel werden Daten aus einer Textdatei mit Trennzeichen eingefügt, deren Felder durch Komma voneinander getrennt sind. Die Tabelle weist drei Spalten auf und die Felder in den Eingabedaten haben gegenüber den Tabellenspalten eine umgekehrte Reihenfolge.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
```

```
 $field3 CHAR(32)
)
INSERT INTO my_table
 VALUES($field3, $field2, $field1);
```

Basisbeispiele für UPDATE, MERGE und DELETE

In den folgenden Beispielen werden die Tabellenzeilen aktualisiert, deren jeweiliger Primärschlüssel mit den entsprechenden Feldern in der Eingabedatei übereinstimmt.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $key1  INTEGER EXTERNAL,
  $key2  INTEGER EXTERNAL,
  $data1 CHAR(8),
  $data2 CHAR(32),
  $data3 DECIMAL(5,2) EXTERNAL  )
 UPDATE my_table
  SET (data1, data2, data3) = ($data1, $data2, $data3)
  WHERE (key1 = $key1) AND (key2 = $key2);
```

oder

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $key1  INTEGER EXTERNAL,
  $key2  INTEGER EXTERNAL,
  $data1 CHAR(8),
  $data2 CHAR(32),
  $data3 DECIMAL(5,2) EXTERNAL  )
 UPDATE my_table
  SET data1 = $data1, data2 = $data2, data3 = $data3
  WHERE (key1 = $key1) AND (key2 = $key2);
```

In diesem Beispiel werden Daten aus der Eingabedatei mit Daten in der Zieltabelle zusammengeführt. Wenn die Primärschlüsselfelder von Eingabezeilen mit einer Tabellenzeile übereinstimmen, so wird durch dieses Beispiel die Tabellenzeile mit der Eingabezeile aktualisiert. Bei den übrigen Eingabezeilen wird die Zeile der Tabelle hinzugefügt.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $key1  INTEGER EXTERNAL,
  $key2  INTEGER EXTERNAL,
  $data1 CHAR(8),
  $data2 CHAR(32),
  $data3 DECIMAL(5,2) EXTERNAL  )
 MERGE INTO my_table
  ON (key1 = $key1) AND (key2 = $key2)
  WHEN MATCHED THEN
   UPDATE SET (data1, data2, data3) = ($data1, $data2, $data3)
  WHEN NOT MATCHED THEN
   INSERT VALUES($key1, $key2, $data1, $data2, $data3);
```

In diesem Beispiel werden die Tabellenzeilen gelöscht, deren Primärschlüssel mit den ensprechenden Primärschlüsselfeldern in der Eingabedatei übereinstimmen.

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $key1  INTEGER EXTERNAL,
  $key2  INTEGER EXTERNAL
```

```
 )
DELETE FROM my_table
 WHERE (key1 = $key1) AND (key2 = $key2);
```

Komplexe SQL - Beispiele

Betrachten Sie das folgende Beispiel, in dem eine Tabelle mit den Spalten KEY, DATA und ACTION vorhanden ist. Durch den folgenden Befehl wird die Spalte DATA von Tabellenzeilen aktualisiert, in denen die Primärschlüsselspalte (KEY) mit dem entsprechenden Feld in der Eingabedatei übereinstimmt und die Spalte ACTION den Wert 'U' aufweist:

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $key_fld  INTEGER EXTERNAL,
  $data_fld INTEGER EXTERNAL
 )
 UPDATE my_table
  SET data = $data_fld
   WHERE (key = $key_fld) AND (action = 'U');
```

Das folgende Beispiel ist dasselbe wie das vorherige Beispiel, außer dass hier die Zeile aus der Tabelle gelöscht wird, wenn die Schlüssel übereinstimmen und der Wert für die Spalte ACTION 'D' lautet:

```
INGEST FROM FILE my_file.txt
 FORMAT DELIMITED
 (
  $key_fld  INTEGER EXTERNAL,
  $data_fld INTEGER EXTERNAL
 )
 MERGE INTO my_table
  ON (keyl = $key_fld)
  WHEN MATCHED AND (action = 'U') THEN
   UPDATE SET data = $data_fld
  WHEN MATCHED AND (action = 'D') THEN
   DELETE;
```

Nächste Schritte

Wenn der Befehl **INGEST** erfolgreich abgeschlossen wird, können Sie die für den Parameter **RESTART NEW** angegebene Zeichenfolge wiederverwenden.

Wenn der Befehl **INGEST** fehlschlägt und Sie ihn erneut starten wollen, müssen Sie die Option **RESTART CONTINUE** mit der Zeichenfolge angeben, die Sie im ursprünglichen Befehl angegeben haben.

Wenn Sie den fehlgeschlagenen Befehl **INGEST** nicht erneut starten wollen und die Einträge in der Neustarttabelle bereinigt werden sollen, führen Sie den Befehl **INGEST** erneut aus und geben Sie dabei die Option **RESTART TERMINATE** an.

Erneutes Starten einer fehlgeschlagenen INGEST-Operation:

Wenn ein Befehl **INGEST** vor dem ordnungsgemäßen Beenden fehlschlägt und Sie den Befehl erneut starten wollen, müssen Sie den Befehl **INGEST** mit der Option RESTART CONTINUE erneut absetzen. Dieser zweite Befehl **INGEST** beginnt am letzten Commitpunkt und kann ebenfalls erneut gestartet werden.

Vorbereitende Schritte

Die Benutzer-ID, mit der der fehlgeschlagene Befehl **INGEST** erneut gestartet wird, muss für die Protokolltabelle für den Neustart über die Zugriffsrechte SELECT, INSERT, UPDATE und DELETE verfügen.

Informationen zu diesem Vorgang

Das Dienstprogramm **INGEST** betrachtet einen Befehl als ordnungsgemäß beendet, wenn es das Ende der Datei oder der Pipe erreicht. Bei Vorhandensein einer beliebigen anderen Bedingung betrachtet das Dienstprogramm **INGEST** den Befehl als nicht beendet. Dazu können gehören:

- Für den Befehl **INGEST** wird beim Lesen der Eingabedatei oder Pipe ein E/A-Fehler ausgegeben.
- Für den Befehl **INGEST** wird vom DB2-Datenbanksystem ein kritischer Systemfehler ausgegeben.
- Für den Befehl **INGEST** wird ein DB2-Datenbanksystemfehler ausgegeben, durch den möglicherweise verhindert wird, dass weitere SQL-Anweisungen im Befehl **INGEST** erfolgreich ausgeführt werden können (wenn beispielsweise die Tabelle nicht mehr vorhanden ist).
- Der Befehl **INGEST** wird abgebrochen oder abnormal beendet.

Einschränkungen

- Wenn sich die Zieltabelle und die Neustarttabelle in verschiedenen Tabellenbereichen befinden, müssen die beiden Tabellenbereiche im Hinblick auf aktualisierende Recovery- oder Restoreoperationen auf demselben Stand sein.
- Der Inhalt der Neustarttabelle kann nicht geändert werden. Sie können lediglich für die gesamte Tabelle einen Restore durchführen, um die Synchronisation mit der Zieltabelle zu gewährleisten.
- Der Konfigurationsparameter **num_flushers_per_partition** muss derselbe sein wie für den ursprünglichen Befehl.
- Wenn die Eingabe aus Dateien oder Pipes erfolgt, muss die Anzahl Eingabedateien oder Pipes dieselbe sein wie für den ursprünglichen Befehl.
- Die Eingabedateien oder Pipes müssen dieselben Datensätze in derselben Reihenfolge bereitstellen wie für den ursprünglichen Befehl.
- Die folgenden Parameter des Befehls **INGEST** müssen dieselben sein wie für den ursprünglichen Befehl.
  - Eingabetyp (Datei oder Pipe)
  - SQL-Anweisung
  - Felddefinitionsliste einschließlich der Anzahl Felder und aller Feldattribute
- Die Spalten der Zieltabelle, die von dem SQL-Befehl aktualisiert werden, müssen dieselbe Definition aufweisen wie zur Zeit des ursprünglichen Befehls.
- In einer Umgebung mit partitionierten Datenbanken dürfen Sie keine Datenbankpartitionen hinzugefügt oder entfernt haben.
- In einer Umgebung mit partitionierten Datenbanken dürfen Sie keine Daten partitionsübergreifend umverteilt haben.
- Wenn für einen Befehl **INGEST** der Parameter DUMPFILE (BADFILE) angegeben wird, ist die Speicherauszugsdatei nur garantiert vollständig, wenn der Befehl **INGEST** in einer einzigen Ausführung normal abgeschlossen wird. Wenn ein Befehl **INGEST** fehlschlägt und ein erneut gestarteter Befehl erfolgreich ausgeführt

wird, fehlen in der Kombination der Speicherauszugsdateien von den beiden Befehlen möglicherweise einige Datensätze bzw. es sind doppelte Datensätze enthalten.

Wenn die dritte, vierte, fünfte oder neunte Einschränkung nicht eingehalten wird, gibt das Dienstprogramm INGEST einen Fehler aus und beendet den Befehl **INGEST**. Im Fall der anderen Einschränkungen gibt das Dienstprogramm INGEST keinen Fehler aus, aber der erneut gestartete Befehl **INGEST** generiert möglicherweise andere Ausgabezeilen als der ursprüngliche Befehl, wäre dieser ordnungsgemäß beendet worden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine fehlgeschlagene Operation **INGEST** erneut zu starten:

1. Verwenden Sie die verfügbaren Informationen, um das Problem, das zum Fehlschlagen geführt hat, zu diagnostizieren und zu beheben.
2. Setzen Sie den Befehl **INGEST** erneut ab und geben Sie dabei die Option **RESTART CONTINUE** mit der entsprechenden Job-ID an.

Ergebnisse

Sobald der erneut gestartete Befehl **INGEST** ordnungsgemäß beendet wird, können Sie die Job-ID für einen späteren Befehl **INGEST** wiederverwenden.

Beispiel

Der folgende Befehl **INGEST** ist fehlgeschlagen:

```
INGEST FROM FILE my_file.txt
   FORMAT DELIMITED
   (
      $field1 INTEGER EXTERNAL,
      $field2 DATE 'mm/dd/yyyy',
      $field3 CHAR(32)
   )
   RESTART NEW 'ingestjob001'
   INSERT INTO my_table
      VALUES($field1, $field2, $field3);
```

Der Datenbankadministrator korrigiert das Problem, durch das das Fehlschlagen verursacht wurde, und startet den Befehl **INGEST** (der am letzten Commitpunkt beginnt) mit dem folgenden Befehl erneut:

```
INGEST FROM FILE my_file.txt
   FORMAT DELIMITED
   (
      $field1 INTEGER EXTERNAL,
      $field2 DATE 'mm/dd/yyyy',
      $field3 CHAR(32)
   )
   RESTART CONTINUE 'ingestjob001'
   INSERT INTO my_table
      VALUES($field1, $field2, $field3);
```

Beenden einer fehlgeschlagenen Operation INGEST:

Wenn der Befehl **INGEST** vor dem ordnungsgemäßen Beenden fehlschlägt und Sie ihn nicht erneut starten wollen, setzen Sie den Befehl **INGEST** erneut mit der Option **RESTART TERMINATE** ab. Durch diese Befehlsoption wird der fehlgeschlagene Befehl **INGEST** aus den Protokollsätzen bereinigt.

Vorbereitende Schritte

Die Benutzer-ID, mit der der fehlgeschlagene Befehl **INGEST** beendet wird, muss für die Protokolltabelle für den Neustart über die Zugriffsrechte SELECT und DELETE verfügen.

Vorgehensweise

Setzen Sie den Befehl **INGEST** erneut ab, um eine fehlgeschlagene Operation **INGEST** zu beenden. Geben Sie den Parameter **RESTART TERMINATE** mit der entsprechenden Zeichenfolge an.

Ergebnisse

Nachdem der erneut gestartete Befehl **INGEST** ordnungsgemäß beendet wurde, können Sie die Zeichenfolge **RESTART NEW** für einen späteren Befehl **INGEST** verwenden.

Beispiel

Der folgende Befehl **INGEST** ist fehlgeschlagen:

```
INGEST FROM FILE my_file.txt
   FORMAT DELIMITED
   (
      $field1 INTEGER EXTERNAL,
      $field2 DATE 'mm/dd/yyyy',
      $field3 CHAR(32)
   )
   RESTART NEW 'ingestjob001'
   INSERT INTO my_table
      VALUES($field1, $field2, $field3);
```

Der Datenbankadministrator möchte den Befehl **INGEST** nicht erneut starten und beendet ihn deshalb mit dem folgenden Befehl (mit dem Parameter **RESTART TERMINATE**):

```
INGEST FROM FILE my_file.txt
   FORMAT DELIMITED
   (
      $field1 INTEGER EXTERNAL,
      $field2 DATE 'mm/dd/yyyy',
      $field3 CHAR(32)
   )
   RESTART TERMINATE 'ingestjob001'
   INSERT INTO my_table
      VALUES($field1, $field2, $field3);
```

Überwachen von INGEST-Operationen

Sie können den Befehl **INGEST LIST** oder den Befehl **INGEST GET STATS** verwenden, um den Fortschritt von **INGEST**-Befehlen zu überwachen.

Vorbereitende Schritte

Zum Absetzen der Befehle **INGEST LIST** und **INGEST GET STATS** benötigen Sie eine separate CLP-Sitzung, die aber auf demselben System ausgeführt werden muss wie der Befehl **INGEST**.

Vorgehensweise

Es gibt viele verschiedene Möglichkeiten, eine Operation INGEST zu überwachen:

- Zum Abrufen von Basisinformationen zu allen momentan aktiven **INGEST**-Befehlen verwenden Sie den Befehl **INGEST LIST**.
- Zum Abrufen von Detailinformationen zu einem bestimmten Befehl **INGEST** oder zu allen momentan aktiven **INGEST**-Befehlen verwenden Sie den Befehl **INGEST GET STATS**.
- Sie können auch die folgenden Monitorelemente abfragen, indem Sie eine Schnittstelle wie z. B. die Tabellenfunktion MON_GET_CONNECTION verwenden:
  - **client_acctng**
  - **client_applname**
  - **appl_name**
  - **client_userid**
  - **client_wrkstnname**

Beispiel

Das folgende Beispiel zeigt, wie die Ausgabe eines Befehls **INGEST LIST** aussehen kann:

```
INGEST LIST

Ingest job ID       = DB21000:20101116.123456.234567:34567:45678
Ingest temp job ID  = 1
Database Name       = MYDB
Input type          = FILE
Target table        = MY_SCHEMA.MY_TABLE
Start Time          = 04/10/2010 11:54:45.773215
Running Time        = 01:02:03
Number of records processed = 30,000
```

Das folgende Beispiel zeigt, wie die Ausgabe eines Befehls **INGEST GET STATS** aussehen kann:

```
INGEST GET STATS FOR 4

Ingest job ID = DB21000:20101116.123456.234567:34567:4567
Database      = MYDB
Target table  = MY_SCHEMA.MY_TABLE1

Records/sec      Flushes/sec       Records/sec     Flushes/sec
since start      since start       since last      since last       Total records
---------------- ----------------- --------------- ---------------- -------------
54321            65432             76543           87654            98765
```

Im folgenden Beispiel wird die Verwendung der Tabellenfunktion MON_GET-_CONNECTION zum Abrufen der Anzahl geänderter Zeilen und der Anzahl Commitoperationen dargestellt:

```
SELECT client_acctng AS "Job ID",
      SUM(rows_modified) AS "Total rows modified",
      SUM(total_app_commits) AS "Total commits"
   FROM TABLE(MON_GET_CONNECTION(NULL, NULL))
   WHERE application_name = 'DB2_INGEST'
   GROUP BY client_acctng
   ORDER BY 1

Job ID                                          Total rows modified Total commits
----------------------------------------------- ------------------- -------------
DB21000:20101116.123456.234567:34567:45678                       92            52
DB21000:20101116.987654.234567:34567:45678                      172           132

  2 record(s) selected.
```

Dienstprogramm INGEST - Einschränkungen und Begrenzungen

Es gibt eine Reihe von Einschränkungen, die Sie bei Verwenden des Dienstprogramms INGEST kennen sollten.

Möglichkeit zum Neustart

- Wenn sich der Typ der Eingabedatenquellen geändert hat, kann das Dienstprogramm INGEST die Änderung möglicherweise nicht erkennen und erstellt andere Ausgabezeilen als der ursprünglich fehlgeschlagene Befehl.

Tabellenunterstützung

- Das Dienstprogramm INGEST unterstützt nur Operationen für Tabellen unter DB2 for Linux, UNIX and Windows.
- Das Dienstprogramm INGEST unterstützt keine Operationen für:
  - Erstellte oder deklarierte temporäre Dateien
  - Typisierte Tabellen
  - Typisierte Sichten

Eingabetypen, -formate und Spaltentypen

- Das Dienstprogramm INGEST bietet keine Unterstützung für die folgenden Spaltentypen:
  - LOB-Typen (LOB, BLOB, CLOB, DBCLOB)
  - XML
  - Strukturierte Typen
  - Spalten mit einem benutzerdefinierten Datentyp auf der Basis aller zuvor aufgelisteten Typen
- Außerdem gelten für das Dienstprogramm INGEST die folgenden Einschränkungen bei generierten Spalten:
  - Das Dienstprogramm INGEST kann einer als GENERATED ALWAYS definierten Spalte keinen Wert zuweisen. Wenn für den Befehl **INGEST** die SQL-Anweisung INSERT oder UPDATE angegeben ist und die Zieltabelle eine als GENERATED ALWAYS definierte Spalte enthält, schlägt die INSERT- oder UPDATE-Operation fehl (SQL0798N) und der Befehl **INGEST** wird beendet, sofern Sie nicht eine der folgenden Aktionen ausführen:

- Schließen Sie die Spalte aus der Liste der zu aktualisierenden Spalten aus.
- Geben Sie bei der INSERT- oder UPDATE-Anweisung DEFAULT als den der Spalte zugewiesenen Wert an.

– Das Dienstprogramm INGEST kann einer als GENERATED BY DEFAULT AS IDENTITY definierten Spalte keine Kombination aus Standardwerten und spezifischen Werten zuweisen. Wenn für den Befehl **INGEST** die SQL-Anweisung INSERT oder UPDATE angegeben ist und die Zieltabelle eine als GENERATED BY DEFAULT AS IDENTITY definierte Spalte enthält, schlägt die INSERT- oder UPDATE-Operation fehl (SQL0407N) und der Befehl **INGEST** weist den Datensatz zurück, sofern Sie nicht eine der folgenden Aktionen ausführen:
  - Schließen Sie die Spalte aus der Liste der zu aktualisierenden Spalten aus.
  - Geben Sie bei der INSERT- oder UPDATE-Anweisung DEFAULT als den der Spalte zugewiesenen Wert an.
  - Geben Sie als den der Spalte zugewiesenen Wert einen Ausdruck an, der nie als NULL ausgewertet wird. Beispiel: Wenn der Ausdruck "$field1" lautet, kann "$field1" nie einen Wert NULL in den Eingabedatensätzen enthalten.

Einschränkungen bei der Verwendung anderer DB2-Funktionen mit dem Dienstprogramm INGEST

- Außer bei dem Parameter **CONNECT_MEMBER** hat der Befehl **SET CLIENT** (für Verbindungseinstellungen) keine Auswirkungen auf den Verbindungsaufbau des Dienstprogramms INGEST.
- Der Befehl **LIST HISTORY** zeigt keine INGEST-Operationen an.
- Der Befehl **SET UTIL_IMPACT_PRIORITY** hat keine Auswirkungen auf den Befehl **INGEST**.
- Der Konfigurationsparameter des Datenbankmanagers **util_impact_lim** hat keine Auswirkungen auf den Befehl **INGEST**.
- Außer bei CURRENT SCHEMA, CURRENT TEMPORAL SYSTEM_TIME und CURRENT TEMPORAL BUSINESS_TIME ignoriert das Dienstprogramm INGEST die Einstellungen der meisten Sonderregister, die Auswirkungen auf die Ausführung von SQL-Anweisungen haben.

Allgemeine Einschränkungen des Dienstprogramms INGEST

- Beim Einpflegen in eine Sicht mit mehreren Basistabellen müssen alle durch eine Sicherheitsrichtlinie geschützten Basistabellen durch dieselbe Sicherheitsrichtlinie geschützt sein. (Einige Basistabellen können weiterhin ungeschützt sein, aber alle geschützten Basistabellen müssen dieselbe Sicherheitsrichtlinie verwenden.)

Unterstützung für Kurznamen

- Wenn mit dem Befehl **INGEST** die Option `RESTART NEW` oder `RESTART CONTINUE` angegeben wird oder standardmäßig angegeben ist und es sich bei der Zieltabelle um einen Kurznamen oder eine aktualisierbare Sicht handelt, die einen Kurznamen aktualisiert, stellen Sie sicher, dass die Serveroption DB2_TWO_PHASE_COMMIT für die Serverdefinition, die den Kurznamen enthält, auf 'Y' gesetzt ist.
- Sie können mit **SET SERVER OPTION** kein zweiphasiges Commit aktivieren, bevor Sie den Befehl **INGEST** ausgeben, da der Befehl nur Auswirkungen auf die CLP-Verbindung hat, während der

Befehl **INGEST** seine eigene Verbindung herstellt. Sie müssen die Serveroption in der Serverdefinition im Katalog festlegen.

- Sie können die Serveroption DB2_TWO_PHASE_COMMIT nicht mit der Datenbankpartitionierungsfunktion verwenden, d. h., dass die Kombination aus Modus für Umgebungen mit partitionierten Datenbanken, einem erneut startbaren Befehl INGEST und dem Einpflegen in einen Kurznamen nicht unterstützt wird.
- Die Leistungsvorteile des Dienstprogramms sind geringer bei der Verwendung mit Kurznamen.

Weitere Aspekte von INGEST-Operationen

Leistungsaspekte von INGEST-Operationen

Mithilfe folgender Richtlinien können Sie die Leistung Ihrer INGEST-Jobs optimieren.

Feldtyp und Spaltentyp
Definieren Felder als denselben Typ wie ihre entsprechenden Spaltentypen. Wenn sich die Typen unterscheiden, muss das Dienstprogramm INGEST oder DB2 die Eingabedaten in den Spaltentyp konvertieren.

MQTs (Materialized Query Tables)
Wenn Sie Daten in eine Tabelle einpflegen, bei der es sich um eine Basistabelle einer als REFRESH IMMEDIATE definierten MQT handelt, kann sich die Leistung aufgrund der für die Aktualisierung der MQT benötigten Zeit bedeutend verschlechtern.

Zeilengröße
Bei Tabellen mit einer kleinen Zeilengröße erhöhen Sie den Wert des INGEST-Konfigurationsparameters **commit_count**. Bei Tabellen mit einer großen Zeilengröße legen Sie einen niedrigeren Wert für den INGEST-Konfigurationsparameter **commit_count** fest.

Andere Auslastungen
Wenn Sie das Dienstprogramm INGEST mit einer anderen Auslastung ausführen, erhöhen Sie den Wert des Datenbankkonfigurationsparameters **locklist** und legen Sie einen niedrigeren Wert für den INGEST-Konfigurationsparameter **commit_count** fest.

Codepageaspekte zum Dienstprogramm INGEST

Wenn das Dienstprogramm INGEST Eingabedaten verarbeitet, sind drei Codepages involviert: Die Codepage der Anwendung (Client), die Codepage der Eingabedaten und die Codepage der Datenbank.

Codepage	Art der Angabe	Standardwert
Codepage der Anwendung (Client), die in der Befehlsdatei des Befehlszeilenprozessors verwendet wird	Ermittelt anhand der aktuellen Ländereinstellung	Ermittelt anhand der aktuellen Ländereinstellung
Codepage der Eingabedaten	`INPUT CODEPAGE` im Befehl **INGEST**	Anwendungscodepage
Datenbankcodepage	Angegeben im Befehl **CREATE DATABASE**	1208 (UTF-8-Codierung von Unicode)

Wenn sich die Codepage der Eingabedaten von der Codepage der Anwendung unterscheidet, überschreibt das Dienstprogramm INGEST die Anwendungscodepage temporär mit der Codepage der Eingabedaten, so dass DB2 die Daten direkt von der Codepage der Eingabedaten in die Codepage der Datenbank konvertiert. Bei einigen Rahmenbedingungen kann das Dienstprogramm INGEST die Anwendungscodepage nicht überschreiben. In diesem Fall konvertiert das Dienstprogramm INGEST Zeichendaten, die nicht als FOR BIT DATA definiert sind, in die Anwendungscodepage, bevor sie an DB2 weitergeleitet werden. Wenn die Spalte nicht als FOR BIT DATA definiert ist, konvertiert DB2 die Daten in jedem Fall in die Codepage der Datenbank.

Codepage der CLP-Befehlsdatei
Außer für hexadezimale Konstanten setzt das Dienstprogramm INGEST voraus, dass der Text des Befehls **INGEST** in der Anwendungscodepage enthalten ist. Jedes Mal, wenn das Dienstprogramm INGEST Zeichenfolgen vergleichen muss, die für den Befehl **INGEST** angegeben wurden (z. B. beim Vergleich des Zeichens DEFAULTIF mit einem Zeichen in den Eingabedaten), führt das Dienstprogramm INGEST alle erforderlichen Codepagekonvertierungen durch, um sicherzustellen, dass die verglichenen Zeichenfolgen in derselben Codepage vorhanden sind. Weder das Dienstprogramm INGEST noch DB2 nehmen die Konvertierung von hexadezimalen Konstanten vor.

Codepage der Eingabedaten
Wenn sowohl ein Feld als auch die Tabellenspalte, der es zugeordnet ist, als FOR BIT DATA definiert sind, führt weder das Dienstprogramm INGEST noch DB2 Codepagekonvertierungen durch. Beispiel: Angenommen, durch den Befehl **INGEST** wird das Feld $c1 der Spalte C1 zugeordnet und beide werden als CHAR FOR BIT DATA definiert. Wenn das Eingabefeld X'E9' enthält, setzt DB2 die Spalte C1 auf X'E9', ungeachtet der Codepage der Eingabedaten oder der Datenbankcodepage.

Wenn eine Spaltendefinition FOR BIT DATA nicht enthält, wird sehr empfohlen, FOR BIT DATA auch in der Definition des zugehörigen Feldes wegzulassen. Wird in einer Spaltendefinition FOR BIT DATA angegeben, so sollte im zugehörigen Feld ebenso FOR BIT DATA angegeben werden. Andernfalls wird der der Spalte zugewiesene Wert unvorhersehbar, da er davon abhängt, ob das Dienstprogramm INGEST in der Lage ist, die Anwendungscodepage zu überschreiben.

Im folgenden Beispiel wird diese Situation beschrieben:

- Die Codepage der Eingabedaten lautet 819.
- Die Anwendungscodepage lautet 850.
- Die Datenbankcodepage lautet 1208 (UTF-8).
- Die Eingabedaten lauten "é" ("e" mit einem Akut), das X'E9' in Codepage 819, X'82' in Codepage 850 und X'C3A9' in UTF-8 entspricht.

In der folgenden Tabelle wird dargestellt, welche Daten schließlich im Server ankommen. Dies hängt davon ab, ob das Feld und/oder die Spalte als FOR BIT DATA definiert ist/sind und ob das Dienstprogramm INGEST die Anwendungscodepage überschreiben kann:

Tabelle 16. Mögliche Ergebnisse bei Definition der Feld- und der Spaltendefinition als FOR BIT DATA

Feld-definition	Spalten-definition	Eingabedaten (Codepage 819)	Daten nach der Konvertierung in Anwendungs-dungscodepage 850 durch das Dienst-programm INGEST	Daten auf dem Server, wenn das Dienstpro-gramm INGEST die Anwendungs-codepage über-schreiben kann	Daten auf dem Server, wenn das Dienstpro-gramm INGEST die Anwendungs-codepage nicht überschreiben kann
CHAR	CHAR	X'E9'	X'82'	X'C3A9'	X'C3A9'
CHAR FOR BIT DATA	CHAR FOR BIT DATA	X'E9'	X'E9'	X'E9'	X'E9'
CHAR FOR BIT DATA	CHAR	X'E9'	X'E9'	**X'C3A9'**	**X'C39A' ("Ú")**
CHAR	CHAR FOR BIT DATA	X'E9'	X'82'	**X'E9'**	**X'82'**

Wenn die Anwendungscodepage überschrieben werden kann, sendet das Dienstprogramm INGEST die in der vierten Spalte enthaltenen Daten an DB2. Wenn die Anwendungscodepage nicht überschrieben werden kann, sendet das Dienstprogramm INGEST die Daten in der vierten Spalte. Beachten Sie, dass die Ergebnisse, wie in der vorhergehenden Tabelle gezeigt, variieren können, wenn das Attribut FOR BIT DATA in der Feld- und der Spaltendefinition unterschiedlich ist.

Codepagefehler

In Fällen, in denen sich die Codepage der Eingabedaten und die Anwendungscodepage oder die Datenbankcodepage unterscheiden, führen das Dienstprogramm INGEST und/oder DB2 eine Codepagekonvertierung durch. Wenn DB2 die Codepagekonvertierung in einem der folgenden Fälle nicht unterstützt, gibt das Dienstprogramm INGEST einen Fehler aus und der Befehl wird beendet.

Konvertierung ist erforderlich, wenn...	In diesem Fall Konvertierung von...	In...	Ausgeführt durch...
Der Befehl **INGEST** enthält Zeichenfolgen oder SQL-Kennungen, die in die Codepage der Eingabedaten konvertiert werden müssen.	Anwendungscode-page	Codepage der Eingabedaten	Dienstprogramm INGEST
Das Dienstprogramm kann die Anwendungscodepage durch die Codepage der Eingabedaten überschreiben.	Codepage der Eingabedaten	Datenbankcodepage	DB2
Das Dienstprogramm kann die Anwendungscodepage nicht überschreiben.	Codepage der Eingabedaten	Anwendungscode-page	Dienstprogramm INGEST
Das Dienstprogramm kann die Anwendungscodepage nicht überschreiben.	Anwendungscode-page	Datenbankcodepage	DB2

INGEST-Operationen in einer DB2 pureScale-Umgebung

Bei Verwendung des Dienstprogramms INGEST in einer DB2 pureScale-Umgebung sind einige weitere Aspekte zu beachten.

Wenn jeder Flusher eine Verbindung zu einer Datenbank in einer DB2 pureScale-Instanz herstellt:

- Wurde der Befehl **SET CLIENT** mit der Option **CONNECT_MEMBER** abgesetzt, stellt der Flusher eine Verbindung zu diesem Member her.
- Andernfalls gibt der Flusher nicht das Member an, zu dem eine Verbindung hergestellt werden soll. In diesem Fall verwendet der DB2-Client Lastausgleich auf Verbindungsebene, um das Member auszuwählen, zu dem eine Verbindung hergestellt werden soll.

Mehrere Aufrufe des Dienstprogramms INGEST können mit demselben Member oder mit verschiedenen Member arbeiten, je nachdem, ob der Befehl **SET CLIENT** mit der Option **CONNECT_MEMBER** angegeben wurde und welches Member der Lastausgleich auswählt.

Wenn Sie außerdem nicht explizit angeben, zu welchem Member eine Verbindung hergestellt werden soll, müssen Sie sicherstellen, dass sich die INGEST-Datei auf einer gemeinsam genutzten Platte befindet, auf die alle Member Zugriff haben.

INGEST-Operationen in einer Umgebung mit partitionierten Datenbanken

Mit dem Dienstprogramm INGEST können Sie Daten in eine Umgebung mit partitionierten Datenbanken versetzen.

INGEST-Befehle, die in einer partitionierten Datenbank ausgeführt werden, verwenden pro Partition mindestens einen Flusher, wie durch den Konfigurationsparameter **num_flushers_per_partition** angegeben. Der Standardwert lautet wie folgt:

```
max(1, ((Anzahl logischer CPUs)/2)/(Anzahl Partitionen) )
```

Sie können diesen Parameter auch auf 0 setzen. Dies bedeutet, dass ein Flusher für alle Partitionen ausgeführt wird.

Jeder Flusher stellt eine direkte Verbindung zu der Partition her, an die er Daten senden soll. Damit die Verbindung erfolgreich ist, müssen alle DB2-Serverpartitionen für den Empfang von Clientverbindungen dieselbe Portnummer verwenden.

Wenn es sich bei der Zieltabelle um einen Typ mit einem Verteilungsschlüssel handelt, bestimmt das Dienstprogramm INGEST die Partition, zu der die einzelnen Datensätze gehören, wie folgt:

1. Es wird festgestellt, ob den einzelnen Verteilungsschlüsseln genau ein entsprechendes Feld bzw. ein konstanter Wert gegenübersteht. Dies ist wahr, falls Folgendes zutrifft:
   - Bei INSERT-Anweisungen enthält die Spaltenliste alle Verteilungsschlüssel. Für jeden Verteilungsschlüssel handelt es sich bei dem entsprechenden Element in der Liste VALUES um einen Feldnamen oder eine Konstante.
   - Bei einer Anweisung UPDATE oder DELETE weist das Vergleichselement WHERE folgendes Format auf:

     ```
     (verteilschlüss-spal1 = wert1) AND (verteilschlüss-spal2 = wert2) AND ...
     (verteilschlüss-spaln = wertn) [AND alle-weiteren-bedingungen]
     ```

Dabei gilt: *verteilschlüss-spal1* bis *verteilschlüss-spaln* sind alle Verteilungsschlüssel und jeder *wert* ist ein Feldname oder eine Konstante.

- Bei einer Anweisung MERGE weist die Suchbedingung das Format wie oben für UPDATE oder DELETE angeführt auf.

2. Wenn jeder Verteilungsschlüssel genau einen entsprechenden Feld- oder Konstantenwert aufweist, verwendet das Dienstprogramm INGEST den Verteilungsschlüssel, um die Partitionsnummer zu bestimmen, und leitet den Datensatz anschließend an einen der Flusher dieser Partition weiter.

   Anmerkung: In den folgenden Fällen bestimmt das Dienstprogramm INGEST nicht die Partition des Datensatzes. Sind mehrere Flusher vorhanden, leitet das Dienstprogramm INGEST den Datensatz an einen beliebig ausgewählten Flusher weiter:

   - Bei der Zieltabelle handelt es sich um einen Typ ohne Verteilungsschlüssel.
   - Die Spaltenliste (INSERT) oder das Vergleichselement (UPDATE, MERGE, DELETE) gibt nicht alle Verteilungsschlüssel an. Im folgenden Beispiel fehlen die Schlüsselspalten 2-8:

     ```
     UPDATE my_table SET data = $data
         WHERE (schlüssel1 = $key1) AND (schlüssel9 = $key9);
     ```

   - Ein Verteilungsschlüssel entspricht mehr als einem einzigen Feld oder Wert, wie im folgenden Beispiel dargestellt:

     ```
     UPDATE my_table SET data = $data
         WHERE schlüssel1 = $key11 OR schlüssel1 = $key12;
     ```

   - Ein Verteilungsschlüssel entspricht einem Ausdruck, wie in folgendem Beispiel dargestellt:

     ```
     INGEST FROM FILE ...
         INSERT INTO my_table(verteilschlüs, spal1, spal2)
             VALUES($field1 + $field2, $col1, $col2);
     ```

   - Eine Verteilungsschlüsselspalte weist den Typ DB2SECURITYLABEL auf.
   - Ein Feld, das einem Verteilungsschlüssel zugeordnet ist, weist einen numerischen Typ auf, der Spaltentyp für den Verteilungsschlüssel hat jedoch einen anderen numerischen Typ bzw. eine andere Genauigkeit oder Skalierung.

Beispielscripts des Dienstprogramms INGEST

Sie können das Beispielscript des Dienstprogramms INGEST verwenden, um das Schreiben eines neuen Befehls INGEST jedes Mal, wenn neue Dateien zur Verarbeitung vorliegen, zu automatisieren.

Bei dem Beispielscript `ingest_files.sh` handelt es sich um ein Shell-Script, das automatisch nach neuen Dateien sucht und einen Befehl INGEST generiert, um die Dateien zu verarbeiten. Das Script führt folgende Tasks der Reihe nach aus:

1. Prüfen des Verzeichnisses, um festzustellen, ob neue Dateien zur Verarbeitung vorhanden sind. Sind neue Dateien vorhanden, wird das Script beendet.

   Anmerkung: Das Script geht davon aus, dass das angegebene Verzeichnis nur Dateien für die Tabelle enthält, die Sie füllen wollen.
2. Abrufen der Namen der neuen Dateien und Generieren eines eigenen Befehls INGEST für jede Datei.
3. Ausführen des Befehls INGEST und Verarbeiten des Rückkehrcodes.
4. Versetzen der verarbeiteten Dateien in ein Verzeichnis für erfolgreiche Verarbeitung oder fehlgeschlagene Verarbeitung.

Das Script steht im Verzeichnis `samples/admin_scripts` unter Ihrem Installationsverzeichnis.

Ändern des Scripts für Ihre Umgebung

Sie können das Script `ingest_files.sh` als Grundlage für Ihr eigenes Script verwenden. Folgendes sind wichtige Änderungen, die Sie vornehmen müssen:

- Ersetzen Sie die Beispielwerte (den Datenbanknamen, Tabellennamen) durch Ihre eigenen Werte.
- Ersetzen Sie den Beispielbefehl INGEST durch Ihren eigenen Befehl.
- Erstellen Sie die im Script angegebenen Verzeichnisse.

Das Script verarbeitet Dateien, die Daten enthalten, um eine einzelne Tabelle zu füllen. Zum Füllen mehrerer Tabellen können Sie das Verfahren für jede zu füllende Tabelle replizieren oder das Verfahren zum Verarbeiten mehrerer Tabellen verallgemeinern.

Beispielszenario

Die Dokumentation enthält ein Beispielszenario, in dem dargestellt wird, wie Sie das Beispielscript an Ihr Data-Warehouse anpassen können, um die Generierung neuer INGEST-Befehle zu automatisieren.

Szenario: Verarbeiten eines Dateienstroms mit dem Dienstprogramm INGEST

Im folgenden Szenario wird gezeigt, wie Sie Ihr Data-Warehouse konfigurieren, um einen kontinuierlichen Datenstrom aus Datendateien einzupflegen.

Problemstellung: In einigen Data-Warehouses treffen den ganzen Tag über kontinuierlich Dateien in einem Datenstrom ein und müssen sofort bei Eingang verarbeitet werden. Dies bedeutet, dass bei jedem Eingang einer neuen Datei ein anderer Befehl **INGEST** ausgeführt werden muss, der die neu zu verarbeitende Datei angibt.

Lösung: Sie können ein Script schreiben, das eine automatische Überprüfung auf neue Dateien ausführt, einen neuen Befehl **INGEST** generiert und den Befehl ausführt. Die Datei `ingest_files.sh` ist ein Beispiel für ein solches Script. Sie müssen auch einen Crontab-Eintrag erstellen, um anzugeben, wie häufig das Shell-Script voraussichtlich ausgeführt wird.

Bevor der Benutzer diesen Mechanismus (d. h. das Script und den Crontab-Eintrag) zum Verarbeiten des Dateienstroms implementiert, müssen folgende Voraussetzungen und Abhängigkeiten erfüllt sein:

- Die Zieltabelle wurde in der Zieldatenbank erstellt.
- Das Dienstprogramm INGEST ist zur Verwendung bereit (d. h. installiert und auf einer Clientmaschine konfiguriert).
- Der Befehl INGEST wurde angegeben und durch manuelles Ausführen für eine Testdatei verifiziert.
- Die Objekte, wie zum Beispiel die Ausnahmetabelle, auf die im Befehl **INGEST** verwiesen wird, wurden erstellt.
- Auf dem System, auf dem das Dienstprogramm INGEST ausgeführt wird, wurde eine Crontab-Datei erstellt.

- Der Benutzer verfügt über einen Prozess zum Erstellen der Eingabedateien und zum Versetzen dieser Dateien in das Quellenverzeichnis, das vom Script verwendet wird.

1. Der Benutzer erstellt ein neues Script und verwendet dazu wie folgt die Datei `ingest_files.sh` als Schablone:
   a. Ersetzen Sie die folgenden Beispielwerte für die Eingabe durch Benutzerwerte:
      - VERZEICHNIS_EINGABEDATEIEN
      - DATENBANKNAME
      - SCHEMANAME
      - TABELLENNAME
      - SCRIPTPFAD
   b. Ersetzen Sie den Beispielbefehl **INGEST**
   c. Speichern Sie das Script als `populate_table1_script`
2. Der Benutzer fügt der Crontab-Datei einen Eintrag hinzu, um anzugeben, wie häufig das Script ausgeführt werden soll. Da der Benutzer möchte, dass das Script an jedem Tag des Jahres 24 Stunden lang pro Minute einmal ausgeführt wird, fügt er folgende Zeile hinzu:

   ```
   1 * * * * $HOME/bin/populate_table1_script
   ```
3. Der Benutzer testet das Script, indem er neue Eingabedateien erstellt und sie dem Quellenverzeichnis hinzufügt.

Weitere Optionen beim Versetzen von Daten

Versetzen von Tabellen im Online-Modus mithilfe der Prozedur ADMIN_MOVE_TABLE

Mithilfe der Prozedur ADMIN_MOVE_TABLE können Sie Tabellen online oder offline versetzen. Versetzen Sie Tabellen nicht im Offline-Modus, sondern im Online-Modus, wenn Verfügbarkeit bei Ihnen einen höheren Stellenwert hat als Kosten, Speicherplatz, Leistung beim Versetzen und Transaktionsaufwand.

Vorbereitende Schritte

Stellen Sie sicher, dass der Plattenspeicherplatz für die Kopien von Tabelle und Index, die Zwischenspeichertabelle und zusätzliche Protokolleinträge ausreicht.

Informationen zu diesem Vorgang

Sie können eine Tabelle online versetzen, indem Sie die gespeicherte Prozedur einmalig oder mehrmals (für jede einzelne von der Prozedur auszuführende Operation) aufrufen. Ein mehrmaliges Aufrufen eröffnet Ihnen zusätzliche Möglichkeiten. So können Sie in diesem Fall z. B. das Versetzen abbrechen oder steuern, wann die Zieltabelle für eine Aktualisierung offline genommen wird.

Wenn Sie die Prozedur SYSPROC.ADMIN_MOVE_TABLE aufrufen, wird eine Spiegelkopie der Quellentabelle erstellt. Während der Kopierphase werden Änderungen (Aktualisierungen, Einfügungen und Löschungen) an der Quellentabelle mit Triggern erfasst und in eine Zwischenspeichertabelle gestellt. Im Anschluss an die Kopierphase werden die in der Zwischenspeichertabelle erfassten Änderungen auf die Spiegelkopie angewendet. Anschließend wird die Quellentabelle von der gespeicherten Prozedur für eine kurze Zeit offline genommen, um der Spiegelkopie und den zugehörigen Indizes den Tabellennamen und die Indexnamen der Quel-

lentabelle zuzuordnen. Die Spiegeltabelle wird anschließend erneut online genommen und tritt an die Stelle der Quellentabelle. Die Quellentabelle wird standardmäßig gelöscht, Sie können jedoch bei Bedarf mithilfe der Option KEEP sicherstellen, dass sie unter einem anderen Namen beibehalten wird.

Vermeiden Sie es, Tabellen ohne Indizes, insbesondere ohne eindeutige Indizes, online zu versetzen. Wenn Sie eine Tabelle ohne einen eindeutigen Index online versetzen, kann dies zu Deadlocks führen und eine komplizierte bzw. kostenintensive Wiederholung von Operationen erforderlich machen.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Tabelle online zu versetzen:

1. Rufen Sie die Prozedur ADMIN_MOVE_TABLE auf eine der folgenden Arten auf:
   - Rufen Sie die Prozedur ADMIN_MOVE_TABLE einmalig auf und geben Sie dabei zumindest den Schemanamen der Quellentabelle, den Namen der Quellentabelle und den Operationstyp MOVE an. Verwenden Sie beispielsweise die folgende Syntax, um die Daten in eine vorhandene Tabelle in demselben Tabellenbereich zu versetzen:

     ```
     CALL SYSPROC.ADMIN_MOVE_TABLE (
     'schemaname',
     'quellentabelle',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     'MOVE')
     ```

   - Rufen Sie die Prozedur ADMIN_MOVE_TABLE mehrmals auf (einmal für jede Operation) und geben Sie dabei zumindest den Schemanamen der Quellentabelle, den Namen der Quellentabelle und einen Operationsnamen an. Verwenden Sie beispielsweise die folgende Syntax, um die Daten in eine neue Tabelle in demselben Tabellenbereich zu versetzen:

     ```
     CALL SYSPROC.ADMIN_MOVE_TABLE (
     'schemaname',
     'quellentabelle',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     '',
     'operationsname')
     ```

     Dabei steht *operationsname* für die folgenden Werte: INIT, COPY, REPLAY, VERIFY oder SWAP. Sie müssen die Prozedur in dieser Operationsreihenfolge aufrufen. Beim ersten Aufruf muss z. B. INIT als Operationsname angegeben werden.

     Anmerkung: Die Operation VERIFY ist kostenintensiv. Führen Sie diese Operation nur aus, wenn sie für das Versetzen der Tabelle benötigt wird.

2. Führen Sie die Prozedur erneut aus, wenn das Versetzen im Online-Modus fehlschlägt:
   a. Beheben Sie den Fehler, der dazu führt, dass das Versetzen der Tabelle fehlschlägt.
   b. Bestimmen Sie die Phase, in der das Versetzen fehlgeschlagen ist, indem Sie den Prozedurstatus in der Protokolltabelle SYSTOOLS.ADMIN_MOVE_TABLE abfragen.
   c. Rufen Sie die gespeicherte Prozedur erneut auf und geben Sie dabei eine der folgenden Optionen an:
      - Verwenden Sie die Option INIT, wenn die Prozedur den Status INIT aufweist.
      - Verwenden Sie die Option COPY, wenn die Prozedur den Status COPY aufweist.
      - Verwenden Sie die Option REPLAY oder SWAP, wenn die Prozedur den Status REPLAY aufweist.
      - Verwenden Sie die Option CLEANUP, wenn die Prozedur den Status CLEANUP aufweist.

   Wenn der Status beim Versetzen einer Tabelle im Online-Modus nicht COMPLETED oder CLEANUP lautet, können Sie das Versetzen abbrechen, indem Sie die Option CANCEL für die gespeicherte Prozedur angeben.

Beispiele

Beispiel 1: Versetzen Sie die Tabelle T1 im Schema SVALENTI in den Tabellenbereich ACCOUNTIN, ohne T1 dazu offline zu nehmen. Geben Sie die Tabellenbereiche DATA, INDEX und LONG an, um die Tabelle in einen neuen Tabellenbereich zu versetzen.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'ACCOUNTING',
'ACCOUNTING',
'ACCOUNTING',
'',
'',
'',
'',
'',
'MOVE')
```

Beispiel 2: Versetzen Sie die Tabelle T1 im Schema EBABANI in den Tabellenbereich ACCOUNTIN, ohne T1 dazu offline zu nehmen und behalten Sie eine Kopie der ursprünglichen Tabelle nach dem Versetzen. Verwenden Sie die Optionen COPY_USE_LOAD und LOAD_MSGPATH um den Pfad der Nachrichtendatei für Ladeoperationen festzulegen. Geben Sie die Tabellenbereiche DATA, INDEX und LONG an, um die Tabelle in einen neuen Tabellenbereich zu versetzen. Die ursprüngliche Tabelle bleibt mit einem Namen, der 'EBABANI' ähnlich ist, erhalten. T1AAAAVxo'.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'EBABANI',
'T1',
'ACCOUNTING',
'ACCOUNTING',
'ACCOUNTING',
'',
'',
```

```
'',
'',
'KEEP, COPY_USE_LOAD,LOAD_MSGPATH "/home/ebabani"',
'MOVE')
```

Beispiel 3: Versetzen Sie die Tabelle T1 in demselben Tabellenbereich. Ändern Sie die Spalte C1 der Tabelle T1, die den nicht weiter unterstützten Datentyp LONG VARCHAR verwendet, damit künftig ein kompatibler Datentyp verwendet wird.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'',
'',
'',
'',
'',
'',
'C1 VARCHAR(1000), C2 INT(5), C3 CHAR(5), C4 CLOB',
'',
'MOVE')
```

Anmerkung: Sie können den Spaltennamen während dieser Operation nicht ändern.

Beispiel 4: Sie haben die Tabelle T1 über die folgende Anweisung erstellt:

```
CREATE TABLE T1(C1 BIGINT,C2 BIGINT,C3 CHAR(20),C4 DEC(10,2),C5 TIMESTAMP,C6
BIGINT GENERATED ALWAYS AS (C1+c2),C7 GRAPHIC(10),C8 VARGRAPHIC(20),C9 XML
```

Versetzen Sie die Tabelle in demselben Tabellenbereich und löschen Sie die Spalten C5 und C6:

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'',
'',
'',
'',
'',
'',
'c1 BIGINT,c2 BIGINT ,c3 CHAR(20),c4 DEC(10,2),c7 GRAPHIC(10),c8 VARGRAPHIC(20),c9 XML',
'',
'MOVE')
```

Beispiel 5: Sie haben eine bereichspartitionierte Tabelle mit zwei in den Tabellenbereichen TS1 und TS2 definierten Bereichen. Verschieben Sie die Tabelle in den Tabellenbereich TS3, belassen Sie jedoch den ersten Bereich in TS1.

```
CREATE TABLE "EBABANI "."T1"  (
 "I1" INTEGER ,
 "I2" INTEGER )
  DISTRIBUTE BY HASH("I1")
 PARTITION BY RANGE("I1")
 (PART "PART0" STARTING(0) ENDING(100) IN "TS1",
 PART "PART1" STARTING(101) ENDING(MAXVALUE) IN "TS2");
```

Verschieben Sie die Tabelle T1 aus dem Schema EBABANI in den Tabellenbereich TS3. Geben Sie die Partitionsdefinitionen an.

```
DB2 "CALL SYSPROC.ADMIN_MOVE_TABLE
('EBABANI',
'T1',
'TS3',
'TS3',
'TS3',
```

```
'',
'',
'(I1) (STARTING 0 ENDING 100 IN TS1 INDEX IN TS1 LONG IN TS1, STARTING 101
      ENDING MAXVALUE IN TS3 INDEX IN TS3 LONG IN TS3)',
'',
'',
'MOVE')"
```

IBM Replikationstools nach Komponenten geordnet

IBM bietet zwei primäre Replikationslösungen an: Q Replication und SQL Replication.

Die Hauptkomponenten von Q Replication sind das Q Capture-Programm und das Q Apply-Programm. Die Hauptkomponenten von SQL Replication sind das Capture-Programm und das Apply-Programm. Beide Replikationstypen nutzen gemeinsam den Replikationsalertmonitor. Sie können diese Replikationskomponenten mithilfe der Replikationszentrale und des Befehlszeilenprogramms ASNCLP konfigurieren und verwalten.

In der folgenden Liste werden diese Replikationskomponenten kurz zusammengefasst:

Q Capture-Programm

Sucht im DB2-Recoveryprotokoll nach Änderungen an DB2-Quellentabellen und setzt festgeschriebene Quellendaten in WebSphere MQ-Nachrichten um, die im XML-Format für eine subskribierende Anwendung veröffentlicht oder in einem komprimierten Format in das Q Apply-Programm repliziert werden können.

Q Apply-Programm

Ruft WebSphere MQ-Nachrichten aus einer Warteschlange ab, setzt die Nachrichten in SQL-Anweisungen um und aktualisiert eine Zieltabelle oder gespeicherte Prozedur. Zu den unterstützten Zielen gehören DB2-Datenbanken oder -Subsysteme sowie Informix und Microsoft SQL Server-Datenbanken, auf die über Kurznamen von Servern mit föderierten Datenbanken zugegriffen wird.

Capture-Programm

Sucht im DB2-Recoveryprotokoll nach Änderungen an registrierten Quellentabellen oder Sichten und speichert anschließend festgeschriebene Transaktionsdaten in relationalen Tabellen (so genannten CD-Tabellen) so lange zwischen, bis das Zielsystem bereit ist, diese Daten zu kopieren. SQL Replication stellt außerdem Capture-Trigger zur Verfügung, die eine Zwischenspeichertabelle (eine so genannte CCD-Tabelle) mit Datensätzen zu Änderungen an Nicht-DB2-Quellentabellen auffüllt.

Apply-Programm

Liest Daten aus Zwischenspeichertabellen und nimmt die erforderlichen Änderungen an Zieltabellen vor. Bei Nicht-DB2-Datenquellen liest das Apply-Programm die CCD-Tabelle mithilfe des Kurznamens dieser Tabelle in der föderierten Datenbank und nimmt die erforderlichen Änderungen an der Zieltabelle vor.

Replikationsalertmonitor

Ein Dienstprogramm, das überprüft, ob das Q Capture-, Q Apply-, Capture- und Apply-Programm ordnungsgemäß funktioniert. Es sucht nach Situationen, in de-

nen ein Programm beendet wird, eine Warnung oder Fehlernachricht ausgibt, einen bestimmten Grenzwert erreicht oder eine bestimmte Aktion ausführt, und gibt anschließend Benachrichtigungen an einen E-Mail-Server, einen Pager oder die z/OS-Konsole aus.

Verwenden Sie die Replikationszentrale für folgende Zwecke:

- Definieren von Registrierungen, Subskriptionen, Veröffentlichungen, Warteschlangenmasken, Alertbedingungen und anderen Objekten
- Starten, Stoppen, Aussetzen, Wiederaufnehmen und Reinitialisieren des Replikationsprogramms
- Festlegen des zeitlichen Ablaufs der automatisierten Kopiervorgänge
- Angeben von SQL-Erweiterungen für die Daten
- Definieren von Beziehungen zwischen den Quellen- und Zieltabellen

Kopieren von Schemata

Das Dienstprogramm **db2move** und die Prozedur ADMIN_COPY_SCHEMA ermöglichen Ihnen die rasche Erstellung von Kopien eines Datenbankschemas. Nach der Einrichtung eines Modellschemas können Sie dieses als Vorlage für die Erstellung neuer Versionen verwenden.

Vorgehensweise

- Verwenden Sie die Prozedur ADMIN_COPY_SCHEMA, um ein einzelnes Schema innerhalb derselben Datenbank zu kopieren.
- Verwenden Sie das Dienstprogramm **db2move** mit der **-co COPY**, um ein einzelnes Schema oder mehrere Schemas aus einer Quellendatenbank in eine Zieldatenbank zu kopieren. Die meisten Datenbankobjekte aus dem Quellenschema werden unter das neue Schema in der Zieldatenbank kopiert.

Tipps zur Fehlerbehebung

Sowohl die Prozedur ADMIN_COPY_SCHEMA als auch das Dienstprogramm **db2move** rufen den Befehl **LOAD** auf. Während der Verarbeitung des Ladevorgangs werden die Tabellenbereiche, in denen sich die Zieldatenbankobjekte befinden, in den Status 'Backup anstehend' versetzt.

Prozedur ADMIN_COPY_SCHEMA

Durch die Verwendung dieser Prozedur mit der Option COPYNO werden die Tabellenbereiche, in denen sich das Zielobjekt befindet, in den Status 'Backup anstehend' versetzt, wie in der obigen Anmerkung beschrieben. Um den Tabellenbereich aus dem Status 'Festlegen der Integrität anstehend' herauszunehmen, setzt diese Prozedur eine Anweisung SET INTEGRITY ab. In Situationen, in denen ein Zieltabellenobjekt definierte referenzielle Integritätsbedingungen besitzt, wird die Zieltabelle ebenfalls in den Status 'Festlegen der Integrität anstehend' versetzt. Da sich die Tabellenbereiche bereits im Status 'Backup anstehend' befinden, schlägt der Versuch der Prozedur ADMIN_COPY_SCHEMA zum Absetzen der Anweisung SET INTEGRITY fehl.

Zur Behebung dieses Problems müssen Sie einen Befehl **BACKUP DATABASE** absetzen, um den Status 'Backup anstehend' für die betroffenen Tabellenbereiche aufzuheben. Als Nächstes müssen Sie die Spalte `Statement_text` der Fehlertabelle prüfen, die von dieser Prozedur generiert wurde, um eine Liste der Tabellen im Status 'Festlegen der Integrität anstehend' zu finden. Führen Sie anschließend die Anweisung SET INTEGRITY für jede in der

Liste aufgeführte Tabelle aus, um jede einzelne Tabelle aus dem Status 'Festlegen der Integrität anstehend' herauszunehmen.

Dienstprogramm 'db2move'
Dieses Dienstprogramm versucht, alle zulässigen Schemaobjekte mit Ausnahme der folgenden Typen zu kopieren:

- Tabellenhierarchie
- Zwischenspeichertabellen (vom Dienstprogramm LOAD in Umgebungen mit mehreren Datenbankpartitionen nicht unterstützt)
- JAR-Dateien (Java Routine Archives)
- Kurznamen
- Pakete
- Sichthierarchien
- Objektzugriffsrechte (Alle neuen Objekte werden mit Standardberechtigungen erstellt.)
- Statistiken (Neue Objekte enthalten keine statistischen Informationen.)
- Indexerweiterungen (in Verbindung mit benutzerdefinierten strukturierten Typen)
- Benutzerdefinierte strukturierte Typen und ihre Umsetzungsfunktionen

Fehler aufgrund nicht unterstützter Typen
Wenn ein Objekt eines der nicht unterstützten Typen im Quellenschema angetroffen wird, wird ein Eintrag in einer Fehlerdatei protokolliert. Die Fehlerdatei gibt an, dass ein nicht unterstützter Objekttyp erkannt wurde. Die Operation COPY wird trotzdem erfolgreich ausgeführt. Der protokollierte Eintrag soll Sie über Objekte informieren, die durch diese Operation nicht kopiert wurden.

Nicht mit Schemata gekoppelte Objekte
Objekte, die nicht mit einem Schema gekoppelt sind, wie zum Beispiel Tabellenbereiche und Ereignismonitore, werden bei einer Schemakopieroperation nicht berücksichtigt. Sie müssen sie in der Zieldatenbank vor dem Aufrufen der Schemakopieroperation erstellen.

Replizierte Tabellen
Wenn eine replizierte Tabelle kopiert wird, ist die neue Kopie der Tabelle nicht für die Replikation eingerichtet. Die Tabelle wird als reguläre Tabelle erneut erstellt.

Unterschiedliche Instanzen
Die Quellendatenbank muss katalogisiert werden, wenn sie sich nicht in derselben Instanz wie die Zieldatenbank befindet.

Option SCHEMA_MAP
Wenn die Option SCHEMA_MAP zur Angabe eines anderen Schemanamens in der Zieldatenbank verwendet wird, führt die Schemakopieroperation nur eine minimale Analyse der Objektdefinitionsanweisungen durch, um den ursprünglichen Schemanamen durch den neuen Schemanamen zu ersetzen. Zum Beispiel werden alle Vorkommen des ursprünglichen Schemanamens, die im Inhalt einer SQL-Prozedur auftreten, nicht durch den neuen Schemanamen ersetzt. Aus diesem Grund könnte die Erstellung dieser Objekte durch die Schemakopieroperation fehlschlagen. Andere Beispiele können möglicherweise die Zwischenspeichertabelle, die Ergebnistabelle und die MQT (Materialized Query Table) sein. Sie können die Anweisun-

gen der Datendefinitionssprache (DDL) in der Fehlerdatei verwenden, um diese nicht erstellten Objekte nach Abschluss der Kopieroperation erneut zu erstellen.

Gegenseitige Abhängigkeiten zwischen Objekten
Die Schemakopieroperation versucht, Objekte in einer Reihenfolge erneut zu erstellen, die die gegenseitigen Abhängigkeiten zwischen diesen Objekten berücksichtigt. Wenn zum Beispiel eine Tabelle T1 eine Spalte enthält, die eine benutzerdefinierte Funktion U1 referenziert, erstellt die Operation die Funktion U1 erneut, bevor sie die Tabelle T1 erneut erstellt. Informationen zu Abhängigkeiten von Prozeduren sind jedoch in den Katalogen nicht einfach verfügbar. Bei der erneuten Erstellung von Prozeduren versucht die Schemakopieroperation daher zunächst, alle Prozeduren erneut zu stellen. Anschließend wiederholt sie den Versuch für die Prozeduren, deren erste Neuerstellung fehlgeschlagen ist. (Dies geschieht in der Annahme, dass sich Prozeduren, die von einer Prozedur abhängig sind, die beim vorherigen Versuch erfolgreich erstellt wurde, bei einem nachfolgenden Versuch erfolgreich neu erstellen lassen.) Die Operation setzt die Versuche, diese Prozeduren, deren Neuerstellung fehlgeschlagen ist, erneut zu erstellen, so lange fort, wie sie in der Lage ist, mindestens eine weitere Prozedur bei einem nachfolgenden Versuch erneut zu erstellen. Bei jedem Versuch, eine Prozedur erneut zu erstellen, wird ein Fehler (mit zugehörigen DDL-Anweisungen) in der Fehlerdatei protokolliert. Möglicherweise sehen Sie in der Fehlerdatei viele Einträge für dieselben Prozeduren. Diese Prozeduren können dennoch bei einem nachfolgenden Versuch erfolgreich erneut erstellt worden sein. Sie sollten nach Abschluss der Schemakopieroperation die Tabelle SYSCAT.PROCEDURES abfragen, um festzustellen, ob diese in der Fehlerdatei aufgeführten Prozeduren erfolgreich erneut erstellt wurden.

Weitere Informationen finden Sie in den Beschreibungen zur Prozedur ADMIN_COPY_SCHEMA und zum Dienstprogramm **db2move**.

Beispiele für das Kopieren eines Schemas mit dem Dienstprogramm 'db2move'

Zum Kopieren eines einzigen Schemas oder mehrerer Schemata aus einer Quellendatenbank in eine Zieldatenbank können Sie das Dienstprogramm **db2move** mit der Aktion **COPY -co** verwenden. Nach der Einrichtung eines Modellschemas können Sie dieses als Vorlage für die Erstellung neuer Versionen verwenden.

Beispiel 1: Verwenden der Optionen für 'COPY -co'
Im folgenden Beispiel für die **db2move**-Optionen für **COPY -co** wird das Schema BAR aus der Datenbank 'sample' in die Zieldatenbank ('target') kopiert und in FOO umbenannt:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" -u benutzer-id -p kennwort
```

Die neuen (Ziel-)Schemaobjekte werden mit den gleichen Objektnamen wie die Objekte im Quellenschema, jedoch mit dem Qualifikationsmerkmal 'target' erstellt. Es ist möglich, Kopien von Tabellen mit oder ohne Daten aus der Quellentabelle zu erstellen. Die Quellendatenbank und die Zieldatenbank können sich auf verschiedenen Systemen befinden.

Beispiel 2: Angeben der Namenszuordnungen für Tabellenbereiche bei der COPY-Operation
Das folgende Beispiel zeigt, wie bestimmte Namenszuordnungen für Tabellenbereiche angegeben werden, die bei einer **COPY**-Operation mit dem Dienstprogramm **db2move** anstelle der Tabellenbereiche aus dem Quellensystem verwendet werden sollen. Sie können das Schlüsselwort SYS_ANY

angeben, um festzulegen, dass der Zieltabellenbereich mithilfe des Standardauswahlalgorithmus für Tabellenbereiche ausgewählt werden muss. In diesem Fall wählt das Dienstprogramm **db2move** alle verfügbaren Tabellenbereiche zur Verwendung als Zielobjekte aus:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" -u benutzer-id -p kennwort
```

Das Schlüsselwort SYS_ANY kann für alle Tabellenbereiche verwendet werden. Alternativ können Sie auch bestimmte Zuordnungen für einige der Tabellenbereiche und den Standardauswahlalgorithmus für die restlichen Tabellenbereiche angeben:

```
db2move sample COPY -sn BAR -co target_db  target schema_map "
  ((BAR,FOO))" tablespace_map "((TS1, TS2),(TS3, TS4), SYS_ANY)"
  -u benutzer-id -p kennwort
```

In diesem Beispiel wird angegeben, dass Tabellenbereich TS1 dem Tabellenbereich TS2 und Tabellenbereich TS3 dem Tabellenbereich TS4 zugeordnet werden, die restlichen Tabellenbereiche jedoch den Standardauswahlalgorithmus für Tabellenbereiche verwenden.

Beispiel 3: Ändern der Objekteigner nach der COPY-Operation

Nach einer erfolgreichen COPY-Operation können Sie den Eigner der neuen Objekte ändern, die im Zielschema erstellt wurden. Der Standardeigner der Zielobjekte ist der Benutzer, der die Verbindung hergestellt hat. Wenn diese Option angegeben wird, wird das Eigentumsrecht an einen neuen Eigner wie folgt übertragen:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" owner jrichards
  -u benutzer-id -p kennwort
```

Der neue Eigner der Zielobjekte ist 'jrichards'.

Das Dienstprogramm **db2move** muss auf dem Zielsystem gestartet werden, wenn sich das Quellenschema und das Zielschema auf verschiedenen Systemen befinden. Um Schemata aus einer Datenbank in eine andere kopieren zu können, muss für diese Aktion eine Liste mit durch Kommata getrennten Schemanamen, die aus einer Quellendatenbank zu kopieren sind, sowie ein Zieldatenbankname eingegeben werden.

Zum Kopieren eines Schemas geben Sie **db2move** in eine Eingabeaufforderung des Betriebssystems wie folgt ein:

```
db2move dbname COPY -co COPY-optionen
 -u benutzer-id -p kennwort
```

db2move - Tool zum Versetzen von Daten

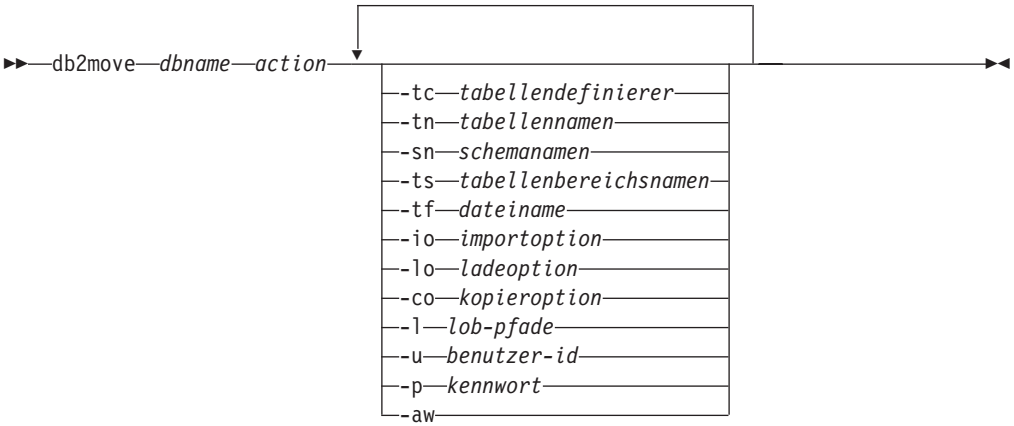
Bei Verwendung im Modus `EXPORT`, `IMPORT` oder `LOAD` erleichtert dieses Tool das Versetzen großer Mengen von Tabellen zwischen DB2-Datenbanken, die sich auf Workstations befinden. Wird dieses Tool im Modus `COPY` verwendet, erleichtert es die Duplizierung eines Schemas.

Das Tool fragt die Systemkatalogtabellen nach einer bestimmten Datenbank ab und stellt eine Liste aller Benutzertabellen zusammen. Anschließend exportiert es diese Tabellen im PC/IXF-Format. Die PC/IXF-Dateien können in eine andere lokale DB2-Datenbank auf demselben System importiert oder geladen werden, oder sie können auf eine andere Workstationplattform übertragen und in eine DB2-Datenbank auf dieser Plattform importiert oder geladen werden. Tabellen mit Spalten strukturierten Typs werden bei Verwendung dieses Tools nicht versetzt.

Berechtigung

Je nach der vom Benutzer angeforderten Aktion ruft dieses Tool die DB2-API EXPORT, IMPORT oder LOAD auf. Darum muss die anfordernde Benutzer-ID über die richtigen Berechtigungen für die APIs verfügen, sonst schlägt die Anforderung fehl.

Befehlssyntax



Befehlsparameter

dbname
Der Name der Datenbank.

action Muss eine der folgenden Aktionen sein:

EXPORT
Exportiert alle Tabellen, die die Filterkriterien in `optionen` erfüllen. Sind keine `optionen` angegeben, werden alle Tabellen exportiert. Interne Zwischenspeicherinformationen werden in der Datei `db2move.lst` gespeichert.

IMPORT
Importiert alle Tabellen, die in der internen Zwischenspeicherdatei `db2move.lst` aufgeführt sind. Bestimmte `IMPORT`-Aktionen können mit der Option **-io** angegeben werden.

LOAD
Lädt alle Tabellen, die in der internen Zwischenspeicherdatei `db2move.lst` aufgeführt sind. Bestimmte `LOAD`-Aktionen können mit der Option **-lo** angegeben werden.

COPY Dupliziert Schemata in eine Zieldatenbank. Die Zieldatenbank muss eine lokale Datenbank sein. Ein Schema bzw. mehrere Schemata wird/werden mit der Option **-sn** angegeben. Siehe die Option **-co** für spezielle `COPY`-Optionen. Mit der Option **-tn** oder **-tf** können Sie Tabellen im Modus LOAD_ONLY filtern. Wenn die gespeicherte Prozedur ADMIN_COPY_SCHEMA() verwendet wird oder das Dienstprogramm **db2move** mit der Option **-COPY** verwendet wird, müssen Sie einen Tabellenbereich mit der Bezeichnung SYSTOOLSPACE verwenden.

Die Liste im folgenden Abschnitt enthält die Dateien, die während der einzelnen Aktionen generiert werden.

-tc *tabellendefinierer*
Standardmäßig werden alle definierenden Benutzer ausgewählt.

Diese Option gilt nur für die Aktion EXPORT. Wenn sie angegeben wird, werden nur die Tabellen exportiert, die von den mit dieser Option angegebenen Benutzern definiert wurden. Wird sie nicht angegeben, werden standardmäßig alle definierenden Benutzer verwendet. Bei Angabe mehrerer definierender Benutzer müssen diese Angaben jeweils durch ein Komma getrennt werden. Leerzeichen sind zwischen den IDs der definierenden Benutzer nicht zulässig. Diese Option kann zusammen mit der Option **-tn** *tabellennamen* verwendet werden, um die Tabellen zum Exportieren auszuwählen.

Der Stern (*) kann an einer beliebigen Stelle in der Zeichenfolge als Platzhalterzeichen angegeben werden.

-tn *tabellennamen*
Standardmäßig werden alle Benutzertabellen ausgewählt.

Diese Option gilt nur für die Aktion EXPORT oder COPY.

Bei Angabe zusammen mit der Aktion EXPORT werden nur die Tabellen exportiert, deren Namen mit den in der angegebenen Zeichenfolge enthaltenen Namen übereinstimmen. Wird sie nicht angegeben, werden standardmäßig alle Benutzertabellen verwendet. Bei der Angabe mehrerer Tabellennamen müssen diese Angaben durch ein Komma getrennt werden. Leerzeichen sind zwischen Tabellennamen nicht zulässig. Tabellennamen sollten ohne Qualifikationsmerkmal aufgelistet werden, und die Option **-sn** sollte zum Filtern von Schemata verwendet werden.

Beim Export kann ein Stern (*) an einer beliebigen Stelle in der Zeichenfolge als Platzhalterzeichen angegeben werden.

Bei Angabe zusammen mit der Aktion COPY muss **-co "MODE" LOAD_ONLY** *kopieroption* ebenfalls angegeben werden, und nur die angegebenen Tabellen werden in der Zieldatenbank erneut aufgefüllt. Die Tabellennamen sollten mit ihrem Schemaqualifikationsmerkmal im Format "schema"."tabelle" aufgelistet werden.

-sn *schemanamen*
Standardmäßig werden bei EXPORT (nicht bei COPY) alle Schemata verwendet.

Falls angegeben, werden nur diejenigen Tabellen exportiert oder kopiert, deren Schemanamen übereinstimmen. Bei der Angabe mehrerer Schemanamen müssen diese Angaben durch ein Komma getrennt werden. Leerzeichen sind zwischen Schemanamen nicht zulässig. Schemanamen mit weniger als 8 Zeichen werden auf 8 Zeichen Länge aufgefüllt.

Das Platzhalterzeichen Stern (*) in den Schemanamen wird beim Exportieren in ein Prozentzeichen (%) geändert, und der Tabellenname (mit Prozentzeichen) wird im Prädikat LIKE der Klausel WHERE verwendet. Wird sie nicht angegeben, werden standardmäßig alle Schemata verwendet. Wenn **db2move** mit der Option **-tn** oder **-tc** verwendet wird, wirkt sich der Befehl nur auf diejenigen Tabellen aus, deren Schemata mit den angegebenen Schemanamen und deren definierende Benutzer mit den angegebe-

nen definierenden Benutzern übereinstimmen. Ein Schemaname fred muss als -sn fr*d* angegeben werden (nicht als -sn fr*d), wenn ein Stern verwendet wird.

-ts *tabellenbereichsnamen*
Standardmäßig werden alle Tabellenbereiche ausgewählt.

Diese Option gilt nur für die Aktion EXPORT. Wenn diese Option angegeben wird, werden nur jene Tabellen exportiert, die sich im angegebenen Tabellenbereich befinden. Wenn das Platzhalterzeichen Stern (*) in den Tabellenbereichsnamen verwendet wird, wird dieser in ein Prozentzeichen (%) geändert und der Tabellenname (mit Prozentzeichen) wird im Prädikat LIKE in der Klausel WHERE verwendet. Wird die Option **-ts** nicht angegeben, werden standardmäßig alle Tabellenbereiche verwendet. Bei der Angabe mehrerer Tabellenbereichsnamen müssen diese Angaben durch ein Komma getrennt werden. Leerzeichen sind zwischen Tabellenbereichsnamen nicht zulässig. Tabellenbereichsnamen mit weniger als 8 Zeichen werden auf 8 Zeichen Länge aufgefüllt. Beispielsweise muss der Tabellenbereichsname meintb als -ts meine*b* angegeben werden (nicht als -sn meine*b), wenn der Stern verwendet wird.

-tf *dateiname*

Bei Angabe zusammen mit der Aktion EXPORT werden nur die Tabellen exportiert, deren Namen genau mit den in der angegebenen Datei enthaltenen Namen übereinstimmen. Wird sie nicht angegeben, werden standardmäßig alle Benutzertabellen verwendet. Pro Zeile darf nur eine Tabelle aufgelistet werden, und jede Tabelle muss vollständig qualifiziert sein. Platzhalterzeichen sind in den Zeichenfolgen nicht zulässig. Es folgt ein Beispiel für den Inhalt einer Datei:

```
"SCHEMA1"."TABLE NAME1"
"SCHEMA NAME77"."TABLE155"
```

Bei Angabe zusammen mit der Aktion COPY muss **-co "MODE" LOAD_ONLY** *kopieroption* ebenfalls angegeben werden, und nur die in der Datei angegebenen Tabellen werden in der Zieldatenbank erneut aufgefüllt. Die Tabellennamen sollten mit ihrem Schemaqualifikationsmerkmal im Format "schema"."tabelle" aufgelistet werden.

-io *importoption*
Der Standardwert ist REPLACE_CREATE. Weitere Informationen zu Einschränkungen der CREATE-Funktion von IMPORT finden Sie unter „Optionen CREATE und REPLACE_CREATE des Befehls IMPORT sind veraltet”.

Gültige Werte sind INSERT, INSERT_UPDATE, REPLACE, CREATE und REPLACE_CREATE.

-lo *ladeoption*
Der Standardwert ist INSERT.

Gültige Optionen sind INSERT und REPLACE.

-co Ist COPY die Aktion des Befehls **db2move**, stehen für **-co** die nachstehenden Optionen für Folgeaktionen zur Verfügung:

"TARGET_DB *db-name* **[USER** *benutzer-id* **USING** *kennwort***]"**
Ermöglicht es Benutzern, den Namen der Zieldatenbank sowie die Benutzer-ID und das Kennwort anzugeben. (Die Benutzer-ID und das Kennwort der Quellendatenbank können mithilfe der vorhandenen Optionen **-p** und **-u** angegeben werden). Die Klausel **USER USING** ist optional. Wird für **USER** eine Benutzer-ID angegeben,

muss das Kennwort nach der Klausel **USING** angegeben werden. Wird das Kennwort nicht angegeben, wird der Benutzer von **db2move** aufgefordert, es anzugeben. Das Kennwort wird aus den im folgenden Abschnitt erläuterten Sicherheitsgründen abgefragt. **TARGET_DB** ist eine verbindliche Option für die Aktion COPY. Die mit **TARGET_DB** angegebene Zieldatenbank darf nicht mit der Quellendatenbank übereinstimmen. Zum Kopieren von Schemata innerhalb derselben Datenbank kann die Prozedur ADMIN_COPY_SCHEMA verwendet werden. Die Aktion COPY erfordert die Angabe mindestens eines Schemas (**-sn**) bzw. einer Tabelle (**-tn** oder **-tf**).

Werden zum Kopieren von Schemata von einer Datenbank in eine andere mehrere **db2move**-Befehle gleichzeitig ausgeführt, kommt es zu Deadlocks. Es darf daher immer nur ein Befehl **db2move** abgesetzt werden. Änderungen an Tabellen im Quellenschema während der Verarbeitung der Kopieraktion können dazu führen, dass die Daten im Zielschema nach dem Kopieren nicht mit denen im Quellenschema übereinstimmen.

"MODE"

DDL_AND_LOAD
Erstellt alle unterstützten Objekte aus dem Quellenschema und füllt die Tabellen mit den Daten der Quellentabellen auf. Dies ist die Standardoption.

DDL_ONLY
Erstellt alle unterstützten Objekte aus dem Quellenschema, füllt die Tabellen jedoch nicht erneut auf.

LOAD_ONLY
Lädt alle angegebenen Tabellen aus der Quellendatenbank in die Zieldatenbank. Die Tabellen müssen im Ziel bereits vorhanden sein. Der Modus LOAD_ONLY erfordert die Eingabe mindestens einer Tabelle mit der Option **-tn** oder **-tf**.

Dies ist eine wahlfreie Option, die nur für die Aktion COPY verwendet wird.

"SCHEMA_MAP"
Ermöglicht es Benutzern, ein Schema beim Kopieren in ein Ziel umzubenennen. Stellt eine Liste der Zuordnungen zwischen Quellen- und Zielschemata bereit, wobei in jedem Zuordnungspaar das Quellen- und Zielschema durch ein Kommata werden und in runden Klammern stehen. Beispiel: `schema_map ((s1, t1), (s2, t2))`. In diesem Fall werden Objekte aus Schema s1 in das Schema t1 im Ziel kopiert, und Objekte aus Schema s2 werden in das Schema t2 im Ziel kopiert. Standardmäßig entspricht der Name des Zielschemas dem Namen des Quellenschemas. Dies wird auch empfohlen. Der Grund hierfür ist, dass der Befehl **db2move** nicht versucht, das Schema für irgendwelche qualifizierten Objekte innerhalb von Objekthauptteilen zu ändern. Daher kann die Verwendung eines unterschiedlichen Zielschemanamens zu Problemen führen, wenn sich im Objekthauptteil qualifizierte Objekte befinden.

Beispiel:`create view FOO.v1 as 'select c1 from FOO.t1'`

In diesem Fall wird beim Kopieren des Schemas FOO nach BAR v1 wie folgt generiert:`create view BAR.v1 as 'select c1 from FOO.t1'`

Dies schlägt entweder fehl, weil das Schema FOO in der Zieldatenbank nicht vorhanden ist, oder es kommt zu einem unerwarteten Ergebnis, da FOO sich von BAR unterscheidet. Wird derselbe Schemaname wie für die Quelle verwendet, können solche Probleme vermieden werden. Gibt es zwischen den Schemata übergreifende Abhängigkeiten, müssen alle voneinander abhängigen Schemata kopiert werden, da ansonsten Fehler beim Kopieren der Objekte mit den übergreifenden Abhängigkeiten auftreten können.

Beispiel:`create view FOO.v1 as 'select c1 from BAR.t1'`

In diesem Fall schlägt das Kopieren von v1 entweder fehl, wenn BAR nicht ebenfalls kopiert wird, oder es kommt zu unerwarteten Ergebnissen, wenn BAR im Ziel sich von BAR in der Quelle unterscheidet. Der Befehl **db2move** versucht nicht, Abhängigkeiten zwischen Schemata zu ermitteln.

Dies ist eine wahlfreie Option, die nur für die Aktion `COPY` verwendet wird.

Wenn bereits ein Zielschema vorhanden ist, schlägt das Dienstprogramm fehl. Löschen Sie mithilfe der Prozedur ADMIN_DROP-_SCHEMA das Schema und alle ihm zugeordneten Objekte.

"NONRECOVERABLE"
Diese Option ermöglicht dem Benutzer, das Standardverhalten der Ladeoperation außer Kraft zu setzen, die zusammen mit COPY-NO ausgeführt wird. Beim Standardverhalten wird der Benutzer gezwungen, Backups aller Tabellenbereiche zu erstellen, in die geladen wurde. Bei Angabe dieses Schlüsselworts **`NONRECOVERABLE`** wird der Benutzer nicht gezwungen, sofort Backups der Tabellenbereiche zu erstellen. Es wird jedoch dringend empfohlen, die Backups so bald wie möglich durchzuführen, um sicherzustellen, dass die neu erstellten Tabellen bei Bedarf ordnungsgemäß wiederhergestellt werden können. Dies ist eine wahlfreie Option, die für die Aktion `COPY` verfügbar ist.

"OWNER"
Diese Option ermöglicht es Benutzern, den Eigner jedes neuen Objekts zu ändern, das im Zielschema nach einer erfolgreichen COPY-Aktion erstellt wird. Der Standardeigner der Zielobjekte ist der Benutzer, der die Verbindung hergestellt hat. Wenn diese Option angegeben ist, wird das Eigentumsrecht auf den neuen Eigner übertragen. Dies ist eine wahlfreie Option, die für die Aktion `COPY` verfügbar ist.

"TABLESPACE_MAP"
Der Benutzer kann Namenszuordnungen für Tabellenbereiche angeben, die bei einer Operation COPY anstelle der Tabellenbereiche aus dem Quellensystem verwendet werden sollen. Dies ist eine Matrix mit Tabellenbereichszuordnungen, die in Klammern eingeschlossen ist. Beispiel: `tablespace_map ((TS1, TS2),(TS3, TS4))`. In diesem Fall werden Objekte aus dem Tabellenbereich TS1 in den Tabellenbereich TS2 der Zieldatenbank kopiert und Objekte aus dem Tabellenbereich TS3 werden in den Tabellenbereich TS4 im Ziel kopiert. Bei `((T1, T2),(T2, T3))` werden alle Objekte in T1 in der Quellendatenbank in T2 in der Zieldatenbank erneut erstellt, und alle Objekte in T2 in der Quellendatenbank werden in T3 in der Zieldatenbank erneut erstellt. Standardmäßig wird derselbe

Tabellenbereichsname wie in der Quelle verwendet. In diesem Fall ist die Eingabezuordnung für diesen Tabellenbereich nicht erforderlich. Wenn der angegebene Tabellenbereich nicht vorhanden ist, schlägt das Kopieren der Objekte, die diesen Tabellenbereich verwenden, fehl und wird in der Fehlerdatei protokolliert.

Der Benutzer kann außerdem das Schlüsselwort SYS_ANY angeben, um festzulegen, dass der Zieltabellenbereich mithilfe des Standardauswahlalgorithmus für Tabellenbereiche ausgewählt werden soll. In diesem Fall kann **db2move** alle verfügbaren Tabellenbereiche zur Verwendung als Zielobjekte auswählen. Das Schlüsselwort SYS_ANY kann für alle Tabellenbereiche verwendet werden (Beispiel: tablespace_map SYS_ANY). Außerdem kann der Benutzer für einige Tabellenbereiche bestimmte Zuordnungen angeben, und für die restlichen Tabellenbereiche den Standardauswahlalgorithmus. Beispiel: tablespace_map ((TS1, TS2),(TS3, TS4), SYS_ANY). In diesem Beispiel wird angegeben, dass der Tabellenbereich TS1 dem Tabellenbereich TS2 zugeordnet wird und der Tabellenbereich TS3 dem Tabellenbereich TS4. Die restlichen Tabellenbereiche verwenden jedoch ein Standardtabellenbereichsziel. Das Schlüsselwort SYS_ANY wird verwendet, da ein Tabellenbereich nicht mit "SYS" beginnen darf.

Dies ist eine wahlfreie Option, die für die Aktion COPY verfügbar ist.

"PARALLEL" *threadanzahl*

Geben Sie diese Option an, damit Ladeoperationen für die Tabellen in den Schemata für eine bestimmte Anzahl von Threads ausgeführt werden. Der Wertebereich für *threadanzahl* liegt zwischen 0 und 16.

- Wurde PARALLEL nicht angegeben, dann werden keine Threads verwendet und die Ladeoperationen werden nacheinander ausgeführt.
- Wenn PARALLEL ohne die Anzahl der Threads angegeben wird, dann wählt das Dienstprogramm **db2move** einen geeigneten Wert aus.
- Wenn PARALLEL mit einem Wert für *threadanzahl* angegeben wird, dann wird die angegebene Anzahl von Threads verwendet. Wenn für *threadanzahl* der Wert 0 oder 1 angegeben wird, dann wird die Ladeoperation seriell ausgeführt.
- Der Maximalwert, der für *threadanzahl* angegeben werden kann, lautet 16.

Dies ist eine wahlfreie Option, die für die Aktion COPY verfügbar ist.

-l *lob-pfade*

Bei Angabe dieser Option für IMPORT und EXPORT wird sie auch für XML-Pfade verwendet. Der Standardwert ist das aktuelle Verzeichnis.

Diese Option gibt die absoluten Namen der Pfade an, in denen LOB- oder XML-Dateien erstellt werden (bei EXPORT) bzw. gesucht werden (bei IMPORT oder LOAD). Bei Angabe mehrerer Pfade muss jeder Pfad durch ein Komma getrennt werden. Leerzeichen sind zwischen Pfaden nicht zulässig. Werden mehrere Pfade angegeben, verwendet EXPORT sie im Umlaufverfahren. So wird ein LOB-Dokument in den ersten Pfad geschrieben, eines in den zweiten Pfad usw. bis zum letzten Pfad und anschließend zurück zum ers-

ten. Das Gleiche gilt für XML-Dokumente. Werden im ersten Pfad keine Dateien gefunden (bei Ausführung von IMPORT oder LOAD), wird der zweite Pfad verwendet usw.

-u *benutzer-id*
Der Standardwert ist die angemeldete Benutzer-ID.

Sowohl die Benutzer-ID als auch das Kennwort sind wahlfrei. Wenn jedoch eines von beiden angegeben wird, müssen beide angegeben werden. Wenn der Befehl auf einem Client ausgeführt wird, der eine Verbindung zu einem fernen Server herstellt, sollten die Benutzer-ID und das Kennwort angegeben werden.

-p *kennwort*
Standardwert ist das bei der Anmeldung verwendete Kennwort. Sowohl die Benutzer-ID als auch das Kennwort sind wahlfrei. Wenn jedoch eines von beiden angegeben wird, müssen beide angegeben werden. Wird die Option **-p** ohne Kennwort angegeben, fordert **db2move** den Benutzer zur Eingabe des Kennworts auf. Dies erfolgt aus Sicherheitsgründen. Die Eingabe des Kennworts über die Befehlszeile kann zu Sicherheitsproblemen führen. Mit dem Befehl **ps -ef** beispielsweise würde das Kennwort angezeigt werden. Wird **db2move** jedoch über ein Script aufgerufen, müssen die Kennwörter angegeben werden. Wenn der Befehl auf einem Client abgesetzt wird, der eine Verbindung zu einem fernen Server herstellt, sollten die Benutzer-ID und das Kennwort angegeben werden.

-aw Warnungen zulassen. Wenn **-aw** nicht angegeben ist, werden Tabellen, für die beim Exportieren Warnungen ausgegeben werden, nicht in die Datei db2move.lst aufgenommen, obwohl die Dateien .ixf und .msg für die betreffende Tabelle generiert werden. In manchen Szenarios (z. B. Abschneiden der Daten) möchte der Benutzer möglicherweise zulassen, dass solche Tabellen in die Datei db2move.lst aufgenommen werden. Bei Angabe dieser Option können Tabellen, für die beim Exportieren Warnungen ausgegeben werden, in die Datei .lst aufgenommen werden.

Beispiele

- Der folgende Befehl kann verwendet werden, um alle Tabellen in der Beispieldatenbank SAMPLE unter Verwendung der Standardwerte für alle Optionen zu exportieren:

```
db2move sample export
```

- Der folgende Befehl kann verwendet werden, um alle Tabellen zu exportieren, die von userid1 oder Benutzer-IDs wie us%rid2 und mit den Namen tbname1 oder Tabellenname wie %tbname2 erstellt wurden:

```
db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2
```

- Setzen Sie den folgenden Befehl ab, um alle Tabellen in der Beispieldatenbank SAMPLE zu importieren (die LOB-Pfade D:\LOBPATH1 und C:\LOBPATH2 werden nach LOB-Dateien durchsucht; dieses Beispiel gilt nur für Windows-Betriebssysteme):

```
db2move sample import -l D:\LOBPATH1,C:\LOBPATH2
```

- Setzen Sie den folgenden Befehl ab, um alle Tabellen aus der Beispieldatenbank SAMPLE zu laden (die Unterverzeichnisse /home/userid/lobpath und tmp werden nach LOB-Dateien durchsucht; dieses Beispiel gilt nur für Linux- und UNIX-Systeme):

```
db2move sample load -l /home/userid/lobpath,/tmp
```

- Der folgende Befehl kann verwendet werden, um alle Tabellen in die Beispieldatenbank SAMPLE im Ersetzungsmodus (REPLACE) zu importieren und dabei die angegebene Kombination aus Benutzer-ID (userid) und Kennwort (password) zu verwenden:

  ```
  db2move sample import -io replace -u userid -p password
  ```

- Der folgende Befehl kann verwendet werden, um das Schema schema1 aus der Quellendatenbank dbsrc in der Zieldatenbank dbtgt zu duplizieren:

  ```
  db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
  ```

- Setzen Sie den folgenden Befehl ab, um das Schema schema1 aus der Quellendatenbank dbsrc in die Zieldatenbank dbtgt zu duplizieren, das Schema in der Zieldatenbank in newschema1 umzubenennen und den Quellentabellenbereich ts1 dem Zieltabellenbereich ts2 zuzuordnen:

  ```
  db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
  ```

Hinweise zur Verwendung

- Beim Kopieren eines oder mehrerer Schemata in eine Zieldatenbank müssen die Schemata unabhängig voneinander sein. Ist dies nicht der Fall, werden möglicherweise einige Objekte nicht erfolgreich in die Zieldatenbank kopiert.
- Das Laden von Daten in Tabellen mit XML-Spalten wird nur für die Aktion **LOAD**, nicht aber für die Aktion **COPY**, unterstützt. Als Ausweichlösung kann der Befehl **IMPORT** oder **EXPORT** manuell abgesetzt werden, oder es können die Verhalten **db2move Export** und **db2move Import** verwendet werden. Wenn diese Tabellen zudem Identitätsspalten des Typs GENERATED ALWAYS enthalten, können keine Daten in die Tabellen importiert werden.
- Eine Aktion **db2move EXPORT**, gefolgt von einer Aktion **db2move IMPORT** oder **db2move LOAD**, erleichtert das Versetzen von Tabellendaten. Alle anderen den Tabellen zugeordneten Datenbankobjekte (z. B. Aliasnamen, Sichten oder Auslöser) sowie Objekte, von denen diese Tabellen möglicherweise abhängen (z. B. benutzerdefinierte Typen oder Funktionen), müssen manuell versetzt werden.
- Wenn die Aktion **IMPORT** mit der Option **CREATE** oder **REPLACE_CREATE** (beide Optionen sind veraltet und werden in einem zukünftigen Release möglicherweise entfernt) verwendet wird, um die Tabellen in der Zieldatenbank zu erstellen, gelten die in „Neuerstellung von importierten Tabellen" beschriebenen Einschränkungen. Werden während der Importphase von **db2move** bei Verwendung der Option **REPLACE_CREATE** unerwartete Fehler festgestellt, sollten Sie die entsprechende Nachrichtendatei tabnnn.msg prüfen und überlegen, ob die Fehler auf die Einschränkungen der Tabellenerstellung zurückzuführen sind.
- Tabellen mit als GENERATED ALWAYS definierten Identitätsspalten können nicht mit **db2move** importiert oder geladen werden. Diese Tabellen können jedoch manuell importiert oder geladen werden. Weitere Informationen hierzu finden Sie unter „Aspekte des Ladens von Identitätsspalten" oder „Aspekte des Identitätsspaltenimports".
- Wenn die APIs EXPORT, IMPORT und LOAD von **db2move** aufgerufen werden, wird der Parameter **FileTypeMod** auf lobsinfile gesetzt. Dies bedeutet, dass LOB-Daten für jede Tabelle in separaten Dateien (nicht in der PC/IXF-Datei) erfasst werden.
- Der Befehl **LOAD** muss lokal auf der Maschine ausgeführt werden, auf der sich die Datenbank und die Datendatei befindet.
- Wenn **db2move LOAD** verwendet wird und **logarchmeth1** für die Datenbank aktiviert ist (d. h., wenn die Datenbank wiederherstellbar ist), gilt Folgendes:
  - Ist die Option **NONRECOVERABLE** nicht angegeben, ruft **db2move** die API db2Load mit der Standardoption **COPY NO** auf, und die Tabellenbereiche, in denen sich die geladenen Tabellen befinden, werden bei Beendigung des Dienstpro-

gramms in den Status "Backup anstehend" versetzt. (Ein vollständiges Backup der Datenbank oder des Tabellenbereichs ist erforderlich, um den Status "Backup anstehend" für die Tabellenbereiche aufzuheben.)

– Ist die Option **NONRECOVERABLE** nicht angegeben, werden die Tabellenbereiche nicht in den Status "Backup anstehend" versetzt. Wird jedoch später eine aktualisierende Recovery durchgeführt, so wird die Tabelle als nicht zugänglich markiert und muss gelöscht werden. Weitere Informationen zu den Wiederherstellbarkeitsoptionen für LOAD finden Sie unter „Optionen zur Verbesserung der Ladeleistung".

- Die Leistung des Befehls **db2move** mit den Aktionen **IMPORT** oder **LOAD** kann durch Ändern des Standardpufferpools IBMDEFAULTBP und durch Aktualisieren der Konfigurationsparameter **sortheap**, **util_heap_sz**, **logfilsiz** und **logprimary** verbessert werden.
- Bei der Ausführung von Dienstprogrammen für das Versetzen von Daten wie **export** und **db2move** stellt der Abfragecompiler möglicherweise fest, dass die zugrunde liegende Abfrage für eine MQT (Materialized Query Table, gespeicherte Abfragetabelle) effizienter ausgeführt wird als für die Basistabelle(n). In diesem Fall wird die Abfrage für eine als REFRESH DEFERRED deklarierte MQT ausgeführt und das Ergebnis der Dienstprogramme zeigt die Daten in der zugrunde liegenden Tabelle möglicherweise nicht exakt an.
- Der Befehl **db2move** ist bei DB2-Clients nicht verfügbar. Wenn Sie den Befehl **db2move** auf einer Clientmaschine absetzen, erhalten Sie eine Fehlernachricht, die angibt, dass db2move nicht als interner oder externer Befehl, ausführbares Programm oder Stapeldatei erkannt wird. Sie können diesen Fehler vermeiden, indem Sie den Befehl **db2move** direkt auf dem Server absetzen.

Bei Verwendung von EXPORT erforderliche/generierte Dateien:

- Eingabe: Keine.
- Ausgabe:

EXPORT.out
Das zusammengefasste Ergebnis der Aktion EXPORT.

db2move.lst
Die Liste der Originaltabellennamen, ihrer entsprechenden PC/IXF-Dateinamen (tabnnn.ixf) sowie der Nachrichtendateinamen (tabnnn.msg). Diese Liste, die exportierten PC/IXF-Dateien und die LOB-Dateien (tabnnnc.yyy) werden als Eingabe für die Aktion IMPORT oder LOAD von **db2move** verwendet.

tab*nnn***.ixf**
Die exportierte PC/IXF-Datei einer bestimmten Tabelle.

tab*nnn***.msg**
Die exportierte Nachrichtendatei der entsprechenden Tabelle.

tab*nnnc***.***yyy*
Die exportierten LOB-Dateien einer bestimmten Tabelle.

nnn ist die Tabellennummer. *c* ist ein Buchstabe des Alphabets. *yyy* ist eine Zahl im Bereich 001 bis 999.

Diese Dateien werden nur erstellt, sofern die Tabelle, die exportiert wird, LOB-Daten enthält. Wenn sie erstellt werden, werden diese LOB-Dateien in den LOB-Pfadverzeichnissen (Wert für *lob-pfade*) gespeichert. Für die LOB-Dateien sind insgesamt 26.000 Namen möglich.

system.msg
Die Nachrichtendatei mit den Systemnachrichten zu den Befehlen für das Erstellen oder Löschen von Dateien oder Verzeichnissen. Diese Datei wird nur verwendet, wenn die Aktion EXPORT ist und ein LOB-Pfad angegeben wird.

Bei Verwendung von IMPORT erforderliche/generierte Dateien

- Eingabe:

  db2move.lst
  Eine Ausgabedatei der Aktion EXPORT.

  tab*nnn***.ixf**
  Eine Ausgabedatei der Aktion EXPORT.

  tab*nnnc*.*yyy*
  Eine Ausgabedatei der Aktion EXPORT.

- Ausgabe:

  IMPORT.out
  Das zusammengefasste Ergebnis der Aktion IMPORT.

  tab*nnn***.msg**
  Die Datei mit IMPORT-Nachrichten für die entsprechende Tabelle.

Bei Verwendung von LOAD erforderliche/generierte Dateien

- Eingabe:

  db2move.lst
  Eine Ausgabedatei der Aktion EXPORT.

  tab*nnn***.ixf**
  Eine Ausgabedatei der Aktion EXPORT.

  tab*nnnc*.*yyy*
  Eine Ausgabedatei der Aktion EXPORT.

- Ausgabe:

  LOAD.out
  Das zusammengefasste Ergebnis der Aktion LOAD.

  tab*nnn***.msg**
  Die Datei mit **LOAD**-Nachrichten für die entsprechende Tabelle.

Bei Verwendung von COPY erforderliche/generierte Dateien

- Eingabe: Keine
- Ausgabe:

  COPYSCHEMA.msg
  Eine Ausgabedatei mit Nachrichten, die während der Operation COPY generiert wurden.

  COPYSCHEMA.err
  Eine Ausgabedatei mit einer Fehlernachricht für jeden Fehler, der während der Operation COPY festgestellt wurde (einschließlich DDL-Anweisungen für jedes Objekt, das in der Zieldatenbank nicht erneut erstellt werden konnte).

  LOADTABLE.msg
  Eine Ausgabedatei mit Nachrichten, die bei den einzelnen Aufrufen des

Dienstprogramms LOAD generiert wurden, das zum erneuten Auffüllen der Daten in der Zieldatenbank verwendet wurde.

LOADTABLE.err
Eine Ausgabedatei mit den Namen der Tabellen, für die bei der Ladeoperation ein Fehler aufgetreten ist, oder die in der Zieldatenbank noch gefüllt werden müssen. Weitere Details finden Sie im Thema „Erneutes Starten einer fehlgeschlagenen Operation zum Kopieren eines Schemas".

Diese Dateien werden mit einer Zeitmarke versehen, wobei alle Dateien, die im Verlauf derselben Ausführung von COPY generiert werden, auch dieselbe Zeitmarke erhalten.

Ausführen eines umgeleiteten Restores mithilfe eines automatisch generierten Scripts

Wenn Sie eine umgeleitete Restoreoperation durchführen, müssen Sie die Positionen physischer Container angeben, die im Backup-Image gespeichert sind, und vollständige Informationen zur Gruppe von Containern für jeden Tabellenbereich bereitstellen, den Sie ändern wollen.

Vorbereitende Schritte

Sie können einen umgeleiteten Restore nur dann ausführen, wenn die Datenbank zuvor mit dem DB2-Backup-Dienstprogramm gesichert wurde.

Informationen zu diesem Vorgang

- Wenn die Datenbank vorhanden ist, müssen Sie eine Verbindung zu ihr herstellen können, um das Script zu generieren. Daher gilt: Wenn für die Datenbank ein Upgrade oder eine Recovery nach Systemabsturz erforderlich ist, muss dieses Upgrade bzw. diese Recovery ausgeführt werden, bevor Sie versuchen, ein Script für den umgeleiteten Restore zu generieren.
- Wenn Sie in einer Umgebung mit partitionierten Datenbanken arbeiten und die Zieldatenbank nicht vorhanden ist, können Sie den Befehl zur Generierung des Scripts für den umgeleiteten Restore nicht gleichzeitig in allen Datenbankpartitionen ausführen. Stattdessen muss der Befehl zur Generierung des Scripts für den umgeleiteten Restore jeweils nur in einer Datenbankpartition gleichzeitig, beginnend mit der Katalogpartition, ausgeführt werden.

  Alternativ können Sie zuerst eine Pseudodatenbank mit demselben Namen wie Ihre Zieldatenbank erstellen. Nach der Erstellung der Pseudodatenbank können Sie das Script für den umgeleiteten Restore in allen Datenbankpartitionen gleichzeitig generieren.
- Auch wenn Sie den Parameter **REPLACE EXISTING** bei der Ausführung des Befehls **RESTORE DATABASE** zur Generierung des Scripts angeben, wird der Parameter **REPLACE EXISTING** in dem Script auf Kommentar gesetzt.
- Aus Sicherheitsgründen wird Ihr Kennwort nicht in das generierte Script eingetragen. Sie müssen das Kennwort manuell eingeben.
- Das Script für den Restore umfasst die Speichergruppenzuordnungen für jeden Tabellenbereich, den Sie wiederherstellen.

Vorgehensweise

Gehen Sie zur Durchführung eines umgeleiteten Restores mithilfe eines Scripts wie folgt vor:

1. Generieren Sie mithilfe des Restoredienstprogramms ein Script für den umgeleiteten Restore. Das Restoredienstprogramm kann über den Befehlszeilenprozes-

sor (CLP) oder die Anwendungsprogrammierschnittstelle db2Restore aufgerufen werden. Das folgende Beispiel zeigt einen Befehl **RESTORE DATABASE** mit dem Parameter **REDIRECT** und dem Parameter **GENERATE SCRIPT**:

```
db2 restore db test from /home/jseifert/backups taken at 20050304090733
              redirect generate script test_node0000.clp
```

Dieser Befehl erstellt ein Script für den umgeleiteten Restore auf dem Client mit dem Namen `test_node0000.clp`.

2. Öffnen Sie das Script für den umgeleiteten Restore in einem Texteditor, um alle erforderlichen Modifikationen vorzunehmen. Folgende Angaben können modifiziert werden:
   - RESTORE-Optionen
   - Pfade für dynamischen Speicher
   - Layout und Pfade von Containern
3. Führen Sie das modifizierte Script für den umgeleiteten Restore aus. Beispiel:

   ```
   db2 -tvf test_node0000.clp
   ```

RESTORE DATABASE

Der Befehl **RESTORE DATABASE** erstellt eine beschädigte Datenbank, die mit dem DB2-Backup-Dienstprogramm gesichert wurde, erneut. Die wiederhergestellte Datenbank weist den gleichen Zustand auf, den sie beim Erstellen der Backup-Kopie hatte.

Dieses Dienstprogramm kann außerdem die folgenden Services ausführen:

- Überschreiben einer Datenbank mit einem anderen Image oder Wiederherstellen der Backup-Kopie in einer neuen Datenbank.
- Das Restoredienstprogramm in DB2 Version 9.8 kann nicht verwendet werden, um Backup-Images wiederherzustellen, die mit einer anderen Version der DB2-Software erstellt wurden.
- Wiederherstellen von Backup-Images in DB2 Version 9.7, die unter DB2 Version 9.5 erstellt wurden.
  - Wenn ein Datenbankupgrade erforderlich sein sollte, wird die Upgradeoperation nach dem Restore automatisch aufgerufen.
- Wenn die Datenbank zum Zeitpunkt der Backup-Operation für aktualisierende Recovery aktiviert war, kann die Datenbank wieder in den vorherigen Zustand versetzt werden, indem nach erfolgreicher Beendigung einer Restoreoperation das Dienstprogramm für aktualisierende Recovery aufgerufen wird.
- Wiederherstellen eines Backups auf Tabellenbereichsebene.
- Transportieren einer Gruppe von Tabellenbereichen, Speichergruppen und SQL-Schemata vom Datenbankbackup-Image in eine Datenbank mithilfe der Option **TRANSPORT** (in DB2 Version 9.7 Fixpack 2 oder späteren Fixpacks). Die Option TRANSPORT wird in der DB2 pureScale-Umgebung nicht unterstützt.
- Wenn der Datenbankname vorhanden ist, wenn dieser Befehl abgesetzt wird, dann ersetzt er alle Speichergruppen und definiert sie erneut. Dabei wird der Zustand zugrunde gelegt, den diese Elemente zum Zeitpunkt der Erstellung des Backup-Images aufwiesen, es sei denn, der Benutzer hat andere Anweisungen erteilt.

Informationen zu den von DB2-Datenbanksystemen unterstützten Restoreoperationen zwischen unterschiedlichen Betriebssystemen und Hardwareplattformen enthält der Abschnitt „Backup- und Restoreoperationen zwischen unterschiedlichen Betriebssystemen und Hardwareplattformen" in der Veröffentlichung *Datenrecovery und hohe Verfügbarkeit Handbuch und Referenz*.

Inkrementelle Images und Images, die nur die Unterschiede seit dem Zeitpunkt der letzten Aufzeichnung erfassen („Delta-Image"), können nicht wiederhergestellt werden, wenn sich die Betriebssysteme oder die Wortgröße (32-Bit oder 64-Bit) unterscheiden.

Nach einer erfolgreichen Restoreoperation von einer Umgebung in eine andere sind inkrementellen Backups oder Delta-Backups erst wieder zulässig, nachdem ein nicht inkrementelles Backup erstellt wurde. (Diese Einschränkung gilt nicht nach Restoreoperationen innerhalb derselben Umgebung.)

Auch bei erfolgreichen Restoreoperationen von einer Umgebung in eine andere sind einige Dinge zu beachten: Pakete müssen vor der Verwendung erneut gebunden werden (mit dem Befehl **BIND**, dem Befehl **REBIND** oder dem Dienstprogramm **db2rbind**), SQL-Prozeduren müssen gelöscht und erneut erstellt werden und alle externen Bibliotheken müssen auf der neuen Plattform erneut erstellt werden. (Diese Überlegungen gelten nicht für die Wiederherstellung in derselben Umgebung.)

Eine Restoreoperation, die für eine vorhandene Datenbank und vorhandene Container ausgeführt wird, verwendet dieselben Container und dieselbe Tabellenbereichszuordnung wieder.

Eine Restoreoperation, die für eine neue Datenbank ausgeführt wird, ordnet alle Container neu zu und stellt eine optimierte Tabellenbereichszuordnung wieder her. Eine Restoreoperation, die für eine vorhandene Datenbank mit mindestens einem fehlenden Container ausgeführt wird, ordnet ebenfalls alle Container neu zu und stellt eine optimierte Tabellenbereichszuordnung wieder her.

Geltungsbereich

Dieser Befehl wirkt sich nur auf den Knoten aus, auf dem er ausgeführt wird.

Für SYSCATSPACE kann keine Restoreoperation im Onlinemodus durchgeführt werden.

Berechtigung

Zum Durchführen eines Restores in eine vorhandene Datenbank ist eine der folgenden Berechtigungen erforderlich:

- SYSADM
- SYSCTRL
- SYSMAINT

Zum Durchführen eines Restores in eine neue Datenbank ist eine der folgenden Berechtigungen erforderlich:

- SYSADM
- SYSCTRL

Wenn ein Benutzername angegeben wird, benötigt dieser Benutzer die Berechtigung CONNECT für die Datenbank.

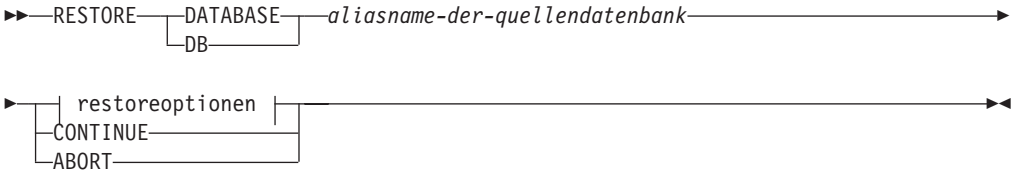
Erforderliche Verbindung

Die erforderliche Verbindung variiert je nach Typ der Restoreaktion:

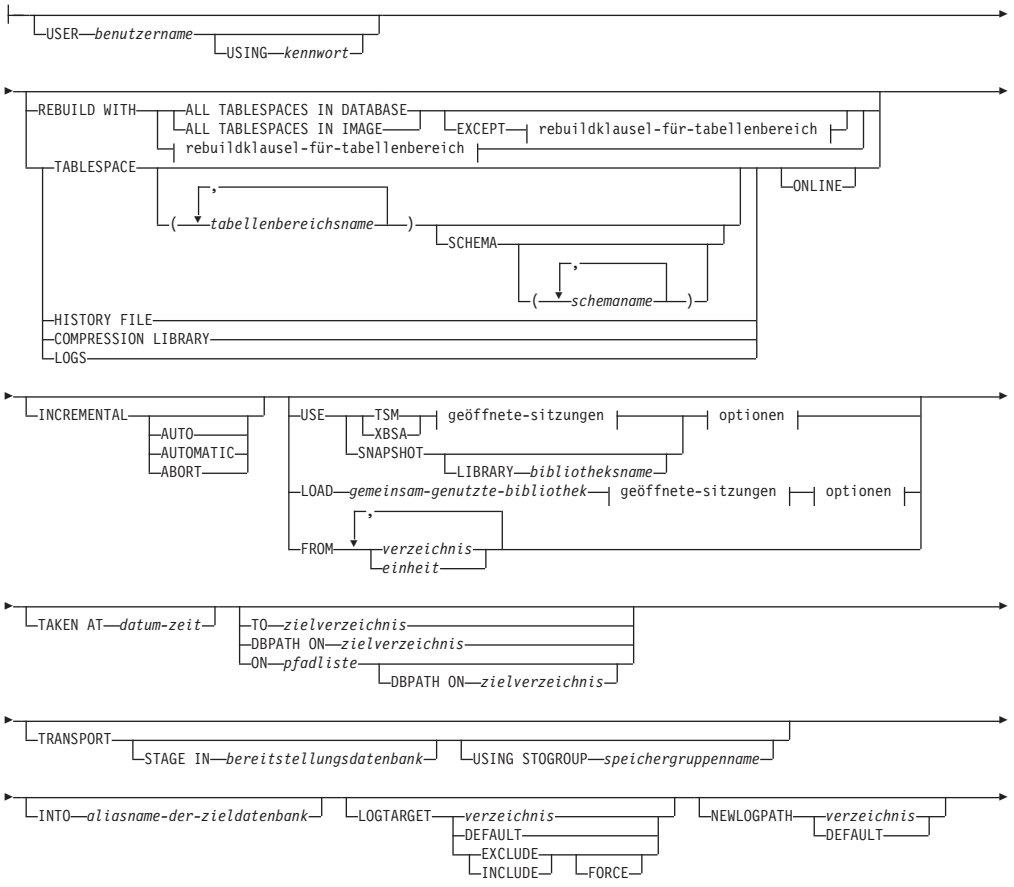
- Für den Restore in eine vorhandene Datenbank ist eine Datenbankverbindung erforderlich. Dieser Befehl stellt automatisch eine exklusive Verbindung zu der angegebenen Datenbank her.
- Für den Restore in eine neue Datenbank ist eine Instanz und eine Datenbankverbindung erforderlich. Der Instanzanschluss wird zum Erstellen der Datenbank benötigt.

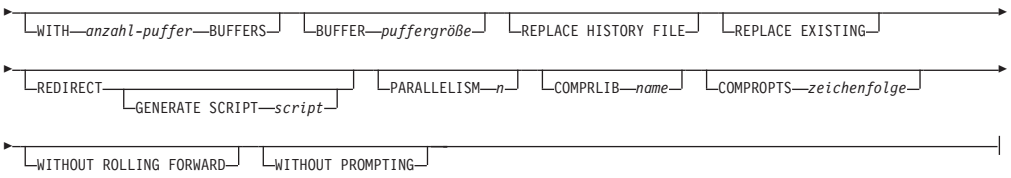
  Beim Restore in eine neue Datenbank in einer anderen als der aktuellen Instanz muss zunächst eine Verbindung zu der Instanz hergestellt werden, in der sich die neue Datenbank befinden soll. Die neue Instanz kann eine lokale oder ferne Instanz sein. Die aktuelle Instanz wird durch den Wert der Umgebungsvariablen **DB2INSTANCE** definiert.
- Für den Momentaufnahmerestore sind *Instanz-* und *Datenbank*verbindungen erforderlich.

Befehlssyntax

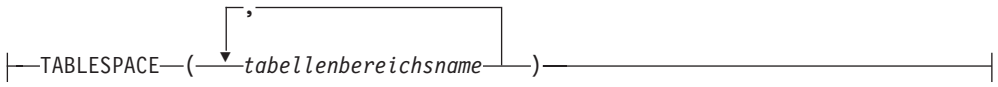


restoreoptionen:

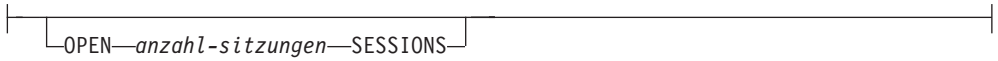




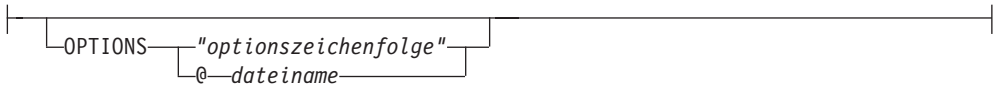
rebuildklausel-für-tabellenbereich:



geöffnete-sitzungen:



optionen:



Befehlsparameter

DATABASE *aliasname-der-quellendatenbank*
Der Aliasname der Quellendatenbank, aus der das Backup stammt.

CONTINUE
Gibt an, dass die Container neu definiert wurden und der abschließende Schritt einer umgeleiteten Restoreoperation ausgeführt werden sollte.

ABORT
Dieser Parameter:

- Stoppt eine umgeleitete Restoreoperation. Dies ist hilfreich, wenn ein Fehler aufgetreten ist, der die Wiederholung mindestens eines Schrittes erforderlich macht. Nach dem Absetzen von **RESTORE DATABASE** mit der Option **ABORT** muss jeder Schritt einer umgeleiteten Restoreoperation wiederholt werden, einschließlich **RESTORE DATABASE** mit der Option **REDIRECT**.
- Beendet eine inkrementelle Restoreoperation, bevor sie abgeschlossen ist.

USER *benutzername*
Gibt den zu verwendenden Benutzernamen beim Herstellen einer Verbindung zur Datenbank an.

USING *kennwort*
Das zur Authentifizierung des Benutzernamens verwendete Kennwort. Wenn das Kennwort übergangen wird, wird der Benutzer aufgefordert, es einzugeben.

REBUILD WITH ALL TABLESPACES IN DATABASE
Stellt die Datenbank mit allen Tabellenbereichen wieder her, die der Daten-

bank zum Zeitpunkt des wiederherzustellenden Images bekannt waren. Bei dieser Restoreoperation wird eine bereits vorhandene Datenbank überschrieben.

REBUILD WITH ALL TABLESPACES IN DATABASE EXCEPT *rebuildklausel-für-tabellenbereich*
Stellt die Datenbank mit allen Tabellenbereichen wieder her, die der Datenbank zum Zeitpunkt des wiederherzustellenden Images bekannt waren, mit Ausnahme der in der Liste angegebenen Tabellenbereiche. Bei dieser Restoreoperation wird eine bereits vorhandene Datenbank überschrieben.

REBUILD WITH ALL TABLESPACES IN IMAGE
Stellt die Datenbank nur mit den Tabellenbereichen aus dem wiederherzustellenden Image wieder her. Bei dieser Restoreoperation wird eine bereits vorhandene Datenbank überschrieben.

REBUILD WITH ALL TABLESPACES IN IMAGE EXCEPT *rebuildklausel-für-tabellenbereich*
Stellt die Datenbank nur mit den Tabellenbereichen aus dem wiederherzustellenden Image wieder her, mit Ausnahme der in der Liste angegebenen Tabellenbereiche. Bei dieser Restoreoperation wird eine bereits vorhandene Datenbank überschrieben.

REBUILD WITH *rebuildklausel-für-tabellenbereich*
Stellt die Datenbank nur mit den in der Liste aufgeführten Tabellenbereichen wieder her. Bei dieser Restoreoperation wird eine bereits vorhandene Datenbank überschrieben.

TABLESPACE *tabellenbereichsname*
Eine Namensliste zum Angeben der Tabellenbereiche, die wiederhergestellt werden sollen.

Wenn die Option **TRANSPORT** festgelegt ist, müssen Tabellenbereichsnamen angegeben werden.

SCHEMA *schemaname*
Eine Namensliste zum Angeben der Schemas, die wiederhergestellt werden sollen.

Wenn die Option **TRANSPORT** festgelegt ist, müssen Schemanamen angegeben werden. Die Option **SCHEMA** ist nur gültig, wenn die Option **TRANSPORT** angegeben wird.

ONLINE
Mit diesem Schlüsselwort (nur anwendbar für Restoreoperationen auf Tabellenbereichsebene) kann angegeben werden, dass ein Backup-Image online wiederhergestellt werden darf. Dies bedeutet, dass andere Agenten Verbindung zu dieser Datenbank herstellen können, während das Backup-Image wiederhergestellt wird, und dass die Daten in anderen Tabellenbereichen zur Verfügung stehen, während die angegebenen Tabellenbereiche wiederhergestellt werden.

HISTORY FILE
Dieses Schlüsselwort gibt an, dass nur die Protokolldatei aus dem Backup-Image wiederhergestellt werden soll.

COMPRESSION LIBRARY
Dieses Schlüsselwort gibt an, dass nur das Komprimierungswörterbuch aus dem Backup-Image wiederhergestellt werden soll. Wenn das Objekt im

Backup-Image enthalten ist, wird es in das Datenbankverzeichnis wiederhergestellt. Ist das Objekt nicht im Backup-Image enthalten, schlägt die Restoreoperation fehl.

LOGS Dieses Schüsselwort gibt an, dass nur die im Backup-Image enthaltenen Protokolldateien wiederhergestellt werden sollen. Enthält das Backup-Image keine Protokolldateien, schlägt die Restoreoperation fehl. Wenn diese Option angegeben wird, muss die Option **LOGTARGET** ebenfalls angegeben werden.

INCREMENTAL
Ohne weitere Parameter bezeichnet **INCREMENTAL** eine manuelle kumulative Restoreoperation. Beim manuellen Restore muss der Benutzer jeden Restorebefehl für jedes an der Restoreoperation beteiligte Image manuell absetzen. Verwenden Sie dabei diese Reihenfolge: Letztes, erstes, zweites, drittes usw. bis zum letzten Image (einschließlich).

INCREMENTAL AUTOMATIC/AUTO
Gibt eine automatische kumulative Restoreoperation an.

INCREMENTAL ABORT
Gibt an, dass eine laufende, manuelle kumulative Restoreoperation abgebrochen werden soll.

USE

TSM Gibt an, dass die Restoreoperation für die Datenbank mit Tivoli Storage Manager (TSM) als Zieleinheit durchgeführt werden soll.

XBSA Gibt an, dass die XBSA-Schnittstelle verwendet werden soll. XBSA (Backup Services APIs) ist eine offene Anwendungsprogrammierschnittstelle für Anwendungen oder Funktionen, die Datenspeicherungsmanagement für Backup- oder Archivierungszwecke benötigen.

SNAPSHOT
Gibt an, dass die Daten aus einem Momentaufnahmebackup wiederhergestellt werden sollen.

Der Parameter **SNAPSHOT** kann nicht zusammen mit einem der folgenden Parameter verwendet werden:

- **INCREMENTAL**
- **TO**
- **ON**
- **DBPATH ON**
- **INTO**
- **NEWLOGPATH**
- **WITH** *anzahl_puffer* **BUFFERS**
- **BUFFER**
- **REDIRECT**
- **REPLACE HISTORY FILE**
- **COMPRESSION LIBRARY**
- **PARALLELISM**
- **COMPRLIB**
- **OPEN** *anzahl-sitzungen* **SESSIONS**
- **HISTORY FILE**

- **LOGS**

Außerdem kann der Parameter **SNAPSHOT** nicht in Verbindung mit einer Restoreoperation verwendet werden, die eine Tabellenbereichsliste enthält (dies schließt die Option **REBUILD WITH** ein).

Das Standardverhalten beim Restore von Daten aus einem Momentaufnahme-Backup-Image ist ein vollständiger Offline-Restore der Datenbank für alle Pfade, die zu der Datenbank gehören, einschließlich aller Container, des lokalen Datenträgerverzeichnisses und des Datenbankpfads (DBPATH). Die Protokolle sind bei einem Momentaufnahmerestore nicht eingeschlossen, es sei denn, der Parameter **LOGTARGET INCLUDE** wird angegeben; der Parameter **LOGTARGET EXCLUDE** ist die Standardeinstellung für alle Momentaufnahmerestores. Wenn Sie eine Zeitmarke angeben, wird das Momentaufnahme-Backup-Image mit der angegebenen Zeitmarke für den Restore verwendet.

LIBRARY *bibliotheksname*
In IBM Data Server ist ein DB2 ACS-API-Treiber für die folgende Speicherhardware integriert:

- IBM TotalStorage SAN Volume Controller
- IBM Enterprise Storage Server Modell 800
- IBM Storwize V7000
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- IBM XIV

Wenn Sie über andere Speicherhardware sowie über einen DB2 ACS-API-Treiber für diese Hardware verfügen, können Sie den DB2 ACS-API-Treiber mit dem Parameter **LIBRARY** angeben.

Der Wert für den Parameter **LIBRARY** ist der vollständig qualifizierte Name der Bibliotheksdatei.

OPTIONS

"*optionszeichenfolge***"**
Gibt an, welche Optionen für die Restoreoperation verwendet werden sollen. Die Zeichenfolge wird genauso übergeben, wie sie eingegeben wurde (ohne die doppelten Anführungszeichen).

@dateiname
Gibt an, dass die für die Restoreoperation zu verwendenden Optionen in einer Datei enthalten sind, die sich auf dem DB2-Server befindet. Die Zeichenfolge wird an die Unterstützungsbibliothek des Herstellers übergeben. Der Dateiname muss ein vollständig qualifizierter Dateiname sein.

Mit dem Datenbankkonfigurationsparameter **VENDOROPT** können keine herstellerspezifischen Optionen für Momentaufnahmerestoreoperationen angegeben werden. Zu diesem Zweck müssen Sie stattdessen den Parameter **OPTIONS** der Restoredienstprogramme verwenden.

OPEN *anzahl-sitzungen* **SESSIONS**
Die Anzahl E/A-Sitzungen, die mit TSM oder einem Programm eines anderen Anbieters verwendet werden soll.

FROM *verzeichnis/einheit*
Der vollständig qualifizierte Pfadname des Verzeichnisses oder der Einheit, in dem bzw. auf der sich das Backup-Image befindet. Wenn die Optionen **USE TSM**, **FROM** und **LOAD** fehlen, wird standardmäßig das aktuelle Arbeitsverzeichnis der Clientmaschine verwendet. Dieses Zielverzeichnis bzw. diese Zieleinheit muss auf dem Zielserver bzw. der Zielinstanz vorhanden sein.

Wenn mehrere Elemente angegeben werden und das letzte Element ein Bandlaufwerk ist, wird der Benutzer aufgefordert, ein anderes Band einzulegen. Gültige Antwortoptionen sind:

c Fortsetzen. Verwenden der Einheit fortsetzen, die die Warnung generiert hat (zum Beispiel, wenn ein neues Band eingelegt wurde).

d Einheit beenden. Verwendung *nur* der Einheit beenden, die die Warnung generiert hat (zum Beispiel, wenn keine Bänder mehr vorhanden sind).

t Beenden. Restoreoperation abbrechen, nachdem der Benutzer eine vom Dienstprogramm angeforderte Aktion nicht ausgeführt hat.

LOAD *gemeinsam-genutzte-bibliothek*
Der Name der gemeinsam genutzten Bibliothek (DLL unter Windows-Betriebssystemen), in der die zu verwendenden E/A-Funktionen für Backup und Restore des Herstellers enthalten sind. Der Name kann einen vollständigen Pfad enthalten. Wenn der vollständige Pfad nicht angegeben wird, wird standardmäßig der Pfad verwendet, in dem sich das Benutzerexitprogramm befindet.

TAKEN AT *datum-zeit*
Die Zeitmarke des Datenbank-Backup-Images. Die Zeitmarke wird nach erfolgreicher Beendigung der Backupoperation angezeigt; sie ist Bestandteil des Pfadnamens für das Backup-Image. Die Zeitmarke hat das Format *jjjjmmtthhmmss*. Die Zeitmarke kann auch teilweise angegeben werden. Sind beispielsweise zwei verschiedene Backup-Images mit den Zeitmarken 20021001010101 und 20021002010101 vorhanden, wird bei Angabe von `20021002` das Image mit der Zeitmarke `20021002010101` verwendet. Ist kein Wert für diesen Parameter angegeben, dürfen die Quellendatenträger nur ein Backup-Image enthalten.

TO *zielverzeichnis*
Dieser Parameter gibt das Zieldatenbankverzeichnis an. Dieser Parameter wird ignoriert, wenn die Restoreoperation in eine vorhandene Datenbank erfolgt. Das von Ihnen angegebene Laufwerk und Verzeichnis muss lokal vorhanden sein. Wenn das Backup-Image eine Datenbank mit aktiviertem dynamischem Speicher enthält, wird nur das Datenbankverzeichnis geändert, die zugeordneten Speicherpfade der Datenbank werden beibehalten.

DBPATH ON *zielverzeichnis*
Dieser Parameter gibt das Zieldatenbankverzeichnis an. Dieser Parameter wird ignoriert, wenn die Restoreoperation in eine vorhandene Datenbank erfolgt. Das von Ihnen angegebene Laufwerk und Verzeichnis muss lokal vorhanden sein. Wenn das Backup-Image eine Datenbank mit aktiviertem dynamischem Speicher enthält und der Parameter **ON** nicht angegeben wird, ist dieser Parameter synonym mit dem Parameter **TO**, d. h. nur das Datenbankverzeichnis wird geändert, die zugeordneten Speicherpfade der Datenbank werden jedoch beibehalten.

ON *pfadliste*

Dieser Parameter definiert die Speicherpfade, die einer Datenbank zugeordnet sind, erneut. Wenn die Datenbank mehrere Speichergruppen enthält, leitet diese Option alle Speichergruppen in die angegebenen Pfade um. Dabei wird so vorgegangen, dass jede definierte Speichergruppe *pfadliste* als neuen Pfad der Speichergruppe verwendet. Die Verwendung dieses Parameters mit einer Datenbank, für die keine Speichergruppen definiert sind oder für die der dynamische Speicher nicht aktiviert wurde, verursacht einen Fehler (SQL20321N). Die im Backup-Image definierten Speicherfade werden nicht länger verwendet und die Tabellenbereiche mit dynamischem Speicher werden automatisch in die neuen Pfade umgeleitet. Wenn dieser Parameter für eine Datenbank mit dynamischem Speicher nicht angegeben ist, werden die im Backup-Image definierten Speicherpfade unverändert übernommen. Wenn mehrere Pfade angegeben werden, sind diese durch Kommas zu trennen. Jeder Pfad muss über einen absoluten Pfadnamen verfügen und lokal vorhanden sein.

Wenn diese Option zusammen mit der Option **REDIRECT** angegeben wird, dann wird sie wirksam, bevor der ursprüngliche Befehl **RESTORE ... REDIRECT** die Steuerung an das aufrufende Modul zurückgibt und bevor alle Anweisungen SET STOGROUP PATHS oder SET TABLESPACE CONTAINERS abgesetzt werden. Wenn Speichergruppenpfade nachfolgend umgeleitet werden, dann überschreiben solche Änderungen alle Pfade, die im ursprünglichen Befehl **RESTORE ... ON** *pfadliste* angegeben wurden.

Für alle Speichergruppen, deren Pfade während einer Restoreoperation umdefiniert wurden, werden Operationen für die Speicherpfade während einer nachfolgenden aktualisierenden Recovery nicht wiederholt.

Wenn die Datenbank auf dem Datenträger noch nicht vorhanden ist und der Parameter **DBPATH ON** nicht angegeben wird, wird der erste Pfad als Zieldatenbankverzeichnis verwendet.

Für eine Mehrpartitionsdatenbank kann die Option **ON pfadliste** nur in der Katalogpartition angegeben werden. Die Katalogpartition muss vor anderen Partitionen wiederhergestellt werden, wenn die Option ON verwendet wird. Durch die Restoreoperation der Katalogpartition mit neuen Speicherpfaden werden alle Datenbankpartitionen, die sich nicht im Katalog befinden, in den Status RESTORE_PENDING versetzt. Danach können die nicht im Katalog enthaltenen Datenbankpartitionen parallel wiederhergestellt werden, ohne die Klausel **ON** im Befehl **RESTORE** anzugeben.

Im Allgemeinen müssen für jede Partition einer Mehrpartitionsdatenbank dieselben Speicherpfade verwendet werden. Diese Speicherpfade müssen vor dem Ausführen des Befehls **RESTORE DATABASE** vorhanden sein. Diese Regel gilt nicht bei Verwendung von Datenbankpartitionsausdrücken innerhalb des Speicherpfads. Dieses Verfahren bietet die Möglichkeit, die Datenbankpartitionsnummer in die Speicherpfade einzufügen, sodass der resultierende Pfadname für jede Partition unterschiedlich ist.

Wenn der Befehl RESTORE mit der Klausel ON verwendet wird, dann hat dies dieselben Auswirkungen wie eine umgeleitete Restoreoperation.

Der Parameter **ON** kann nicht für die erneute Definition von Speicherpfaden für den Schematransport verwendet werden. Für den Schematransport werden bereits vorhandene Speicherpfade in der Zieldatenbank verwendet.

INTO *aliasname-der-zieldatenbank*
Der Aliasname der Zieldatenbank. Wenn die Zieldatenbank nicht vorhanden ist, wird sie erstellt.

Beim Restore eines Datenbankbackups in eine vorhandene Datenbank übernimmt die wiederhergestellte Datenbank den Aliasnamen und den Datenbanknamen der vorhandenen Datenbank. Beim Restore eines Datenbankbackups in eine nicht vorhandene Datenbank wird die neue Datenbank mit dem von Ihnen angegebenen Aliasnamen und Datenbanknamen erstellt. Dieser neue Datenbankname muss auf dem Zielsystem der Restoreoperation eindeutig sein.

TRANSPORT INTO *aliasname-der-zieldatenbank*
Gibt den vorhandenen Aliasnamen der Zieldatenbank für eine Transportoperation an. Die Tabellenbereiche und Schemas, die transportiert werden, werden der Datenbank hinzugefügt.

Die Optionen **TABLESPACE** und **SCHEMA** müssen Tabellenbereichs- und Schemanamen angeben, die eine gültige transportierbare Gruppe definieren. Andernfalls schlägt die Transportoperation fehl. SQLCODE=SQL2590N rc=1.

Die Systemkataloge können nicht transportiert werden. SQLCODE=SQL2590N rc=4.

Nachdem die Schemata mithilfe des Befehls **RESTORE** geprüft wurden, werden die Systemkatalogeinträge, die die Objekte in den transportierten Tabellenbereichen beschreiben, in der Zieldatenbank erstellt. Nach Abschluss der erneuten Erstellung der Schemata übernimmt die Zieldatenbank das Eigentumsrecht an den physischen Tabellenbereichscontainern.

Die in den wiederherzustellenden Tabellenbereichen enthaltenen physischen und logischen Objekte werden in der Zieldatenbank erneut erstellt und die Tabellenbereichsdefinitionen und -container werden zur Zieldatenbank hinzugefügt. Das Fehlschlagen einer Objekterstellung oder der Wiedergabe der DDL führt zu einem Fehler.

STAGE IN *bereitstellungsdatenbank*
Gibt den Namen einer temporären Bereitstellungsdatenbank für das Back-up-Image an, die die Quelle für die Transportoperation ist. Wird die Option**STAGE IN** angegeben, wird die temporäre Datenbank nach Abschluss der Transportoperation nicht gelöscht. Die Datenbank wird jedoch nach dem Transport nicht mehr benötigt und kann vom Datenbankadministrator gelöscht werden.

Folgendes trifft zu, wenn die Option **STAGE IN** nicht angegeben ist:

- Der Datenbankname weist das Format SYSTG*xxx* auf, wobei *xxx* einen Ganzzahlwert darstellt.
- Die temporäre Bereitstellungsdatenbank wird nach Abschluss der Transportoperation gelöscht.

USING STOGROUP *speichergruppenname*
Für Tabellenbereiche mit dynamischem Speicher gibt dieses Element die Zielspeichergruppe an, die allen transportierten Tabellenbereichen zugeordnet wird. Wenn die Speichergruppe nicht angegeben wird, dann wird die momentan designierte Standardspeichergruppe der Zieldatenbank verwendet. Diese Klausel gilt nur für Tabellenbereiche mit dynamischem Speicher und ist nur während einer Schemaoperation `transport` zulässig.

Identifiziert die Speichergruppe, in der die Tabellenbereichsdaten gespeichert werden. In *speichergruppenname* muss eine Speichergruppe angegeben werden, die unter *aliasname-der-zieldatenbank* für die Operation **TRANSPORT** vorhanden ist. (SQLSTATE 42704). Diese Name besteht aus einem Teil.

LOGTARGET *verzeichnis*

Restoreoperationen ohne Momentaufnahme:

Der absolute Pfadname eines vorhandenen Verzeichnisses auf dem Datenbankserver, das als Zielverzeichnis zum Extrahieren von Protokolldateien aus einem Backup-Image verwendet werden soll. Wenn diese Option angegeben wird, werden alle in dem Backup-Image enthaltenen Protokolldateien in das Zielverzeichnis extrahiert. Wenn diese Option nicht angegeben ist, werden die in einem Backup-Image enthaltenen Protokolldateien nicht extrahiert. Geben Sie die Option **LOGS** an, um nur die Protokolldateien aus dem Backup-Image zu extrahieren. Diese Option fügt automatisch die Datenbankpartitionsnummer und eine Protokolldatenstrom-ID zum Pfad hinzu.

DEFAULT

Führen Sie eine Restoreoperation für die Protokolldateien vom Backup-Image in das Standardprotokollverzeichnis der Datenbank (z. B. `/home/db2user/db2inst/NODE0000/SQL00001/LOGSTREAM0000`) durch.

Restoreoperationen aus Momentaufnahmen:

INCLUDE

Protokollverzeichnisdatenträger aus dem Momentaufnahme-Image wiederherstellen. Wenn diese Option angegeben ist und das Backup-Image Protokollverzeichnisse enthält, werden diese wiederhergestellt. Vorhandene Protokollverzeichnisse und Protokolldateien auf dem Datenträger bleiben erhalten, wenn sie nicht mit den Protokollverzeichnissen im Backup-Image in Konflikt stehen. Stehen vorhandene Protokollverzeichnisse in Konflikt mit den Protokollverzeichnissen im Backup-Image, wird ein Fehler zurückgegeben.

EXCLUDE

Protokollverzeichnisdatenträger nicht wiederherstellen. Bei Angabe dieser Option werden keine Protokollverzeichnisse aus dem Backup-Image wiederhergestellt. Vorhandene Protokollverzeichnisse und Protokolldateien auf dem Datenträger bleiben erhalten, wenn sie nicht mit den Protokollverzeichnissen im Backup-Image in Konflikt stehen. Wird beim Restore eines zur Datenbank gehörenden Pfads implizit ein Protokollverzeichnis wiederhergestellt und dabei ein Protokollverzeichnis überschrieben, wird ein Fehler zurückgegeben.

FORCE

Erlaubt beim Wiederherstellen eines Momentaufnahme-Images das Überschreiben und Ersetzen vorhandener Protokollverzeichnisse in der aktuellen Datenbank. Ohne diese Option schlägt die Restoreoperation fehl, wenn Konflikte zwischen vorhandenen Protokollverzeichnissen und -dateien auf dem Datenträger und Protokollverzeichnissen im Momentaufnahme-Image bestehen. Geben Sie diese Option an, um zuzulassen, dass die vorhandenen Protokollverzeichnisse bei der Restoreoperation überschrieben und ersetzt werden können.

Anmerkung: Verwenden Sie diese Option mit Vorsicht, und stellen Sie dabei sicher, dass alle Protokolle gesichert und archiviert sind, die für die Recovery erforderlich sein können.

Bei Momentaufnahmerestores lautet der Standardwert der Verzeichnisoption **LOGTARGET EXCLUDE**.

NEWLOGPATH *verzeichnis*
Der absolute Pfadname eines Verzeichnisses, das nach der Restoreoperation für aktive Protokolldateien verwendet wird. Dieser Parameter hat dieselbe Funktion wie der Datenbankkonfigurationsparameter **newlogpath**. Der Parameter kann verwendet werden, wenn der Protokollpfad im Backup-Image nach erfolgter Restoreoperation nicht weiter verwendet werden kann (z. B. weil der Pfad ungültig ist oder von einer anderen Datenbank verwendet wird).

Anmerkung: Wenn der Befehlsparameter **newlogpath** definiert ist, wird die Knotennummer nicht automatisch an den Wert für **logpath** angehängt, ebenso wie bei der Aktualisierung des Datenbankkonfigurationsparameters **newlogpath**. Weitere Informationen hierzu finden Sie im Abschnitt newlogpath - Datenbankprotokollpfad ändern.

DEFAULT
Nach Abschluss der Restoreoperation muss die Datenbank das Standardprotokollverzeichnis (`/home/db2user/db2inst/NODE0000/SQL00001/LOGSTREAM0000`) für die Protokollierung verwenden.

WITH *anzahl-puffer* **BUFFERS**
Die Anzahl der Puffer, die verwendet werden sollen. Das DB2-Datenbanksystem wählt automatisch einen optimalen Wert für diesen Parameter. Eine größere Anzahl Puffer kann verwendet werden, um die Leistung zu verbessern, wenn aus mehreren Quellen gelesen wird oder wenn der Wert für **PARALLELISM** erhöht wurde.

BUFFER *puffergröße*
Die Größe des für die Restoreoperation verwendeten Puffers in Seiten. Das DB2-Datenbanksystem wählt automatisch einen optimalen Wert für diesen Parameter. Der Mindestwert für diesen Parameter ist 8 Seiten.

Die Größe der Restorepuffer muss eine positive ganze Zahl und ein Vielfaches der Backup-Puffergröße sein, die bei der Backup-Operation angegeben wurde. Wird eine nicht korrekte Puffergröße angegeben, erhalten die zugeordneten Puffer die kleinstmögliche Größe.

REPLACE HISTORY FILE
Gibt an, dass die Restoreoperation die Protokolldatei auf dem Datenträger durch die Protokolldatei aus dem Backup-Image ersetzen soll.

REPLACE EXISTING
Wenn bereits eine Datenbank mit demselben Aliasnamen wie die Zieldatenbank vorhanden ist, gibt dieser Parameter an, dass das Restoredienstprogramm die vorhandene Datenbank durch die wiederhergestellte Datenbank ersetzen soll. Dies ist hilfreich für Scripts, die das Restoredienstprogramm aufrufen, weil der Befehlszeilenprozessor den Benutzer nicht auffordert, das Löschen einer vorhandenen Datenbank zu prüfen. Wenn der Parameter **WITHOUT PROMPTING** angegeben wird, muss **REPLACE EXISTING** nicht angegeben werden, aber in diesem Fall schlägt die Operation fehl, wenn Ereignisse auftreten, die normalerweise einen Benutzereingriff erfordern.

REDIRECT
Gibt eine umgeleitete Restoreoperation an. Um eine umgeleitete Restoreoperation abzuschließen, sollte auf diesen Befehl mindestens ein Befehl **SET**

TABLESPACE CONTAINERS oder **SET STOGROUP PATHS** folgen und anschließend ein Befehl **RESTORE DATABASE** mit der Option **CONTINUE**. Beispiel:

```
RESTORE DB SAMPLE REDIRECT

SET STOGROUP PATHS FOR sg_hot ON '/ssd/fs1', '/ssd/fs2'
SET STOGROUP PATHS FOR sg_cold ON '/hdd/path1', '/hdd/path2'

RESTORE DB SAMPLE CONTINUE
```

Wenn eine Speichergruppe seit der Erstellung des Backup-Images umbenannt wurde, dann verweist der im Befehl **SET STOGROUP PATHS** angegebene Speichergruppenname auf den Speichergruppennamen aus dem Backup-Image und nicht auf den aktuellsten Namen.

Alle Befehle, die einer einzigen, umgeleiteten Restoreoperation zugeordnet sind, müssen im selben Fenster oder in derselben CLP-Sitzung aufgerufen werden.

GENERATE SCRIPT *script*
Erstellt ein Script für umgeleiteten Restore mit dem angegebenen Dateinamen. Der Scriptname kann relativ oder absolut sein, und das Script wird auf der Clientseite generiert. Wenn die Datei nicht auf der Clientseite erstellt werden kann, wird eine Fehlernachricht (SQL9304N) zurückgegeben. Wenn die Datei bereits vorhanden ist, wird sie überschrieben. Weitere Informationen zur Verwendung finden Sie in den nachfolgenden Beispielen.

WITHOUT ROLLING FORWARD
Gibt an, dass die Datenbank nicht in den Status für anstehende aktualisierende Recovery versetzt werden soll, nachdem sie erfolgreich wiederhergestellt wurde.

Wenn sich die Datenbank nach einer erfolgreichen Restoreoperation im Status für anstehende aktualisierende Recovery befindet, muss der Befehl **ROLLFORWARD** aufgerufen werden, bevor die Datenbank wieder verwendet werden kann.

Wenn diese Option für eine Restoreoperation aus einem Online-Backup-Image angegeben wird, wird der Fehler SQL2537N zurückgegeben.

Wenn das Backup-Image eine wiederherstellbare Datenbank enthält, darf **WITHOUT ROLLING FORWARD** nicht mit der Option **REBUILD** angegeben werden.

PARALLELISM *n*
Gibt an, wie viele Puffermanipulatoren während der Restoreoperation erstellt werden sollen. Das DB2-Datenbanksystem wählt automatisch einen optimalen Wert für diesen Parameter.

COMPRLIB *name*
Gibt den Namen der Bibliothek an, die für die Dekomprimierung verwendet werden soll (z. B. `db2compr.dll` für Windows- oder `libdb2compr.so` für Linux- oder UNIX-Systeme). Der Name muss ein vollständig qualifizierter Pfad sein, der auf eine Datei auf dem Server verweist. Wenn dieser Parameter nicht angegeben wird, versucht das DB2-Datenbanksystem, die in dem Image gespeicherte Bibliothek zu verwenden. Wenn das Backup nicht komprimiert war, wird der Wert dieses Parameters ignoriert. Wenn die angegebene Bibliothek nicht geladen werden kann, schlägt die Restoreoperation fehl.

COMPROPTS *zeichenfolge*
Beschreibt einen Block mit Binärdaten, der an die Initialisierungsroutine der Dekomprimierungsbibliothek übergeben wird. Das DB2-Datenbanksys-

tem leitet diese Zeichenfolge direkt vom Client zum Server weiter, d. h. alle Probleme durch Bytefolgeumkehrung oder Codepagekonvertierung werden von der Dekomprimierungsbibliothek verarbeitet. Wenn das erste Zeichen des Datenblocks „@" ist, werden die übrigen Daten vom DB2-Datenbanksystem als der Name einer Datei auf dem Server angesehen. Das DB2-Datenbanksystem ersetzt den Inhalt von *zeichenfolge* durch den Inhalt dieser Datei und gibt den neuen Wert an die Initialisierungsroutine weiter. Die maximal zulässige Länge für die Zeichenfolge beträgt 1.024 Byte.

WITHOUT PROMPTING
Gibt an, dass die Restoreoperation nicht überwacht ausgeführt werden soll. Bei allen Aktionen, die normalerweise einen Benutzereingriff erfordern, wird eine Fehlernachricht zurückgegeben. Bei Verwendung eines austauschbaren Datenträgers wie Band oder Diskette wird der Benutzer zum Wechseln des Datenträgers aufgefordert, selbst wenn diese Option angegeben ist.

Beispiele

1. Im folgenden Beispiel ist die Datenbank WSDB auf allen 4 Datenbankpartitionen mit den Nummern 0 bis 3 definiert. Der Pfad `/dev3/backup` ist über alle Datenbankpartitionen zugänglich. Die folgenden Offline-Backup-Images sind in `/dev3/backup` verfügbar:

```
wsdb.0.db2inst1.DBPART000.200802241234.001
wsdb.0.db2inst1.DBPART001.200802241234.001
wsdb.0.db2inst1.DBPART002.200802241234.001
wsdb.0.db2inst1.DBPART003.200802241234.001
```

   Um zuerst die Katalogpartition und anschließend alle anderen Partitionen der Datenbank WSDB aus dem Verzeichnis `/dev3/backup` wiederherzustellen, setzen Sie die folgenden Befehle in einer der Datenbankpartitionen ab:

```
db2_all '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234149
  INTO wsdb REPLACE EXISTING'
db2_all '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234427
  INTO wsdb REPLACE EXISTING'
db2_all '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234828
  INTO wsdb REPLACE EXISTING'
db2_all '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331235235
  INTO wsdb REPLACE EXISTING'
```

   Das Dienstprogramm **db2_all** setzt den Restorebefehl an jede angegebene Datenbankpartition ab. Beim Ausführen einer Restoreoperation mit `db2_all` sollten Sie unbedingt **REPLACE EXISTING** und/oder **WITHOUT PROMPTING** angeben. Andernfalls scheint die Anwendung zu blockieren, wenn Eingabeaufforderungen ausgegeben werden. Dies liegt daran, dass **db2_all** keine Benutzereingabeaufforderung unterstützt.

2. Es folgt ein typisches Beispielszenario für den umgeleiteten Restore einer Datenbank mit dem Aliasnamen MEINEDB:
   a. Führen Sie einen Befehl **RESTORE DATABASE** mit der Option **REDIRECT** aus.

```
restore db meinedb replace existing redirect
```

      Nach der erfolgreichen Ausführung von Schritt 1 und vor der Vollendung von Schritt 3 kann die Restoreoperation durch Absetzen des folgenden Befehls abgebrochen werden:

```
restore db meinedb abort
```

b. Setzen Sie einen Befehl **SET TABLESPACE CONTAINERS** für jeden Tabellenbereich ab, dessen Container erneut definiert werden müssen. Beispiel:

```
set tablespace containers for 5 using
        (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
```

Setzen Sie den Befehl **LIST TABLESPACE CONTAINERS** ab, um sicherzustellen, dass es sich bei den Containern der wiederhergestellten Datenbank um die in diesem Schritt angegebenen Container handelt.

c. Nach der erfolgreichen Durchführung der Schritte 1 und 2 setzen Sie den folgenden Befehl ab:

```
restore db meinedb continue
```

Dies ist der letzte Schritt des umgeleiteten Restores.

d. Falls Schritt 3 fehlschlägt oder die Restoreoperation abgebrochen wurde, kann der umgeleitete Restore erneut gestartet werden, indem Sie wieder bei Schritt 1 beginnen.

3. Im Folgenden sehen Sie ein Beispiel für eine wöchentliche inkrementelle Backup-Strategie für eine wiederherstellbare Datenbank. Diese Strategie umfasst ein wöchentliches Gesamtbackup der Datenbank, ein tägliches nicht kumulatives Backup (Delta-Backup) sowie ein kumulatives Backup (inkrementell) in der Wochenmitte:

```
(So) backup db meinedb use tsm
(Mo) backup db meinedb online incremental delta use tsm
(Di) backup db meinedb online incremental delta use tsm
(Mi) backup db meinedb online incremental use tsm
(Do) backup db meinedb online incremental delta use tsm
(Fr) backup db meinedb online incremental delta use tsm
(Sa) backup db meinedb online incremental use tsm
```

Setzen Sie für ein automatisches Restore der Datenbank aus den am Freitagmorgen erstellten Images den folgenden Befehl ab:

```
restore db meinedb incremental automatic taken at (Fr)
```

Setzen Sie für ein manuelles Restore der Datenbank aus den am Freitagmorgen erstellten Images den folgenden Befehl ab:

```
restore db meinedb incremental taken at (Fr)
restore db meinedb incremental taken at (So)
restore db meinedb incremental taken at (Mi)
restore db meinedb incremental taken at (Do)
restore db meinedb incremental taken at (Fr)
```

4. Geben Sie Folgendes ein, um ein Backup-Image, das Protokolle einschließt, für die Übertragung auf ein fernes System zu erstellen:

```
backup db sample online to /dev3/backup include logs
```

Um dieses Backup-Image wiederherzustellen, geben Sie einen **LOGTARGET**-Pfad an. Der Pfad muss innerhalb von **ROLLFORWARD** angegeben werden:

```
restore db sample from /dev3/backup logtarget /dev3/logs
rollforward db sample to end of logs and stop overflow log path /dev3/logs
```

5. Geben Sie Folgendes ein, um nur die Protokolldateien aus einem Backup-Image, das Protokolle einschließt, abzurufen:

```
restore db sample logs from /dev3/backup logtarget /dev3/logs
```

6. Im folgenden Beispiel werden drei identische Zielverzeichnisse für eine Backup-Operation für die Datenbank SAMPLE angegeben. Das Backup der Daten

erfolgt gleichzeitig in die drei Zielverzeichnisse und es werden drei Backup-Images mit den Erweiterungen .001, .002 und .003 generiert.

```
backup db sample to /dev3/backup, /dev3/backup, /dev3/backup
```

Geben Sie Folgendes ein, um das Backup-Image aus den Zielverzeichnis wiederherzustellen:

```
restore db sample from /dev3/backup, /dev3/backup, /dev3/backup
```

7. Mit den Schlüsselwörtern **USE TSM OPTIONS** können die TSM-Informationen angegeben werden, die für die Restoreoperation verwendet werden sollen. Lassen Sie auf Windows-Plattformen die Option `-fromowner` weg.
   - Angabe einer Zeichenfolge mit Begrenzer:

     ```
     restore db sample use TSM options '"-fromnode=bar -fromowner=dmcinnis"'
     ```

   - Angabe eines vollständig qualifizierten Dateinamens:

     ```
     restore db sample use TSM options @/u/dmcinnis/meineoptionen.txt
     ```

     Die Datei `meineoptionen.txt` enthält die folgenden Informationen:
     `-fromnode=bar -fromowner=dmcinnis`
8. Das folgende Beispiel enthält eine einfache Restoreoperation für eine Mehrpartitionsdatenbank mit aktiviertem dynamischem Speicher und neuen Speicherpfaden. Die Datenbank wurde ursprünglich mit einem Speicherpfad (`/meinPfad0:`) erstellt.
   - Setzen Sie in der Katalogpartition den folgenden Befehl ab: `restore db meinedb on /meinPfad1,/meinPfad2`
   - Setzen Sie in allen Nichtkatalogpartitionen den folgenden Befehl ab: `restore db meinedb`
9. Eine Scriptausgabe auf einer Datenbank ohne dynamischen Speicher für den Befehl

   ```
   restore db sample from /home/jseifert/backups taken at 20050301100417 redirect
   generate script SAMPLE_NODE0000.clp
   ```

   würde wie folgt aussehen:

   ```
   -- *****************************************************************************
   -- ** automatically created redirect restore script
   -- *****************************************************************************
   UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
   SET CLIENT ATTACH_DBPARTITIONNUM  0;
   SET CLIENT CONNECT_DBPARTITIONNUM 0;
   -- *****************************************************************************
   -- ** initialize redirected restore
   -- *****************************************************************************
   RESTORE DATABASE SAMPLE
   -- USER  '<benutzername>'
   -- USING '<kennwort>'
   FROM '/home/jseifert/backups'
   TAKEN AT 20050301100417
   -- DBPATH ON '<zielverzeichnis>'
      INTO SAMPLE
   -- NEWLOGPATH '/home/jseifert/jseifert/SAMPLE/NODE0000/LOGSTREAM0000/'
      -- WITH <pufferanzahl> BUFFERS
      -- BUFFER <puffergröße>
      -- REPLACE HISTORY FILE
      -- REPLACE EXISTING
   REDIRECT
      -- PARALLELISM <n>
      -- WITHOUT ROLLING FORWARD
      -- WITHOUT PROMPTING
   ;
   -- *****************************************************************************
   ```

```
-- ** tablespace definition
-- ********************************************************************************
-- ********************************************************************************
-- ** Tablespace name                         = SYSCATSPACE
-- **   Tablespace ID                         = 0
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 5572
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0000.0'
);
-- ********************************************************************************
-- ** Tablespace name                         = TEMPSPACE1
-- **   Tablespace ID                         = 1
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = System Temporary data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 0
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0001.0'
);
-- ********************************************************************************
-- ** Tablespace name                       = USERSPACE1
-- **   Tablespace ID                         = 2
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 1
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0002.0'
);
-- ********************************************************************************
-- ** Tablespace name                         = DMS
-- **   Tablespace ID                         = 3
-- **   Tablespace Type                       = Database managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Auto-resize enabled                   = No
-- **   Total number of pages                 = 2000
-- **   Number of usable pages                = 1960
-- **   High water mark (pages)               = 96
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   /tmp/dms1                                                    1000
, FILE   /tmp/dms2                                                    1000
);
-- ********************************************************************************
```

```
-- ** Tablespace name                          = RAW
-- **   Tablespace ID                          = 4
-- **   Tablespace Type                        = Database managed space
-- **   Tablespace Content Type                = Any data
-- **   Tablespace Page size (bytes)           = 4096
-- **   Tablespace Extent size (pages)         = 32
-- **   Using automatic storage                = No
-- **   Auto-resize enabled                    = No
-- **   Total number of pages                  = 2000
-- **   Number of usable pages                 = 1960
-- **   High water mark (pages)                = 96
-- ********************************************************************************
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                          1000
, DEVICE '/dev/hdb2'                                          1000
);
-- ********************************************************************************
-- ** start redirect restore
-- ********************************************************************************
   RESTORE DATABASE SAMPLE CONTINUE;
-- ********************************************************************************
-- ** end of file
-- ********************************************************************************
```

10. Eine Scriptausgabe auf einer Datenbank mit dynamischem Speicher für den Befehl

    ```
    restore db test from /home/jseifert/backups taken at 20050304090733 redirect
    generate script TEST_NODE0000.clp
    ```

    würde wie folgt aussehen:

    ```
    -- ********************************************************************************
    -- ** automatically created redirect restore script
    -- ********************************************************************************
    UPDATE COMMAND OPTIONS USING S ON Z ON TEST_NODE0000.out V ON;
    SET CLIENT ATTACH_MEMBER  0;
    SET CLIENT CONNECT_MEMBER 0;
    -- ********************************************************************************
    -- ** initialize redirected restore
    -- ********************************************************************************
    RESTORE DATABASE TEST
    -- USER  '<benutzername>'
    -- USING '<kennwort>'
    FROM '/home/jseifert/backups'
    TAKEN AT 20050304090733
    ON '/home/jseifert'
    -- DBPATH ON <zielverzeichnis>
    INTO TEST
    -- NEWLOGPATH '/home/jseifert/jseifert/TEST/NODE0000/LOGSTREAM0000/'
       -- WITH <pufferanzahl> BUFFERS
       -- BUFFER <puffergröße>
       -- REPLACE HISTORY FILE
       -- REPLACE EXISTING
    REDIRECT
       -- PARALLELISM <n>
       -- WITHOUT ROLLING FORWARD
       -- WITHOUT PROMPTING
    ;
    -- ********************************************************************************
    -- ** storage group definition
    -- **   Default storage group ID               = 0
    -- **   Number of storage groups               = 3
    -- ********************************************************************************
    -- ********************************************************************************
    -- ** Storage group name                       = SG_DEFAULT
    -- **   Storage group ID                       = 0
    -- **   Data tag                               = None
    -- ********************************************************************************
    -- SET STOGROUP PATHS FOR SG_DEFAULT
    ```

```
-- ON '/hdd/path1'
-- ,  '/hdd/path2'
-- ;
-- *****************************************************************************
-- ** Storage group name                        = SG_HOT
-- **   Storage group ID                        = 1
-- **   Data tag                                = 1
-- *****************************************************************************
-- SET STOGROUP PATHS FOR SG_HOT
-- ON '/ssd/fs1'
-- ,  '/ssd/fs2'
-- ;
-- *****************************************************************************
-- ** Storage group name                        = SG_COLD
-- **   Storage group ID                        = 2
-- **   Data tag                                = 9
-- *****************************************************************************
-- SET STOGROUP PATHS FOR SG_COLD
-- ON '/hdd/slowpath1'
-- ;
-- *****************************************************************************
-- ** tablespace definition
-- *****************************************************************************
-- *****************************************************************************
-- ** Tablespace name                           = SYSCATSPACE
-- **   Tablespace ID                           = 0
-- **   Tablespace Type                         = Database managed space
-- **   Tablespace Content Type                 = Any data
-- **   Tablespace Page size (bytes)            = 4096
-- **   Tablespace Extent size (pages)          = 4
-- **   Using automatic storage                 = Yes
-- **   Storage group ID                        = 0
-- **   Source storage group ID                 = -1
-- **   Data tag                                = None
-- **   Auto-resize enabled                     = Yes
-- **   Total number of pages                   = 6144
-- **   Number of usable pages                  = 6140
-- **   High water mark (pages)                 = 5968
-- *****************************************************************************
-- *****************************************************************************
-- ** Tablespace name                           = TEMPSPACE1
-- **   Tablespace ID                           = 1
-- **   Tablespace Type                         = System managed space
-- **   Tablespace Content Type                 = System Temporary data
-- **   Tablespace Page size (bytes)            = 4096
-- **   Tablespace Extent size (pages)          = 32
-- **   Using automatic storage                 = Yes
-- **   Total number of pages                   = 0
-- *****************************************************************************
-- *****************************************************************************
-- ** Tablespace name                         = USERSPACE1
-- **   Tablespace ID                           = 2
-- **   Tablespace Type                         = Database managed space
-- **   Tablespace Content Type                 = Any data
-- **   Tablespace Page size (bytes)            = 4096
-- **   Tablespace Extent size (pages)          = 32
-- **   Using automatic storage                 = Yes
-- **   Storage group ID                        = 1
-- **   Source storage group ID                 = -1
-- **   Data tag                                = 1
-- **   Auto-resize enabled                     = Yes
-- **   Total number of pages                   = 256
-- **   Number of usable pages                  = 224
-- **   High water mark (pages)                 = 96
-- *****************************************************************************
-- *****************************************************************************
-- ** Tablespace name                           = DMS
-- **   Tablespace ID                           = 3
-- **   Tablespace Type                         = Database managed space
-- **   Tablespace Content Type                 = Any data
-- **   Tablespace Page size (bytes)            = 4096
-- **   Tablespace Extent size (pages)          = 32
-- **   Using automatic storage                 = No
-- **   Storage group ID                        = 2
-- **   Source storage group ID                 = -1
```

```
-- **   Data tag                              = 9
-- **   Auto-resize enabled                   = No
-- **   Total number of pages                 = 2000
-- **   Number of usable pages                = 1960
-- **   High water mark (pages)               = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   '/tmp/dms1'                                        1000
, FILE   '/tmp/dms2'                                        1000
);
-- ******************************************************************************
-- ** Tablespace name                       = RAW
-- **   Tablespace ID                         = 4
-- **   Tablespace Type                       = Database managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Auto-resize enabled                   = No
-- **   Total number of pages                 = 2000
-- **   Number of usable pages                = 1960
-- **   High water mark (pages)               = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                        1000
, DEVICE '/dev/hdb2'                                        1000
);
-- ******************************************************************************
-- ** start redirect restore
-- ******************************************************************************
RESTORE DATABASE TEST CONTINUE;
-- ******************************************************************************
-- ** end of file
-- ******************************************************************************
```

11. Es folgen Beispiele für den Befehl **RESTORE DB** mit der Option **SNAPSHOT**:

    Protokollverzeichnisdatenträger aus dem Momentaufnahme-Image ohne Benutzereingabeaufforderung wiederherstellen.

    ```
    db2 restore db sample use snapshot LOGTARGET INCLUDE without prompting
    ```

    Keine Protokollverzeichnisdatenträger wiederherstellen und keine Benutzereingabeaufforderung verwenden.

    ```
    db2 restore db sample use snapshot LOGTARGET EXCLUDE without prompting
    ```

    Keine Protokollverzeichnisdatenträger wiederherstellen und keine Benutzereingabeaufforderung verwenden. Wenn **LOGTARGET** nicht angegeben ist, wird der Standardwert **LOGTARGET EXCLUDE** verwendet.

    ```
    db2 restore db sample use snapshot without prompting
    ```

    Beim Wiederherstellen des Momentaufnahme-Images, das in Konflikt stehende Protokollverzeichnisse enthält, dürfen vorhandene Protokollverzeichnisse in der aktuellen Datenbank ohne Benutzereingabeaufforderung überschrieben und ersetzt werden.

    ```
    db2 restore db sample use snapshot LOGTARGET EXCLUDE FORCE without prompting
    ```

    Beim Wiederherstellen des Momentaufnahme-Images, das in Konflikt stehende Protokollverzeichnisse enthält, dürfen vorhandene Protokollverzeichnisse in der aktuellen Datenbank ohne Benutzereingabeaufforderung überschrieben und ersetzt werden.

    ```
    db2 restore db sample use snapshot LOGTARGET INCLUDE FORCE without prompting
    ```

12. Im Folgenden sind Beispiele einer Transportoperation unter Verwendung des Befehls **RESTORE** mit der Option **TRANSPORT REDIRECT** aufgeführt:

Hierbei wird das Backup-Image einer Quellendatenbank (TT_SRC) mit Speicherpfaden unter /src und eine Zieldatenbank (TT_TGT) mit Speicherpfaden unter /tgt angegeben:

```
> RESTORE DB TT_SRC TABLESPACE (AS1) SCHEMA (KRODGER) TRANSPORT INTO TT_TGT REDIRECT

   SQL1277W Zurzeit wird eine umgeleitete Restoreoperation ausgeführt.
Die Tabellenbereichskonfiguration kann jetzt angezeigt werden, und Container für
Tabellenbereiche, die keinen dynamischen Speicher verwenden, können rekonfiguriert werden.
   DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Der Tabellenbereich 'AS1' wird in einen Containerpfad wie den folgenden transportiert: /tgt/krodger/NODE0000/TT_TGT/T0000003/C0000000.LRG

Um eine Zielspeichergruppe für die transportierten Tabellenbereiche anzugeben, kann die Option **USING STOGROUP** des Befehls **RESTORE** verwendet werden. Im folgenden Beispiel werden die beiden Tabellenbereiche TS1 und TS2 in der Speichergruppe SG_COLD wiederhergestellt:

```
RESTORE DB TT_SRC TABLESPACE (TS1, TS2) SCHEMA (KRODGER) TRANSPORT INTO TT_TGT USING STOGROUP SG_COLD
```

Anmerkung: Die Option **USING STOGROUP** des Befehls **RESTORE** ist nur während einer Transportoperation gültig und kann nicht verwendet werden, um eine Zielspeichergruppe während einer anderen Restoreoperation anzugeben. Um einen Transport in die Standardspeichergruppe der Zieldatenbank auszuführen, muss die Option **USING STOGROUP** nicht angegeben werden:

```
RESTORE  DB TT_SRC TABLESPACE (TS3) SCHEMA (KRODGER) TRANSPORT INTO TT_TGT
```

Der Name der Speichergruppe, der im Befehl **RESTORE** während der Transportoperation (TRANSPORT) angegeben wurde, muss momentan in der Zieldatenbank definiert sein. Er muss nicht im Backup-Image oder der Quellendatenbank definiert sein.

Wenn mit dem Parameter **AT DBPARTITIONNUM** eine Datenbankpartition neu erstellt wird, die (aufgrund einer Beschädigung) gelöscht worden war, befindet sich die Datenbank in dieser Datenbankpartition im Status 'Restore anstehend'. Nach der Neuerstellung der Datenbankpartition muss die Datenbank in dieser Datenbankpartition sofort neu erstellt werden.

Hinweise zur Verwendung

- Ein Befehl **RESTORE DATABASE** im Format db2 restore db *name* führt einen vollständigen Datenbankrestore mit einem Datenbankimage durch und eine Restoreoperation für die Tabellenbereiche, die in einem Tabellenbereichs-Image gefunden werden. Ein Befehl **RESTORE DATABASE** im Format db2 restore db *name* tabellenbereich führt eine Restoreoperation für die Tabellenbereiche durch, die in dem Image gefunden werden. Wird in einem solchen Befehl zusätzlich eine Liste mit Tabellenbereichen angegeben, werden die explizit aufgelisteten Tabellenbereiche wiederhergestellt.
- Im Anschluss an die Restoreoperation eines Online-Backups müssen Sie eine aktualisierende Recovery ausführen.
- Mit dem Parameter **OPTIONS** können Sie Restoreoperationen in TSM-Umgebungen aktivieren, die Proxy-Knoten unterstützen. Weitere Informationen finden Sie im Thema „Konfigurieren eines Tivoli Storage Manager-Clients".
- Wenn ein Backup-Image komprimiert ist, erkennt das DB2-Datenbanksystem die Komprimierung und dekomprimiert die Daten automatisch, bevor die Restoreoperation ausgeführt wird. Wenn in einer db2Restore-API eine Bibliothek angegeben ist, wird sie zum Dekomprimieren der Daten verwendet. Andernfalls wird geprüft, ob in dem Backup-Image eine Bibliothek gespeichert ist. Ist dies der

Fall, wird die Bibliothek verwendet. Wenn in dem Backup-Image keine Bibliothek gespeichert ist, können die Daten nicht dekomprimiert werden, und die Restoreoperation schlägt fehl.

- Soll die Komprimierungsbibliothek aus einem Backup-Image wiederhergestellt werden (entweder explizit durch Angeben der Option **COMPRESSION LIBRARY** oder implizit durch einen regulären Restore aus einem komprimierten Backup), muss die Restoreoperation auf derselben Plattform und unter demselben Betriebssystem erfolgen wie die Erstellung des Backups. Wurde das Backup nicht auf derselben Plattform erstellt, auf der die Restoreoperation erfolgt, schlägt die Restoreoperation fehl, auch wenn das DB2-Datenbanksystem normalerweise plattformübergreifende Restoreoperationen unterstützt, an denen zwei Systeme beteiligt sind.
- Ein gesicherter SMS-Tabellenbereich kann nur in einem SMS-Tabellenbereich wiederhergestellt werden. Er kann nicht in einem DMS-Tabellenbereich wiederhergestellt werden oder umgekehrt.
- Zum Wiederherstellen von Protokolldateien aus dem Backup-Image, in dem sie enthalten sind, muss die Option **LOGTARGET** mit dem vollständig qualifizierten und gültigen Pfad angegeben werden, der auf dem DB2-Server vorhanden ist. Wenn diese Bedingungen erfüllt sind, schreibt das Restoredienstprogramm die Protokolldateien aus dem Image in den Zielpfad. Wird **LOGTARGET** beim Restore eines Backup-Images angegeben, das keine Protokolle enthält, gibt die Restoreoperation einen Fehler zurück, bevor versucht wird, Tabellenbereichsdaten wiederherzustellen. Eine Restoreoperation schlägt ebenfalls mit einer Fehlernachricht fehl, wenn ein ungültiger oder schreibgeschützter Protokollzielpfad (**LOGTARGET**) angegeben ist.
- Sind im Protokollzielpfad (**LOGTARGET**) zu dem Zeitpunkt, an dem der Befehl **RESTORE WDATABASE** abgesetzt wird, bereits Protokolldateien vorhanden, wird eine Warnung an den Benutzer zurückgegeben. Diese Warnung wird nicht zurückgegeben, wenn **WITHOUT PROMPTING** angegeben ist.
- Kann bei einer Restoreoperation, in der **LOGTARGET** angegeben ist, keine Protokolldatei extrahiert werden, schlägt die Restoreoperation fehl und gibt einen Fehler zurück. Trägt eine der aus dem Backup-Image extrahierten Protokolldateien denselben Namen wie eine im Protokollzielpfad (**LOGTARGET**) vorhandene Datei, schlägt die Restoreoperation fehl, und es wird ein Fehler zurückgegeben. Das Datenbankrestoredienstprogramm überschreibt keine vorhandenen Protokolldateien im Protokollzielverzeichnis (**LOGTARGET**).
- Es ist auch möglich, nur die gespeicherten Protokolldateien aus einem Backup-Image wiederherzustellen. Um anzugeben, dass nur die Protokolldateien wiederhergestellt werden sollen, geben Sie zusätzlich zum Protokollzielpfad (**LOGTARGET**) die Option **LOGS** an. Das Angeben der Option **LOGS** ohne einen Protokollzielpfad (**LOGTARGET**) führt zu einem Fehler. Wenn beim Wiederherstellen von Protokolldateien in dieser Betriebsart Probleme auftreten, wird die Restoreoperation sofort abgebrochen und ein Fehler zurückgegeben.
- Bei einer automatischen inkrementellen Restoreoperation werden nur die Protokolldateien aus dem Backup-Image abgerufen, die sich im Zielimage der Restoreoperation befinden. Protokolle, die sich in temporären Images befinden, auf die während des inkrementellen Restores verwiesen wird, werden nicht aus diesen temporären Backup-Images extrahiert. Bei einer manuellen inkrementellen Restoreoperation sollte der Protokollzielpfad (**LOGTARGET**) nur in dem endgültigen Restorebefehl angegeben werden, der abgesetzt werden soll.
- Offline-Backups der gesamten Datenbank sowie inkrementelle Offline-Datenbankbackups können im Gegensatz zu Online-Backups in einer späteren Datenbankversion wiederhergestellt werden. Bei Mehrpartitionsdatenbanken muss die Katalogpartition zuerst einzeln wiederhergestellt werden. Anschließend können

die übrigen Datenbankpartitionen parallel oder seriell wiederhergestellt werden. Das von der Restoreoperation durchgeführte implizite Datenbankupgrade kann jedoch fehlschlagen. Bei einer Mehrpartitionsdatenbank kann sie auf einer oder mehreren Datenbankpartitionen fehlschlagen. In diesem Fall können Sie nach dem Befehl **RESTORE DATABASE** den Befehl **UPGRADE DATABASE** einmalig von der Katalogpartition absetzen, um die Datenbank erfolgreich aufzurüsten.

- In einer Umgebung mit partitionierten Datenbanken kann ein Tabellenbereich eine unterschiedliche Speichergruppenzuordnung in unterschiedlichen Datenbankpartitionen aufweisen. Wenn eine umgeleitete Restoreoperation Tabellenbereichscontainer von DMS in dynamischen Speicher ändert, dann wird dem Tabellenbereich die Standardspeichergruppe zugeordnet. Wenn eine neue Standardspeichergruppe zwischen den umgeleiteten Restoreoperationen unterschiedlicher Datenbankpartitionen ausgewählt wird, dann weist der Tabellenbereich eine inkonsistente Speichergruppenzuordnung innerhalb der Umgebung mit partitionierten Datenbanken auf. Wenn dieser Fall eintritt, dann verwenden Sie die Anweisung ALTER TABLESPACE, um den Tabellenbereich so zu ändern, dass in allen Datenbankpartitionen dynamischer Speicher verwendet wird, und führen Sie bei Bedarf einen Neuausgleich durch.
- Die Option **TRANSPORT** wird nur dann unterstützt, wenn die Codepages von Client und Datenbank gleich sind.

Restoreoperation aus Momentaufnahme

Wie bei einem traditionellen Restore (nicht aus einer Momentaufnahme), werden auch beim Restore eines Momentaufnahme-Backup-Images standardmäßig keine Protokollverzeichnisse wiederhergestellt (**LOGTARGET EXCLUDE**).

Stellt der DB2-Datenbankmanager fest, dass die Gruppen-ID eines Protokollverzeichnisses von anderen wiederherzustellenden Pfaden gemeinsam genutzt wird, wird ein Fehler zurückgegeben. In diesem Fall muss **LOGTARGET INCLUDE** oder **LOGTARGET INCLUDE FORCE** angegeben werden, da die Protokollverzeichnisse Teil der Restoreoperation sein müssen.

Der DB2-Datenbankmanager versucht mit allen Mitteln, vorhandene Protokollverzeichnisse (Primär-, Spiegel- und Überlaufprotokolle) zu sichern, bevor die Restoreoperation der Pfade aus dem Backup-Image ausgeführt wird.

Sollen die Protokollverzeichnisse wiederhergestellt werden und erkennt der DB2-Datenbankmanager, dass die vorhandenen Protokollverzeichnisse auf dem Datenträger in Konflikt mit den Protokollverzeichnissen im Backup-Image stehen, meldet der DB2-Datenbankmanager einen Fehler. Wenn Sie in einem solchen Fall **LOGTARGET INCLUDE FORCE** angegeben haben, wird dieser Fehler unterdrückt, und die Protokollverzeichnisse aus dem Image werden wiederhergestellt (dabei werden bereits vorhandene Protokolle gelöscht).

Es gibt einen Sonderfall, in dem die Option **LOGTARGET EXCLUDE** angegeben ist und sich ein Protokollverzeichnispfad im Datenbankverzeichnis (beispielsweise /NODE*xxxx*/SQL*xxxxx*/LOGSTREAM*xxxxx*/) befindet. In diesem Fall würde das Protokollverzeichnis bei der Restoreoperation dennoch überschrieben, während der Datenbankpfad mit allen untergeordneten Elementen wiederhergestellt würde. Wenn der DB2-Datenbankmanager dieses Szenario feststellt und dieses Protokollverzeichnis Protokolldateien enthält, wird ein Fehler gemeldet. Wenn Sie **LOGTARGET EXCLUDE FORCE** angegeben

haben, wird dieser Fehler unterdrückt, und die Protokollverzeichnisse aus dem Backup-Image überschreiben die in Konflikt stehenden Protokollverzeichnisse auf dem Datenträger.

Transport von Tabellenbereichen und Schemata

Es muss die vollständige Liste der Tabellenbereiche und Schemata angegeben werden.

Die Zieldatenbank muss zum Zeitpunkt des Transports aktiv sein.

Wenn ein Online-Backup-Image verwendet wird, wird die Bereitstellungsdatenbank zum Endpunkt des Backups aktualisierend wiederhergestellt. Wird ein Offline-Backup-Image verwendet, findet keine aktualisierende Wiederherstellung durchgeführt.

Eine Bereitstellungsdatenbank, die aus dem Systemkatalog-Tabellenbereich des Backup-Images besteht, wird unter dem durch den Datenbankparameter **dftdbpath** angegebenen Pfad erstellt. Diese Datenbank wird nach Abschluss des Befehls **RESTORE DATABASE** gelöscht. Die Bereitstellungsdatenbank wird benötigt, um die DDL (Data Definition Language) zu extrahieren, die zur erneuten Generierung der Objekte in der transportierten Tabelle verwendet wird.

Beim Transport von Tabellenbereichen versucht der DB2-Datenbankmanager, den ersten verfügbaren Pufferpool mit der übereinstimmenden Seitengröße dem Tabellenbereich zuzuweisen, der transportiert wird. Wenn die Zieldatenbank über keine Pufferpools mit entsprechender Seitengröße für transportierte Tabellenbereiche verfügt, wird möglicherweise ein verdeckter Pufferpool zugewiesen. Verdeckte Pufferpools sind temporäre Platzhalter für transportierte Tabellenbereiche. Sie können die den transportierten Tabellenbereichen zugewiesenen Pufferpools nach Abschluss des Transports prüfen. Zum Aktualisieren der Pufferpools können Sie den Befehl **ALTER TABLESPACE** absetzen.

Wenn die aktualisierende Recovery einer Datenbank einen Protokolleintrag für ein transportiertes Tabellenbereichsschema findet, wird der entsprechende Tabellenbereich offline gesetzt und in den Status "Löschen anstehend" versetzt. Der Grund dafür ist, dass die Datenbank nicht über vollständige Protokolle zu transportierten Tabellenbereichen verfügt, um diese sowie deren Inhalte erneut zu erstellen. Sie können nach Beendigung des Transports einen vollständiges Backup der Zieldatenbank durchführen, damit bei der nachfolgenden aktualisierenden Recoveryoperation der Punkt des Schematransports im Protokolldatenstrom nicht übergangen wird.

Transport von Speichergruppen

Eine Transportoperation kann die momentan definierten Speichergruppen der Zieldatenbank nicht ändern und neue Speichergruppen können während eines Transports nicht explizit erstellt werden.

Die Standardzielspeichergruppe des Transports ist die Standardspeichergruppe der Zieldatenbank der Operation. Es ist außerdem möglich, alle Tabellenbereiche, die während einer Transportoperation in eine bestimmte Speichergruppe der Zieldatenbank wiederhergestellt werden, explizit umzuleiten.

Wenn während einer Transportoperation der Befehl **RESTORE** mit der Option **TRANSPORT REDIRECT** abgesetzt wird, dann stimmt die Standardspeichergruppenkonfiguration für Tabellenbereiche mit dynamischem Speicher nicht mit der Konfiguration überein, die im Backup-Image enthalten ist.

Stattdessen stimmt sie mit den Speichergruppen und Speichergruppenpfaden der Zieldatenbank überein. Dies ist darauf zurückzuführen, dass Tabellenbereiche mit dynamischem Speicher wiederhergestellt und (gemäß der Definition in der Zieldatenbank) direkt in die vorhandenen Speichergruppenpfade umgeleitet werden müssen.

Hohe Verfügbarkeit durch Unterstützung der ausgesetzten E/A und der Onlineteilung einer Spiegeldatenbank

Durch die Unterstützung der ausgesetzten E/A des IBM DB2-Servers können Sie gespiegelte Kopien Ihrer Primärdatenbank teilen, ohne die Datenbank in den Offlinestatus versetzen zu müssen. Sie können diese Funktion verwenden, um rasch eine Bereitschaftsdatenbank zu erstellen, die die Arbeit übernimmt, wenn die Primärdatenbank fehlschlägt.

Die Plattenspiegelung ist ein Vorgang, bei dem Daten gleichzeitig auf zwei unterschiedliche Festplatten geschrieben werden. Die eine Kopie der Daten wird als der "Spiegel" der anderen Kopie bezeichnet. Das Teilen einer solchen Spiegelung bedeutet, dass die beiden Kopien voneinander getrennt werden.

Sie können mit der Plattenspiegelung eine Kopie Ihrer Primärdatenbank verwalten. Sie können die DB2-Server-Funktionalität der ausgesetzten E/A dazu verwenden, die primären und sekundären gespiegelten Kopien der Datenbank zu teilen, ohne die Datenbank in den Offlinestatus versetzen zu müssen. Sobald die primären und sekundären Kopien der Datenbank geteilt sind, kann die sekundäre Datenbank die Operationen übernehmen, falls die Primärdatenbank ausfällt.

Wenn Sie eine große Datenbank lieber nicht mit dem Backup-Dienstprogramm des DB2-Servers sichern möchten, können Sie Kopien von einem Spiegelimage erstellen, indem Sie dazu die ausgesetzte E/A und die Funktion zur Erstellung einer Spiegeldatenbank verwenden. Außerdem erreichen Sie mit dieser Methode Folgendes:

- Sie vermeiden hohen Backup-Aufwand auf der Produktionsmaschine.
- Sie verfügen über eine schnelle Methode, Systeme zu klonen.
- Sie verfügen über eine schnelle Implementierung der Funktionsübernahme aus dem Bereitschaftsmodus. Ein einleitender Restore findet nicht statt, und sollte eine aktualisierende Recovery sich als zu langsam erweisen oder sollten Fehler auftreten, ist die erneute Initialisierung sehr schnell.

Mit dem Befehl **db2inidb** wird die geteilte Spiegeldatenbank initialisiert, sodass Sie sie wie folgt einsetzen können:

- Als Klondatenbank
- Als Bereitschaftsdatenbank
- Als Backup-Image

Dieser Befehl kann nur für eine geteilte Spiegeldatenbank abgesetzt werden, und die geteilte Spiegeldatenbank kann erst nach Ausführung des Befehls verwendet werden.

In einer Umgebung mit partitionierten Datenbanken müssen Sie E/A-Schreibvorgänge nicht auf allen Datenbankpartitionen gleichzeitig aussetzen. Sie können die E/A für eine Untergruppe aus einer oder mehreren Datenbankpartitionen aussetzen, um zum Ausführen von Offline-Backups geteilte Spiegeldatenbanken zu erstellen. Wenn in dieser Untergruppe die Katalogpartition enthalten ist, muss das Aussetzen für diese Datenbankpartition zuletzt erfolgen.

In einer Umgebung mit partitionierten Datenbanken müssen Sie den Befehl **db2inidb** auf jeder Datenbankpartition ausführen, bevor Sie das geteilte Image irgendeiner der Datenbankpartitionen verwenden können. Sie können das Tool in allen Datenbankpartitionen gleichzeitig ausführen, indem Sie den Befehl **db2_all** verwenden. Wenn Sie jedoch die Option RELOCATE USING verwenden, können Sie den Befehl **db2inidb** nicht mit dem Befehl **db2_all** auf allen Datenbankpartitionen gleichzeitig ausführen. Für jede Datenbankpartition muss eine eigene Konfigurationsdatei angegeben werden, die den Wert für NODENUM der zu ändernden Datenbankpartition enthält. Wenn beispielsweise der Name der Datenbank geändert wird, sind alle Datenbankpartitionen von dieser Änderung betroffen, und der Befehl **db2relocatedb** muss auf jeder Datenbankpartition mit einer eigenen Konfigurationsdatei ausgeführt werden. Wenn Container versetzt werden, die zu einer einzelnen Datenbankpartition gehören, muss der Befehl **db2relocatedb** nur einmal auf dieser Datenbankpartition ausgeführt werden.

Anmerkung: Stellen Sie sicher, dass die geteilte Spiegeldatenbank alle Container und Verzeichnisse enthält, aus denen die Datenbank besteht, einschließlich des Datenträgerverzeichnisses. Eine Zusammenstellung dieser Informationen finden Sie in der Verwaltungssicht DBPATHS, in der alle Dateien und Verzeichnisse der Datenbank angezeigt werden, die geteilt werden müssen.

db2inidb - Initialisieren einer Spiegeldatenbank

Initialisiert eine Spiegeldatenbank in einer Umgebung mit geteilten Spiegeldatenbanken. Die Spiegeldatenbank kann als Klon der Primärdatenbank initialisiert, in den Status "Aktualisierende Recovery anstehend" versetzt oder als Backup-Image zum Wiederherstellen der Primärdatenbank verwendet werden.

Sie müssen diesen Befehl eingeben, um eine geteilte Spiegeldatenbank verwenden zu können.

Berechtigung

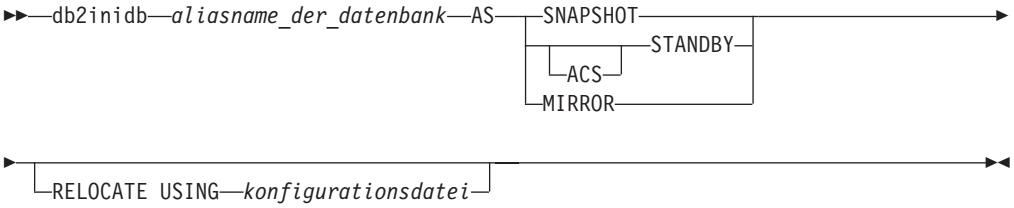
Eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL
- SYSMAINT

Erforderliche Verbindung

Keine

Befehlssyntax



Befehlsparameter

aliasname_der_datenbank
Gibt den Aliasnamen der zu initialisierenden Datenbank an.

SNAPSHOT
Gibt an, dass die Spiegeldatenbank als Klon der Primärdatenbank initialisiert werden soll.

STANDBY
Gibt an, dass die Datenbank in den Status "Aktualisierende Recovery anstehend" versetzt wird. Neue Protokolle aus der Primärdatenbank können abgerufen und auf die Bereitschaftsdatenbank angewendet werden. Anschließend kann die Bereitschaftsdatenbank anstelle der Primärdatenbank verwendet werden, falls diese ausfällt.

ACS
Gibt an, dass der Befehl **db2inidb** für eine ACS-Momentaufnahmekopie der Datenbank verwendet werden muss, um die Aktion **STANDBY** durchzuführen. Diese Option ist erforderlich, da der Befehl **db2inidb** nur für Momentaufnahmen von geteilten Spiegeldatenbanken abgesetzt werden kann, die mit dem Befehl **SET WRITE SUSPEND | RESUME** erstellt wurden.

Die Verwendung der Option **ACS STANDBY** sorgt dafür, dass die ACS-Momentaufnahmenkopie in den Status 'Aktualisierende Recovery anstehend' versetzt wird, sodass der DB2-Befehl **BACKUP** erfolgreich für das Momentaufnahme-Image abgesetzt werden kann. Ohne diese Option wird bei jedem Versuch, eine Verbindung zum Momentaufnahme-Image herzustellen, diese Kopie in den Status 'Restore anstehend' versetzt, wodurch sie bei zukünftigen Recoveryoperationen nicht mehr als Backup-Kopie verwendet werden kann.

Diese Funktion wurde speziell für das Einrichten von Schnittstellen mit Speichermanagern wie IBM Tivoli Storage FlashCopy Manager mit dem Zweck eingeführt, ein ausgelagertes DB2-Backup zu erstellen, das auf einer ACS-Momentaufnahme basiert. Die Verwendung dieser Option zu anderen Zwecken, beispielsweise zum Bereitstellen oder Ändern des Inhalts einer ACS-Momentaufnahmekopie oder zur Generierung eines DB2-Backups, kann zu einem späteren Zeitpunkt zu einem nicht definierten Verhalten führen.

MIRROR Gibt an, dass die Spiegeldatenbank als Backup-Image zum Wiederherstellen der Primärdatenbank verwendet werden soll.

RELOCATE USING *konfigurationsdatei*
Gibt an, dass die Datenbankdateien anhand der Informationen aus der angegebenen *konfigurationsdatei* verlagert werden, bevor die Datenbank als Momentaufnahme-, Bereitschafts- oder Spiegeldatenbank initialisiert wird. Das Format für die *konfigurationsdatei* ist in „db2relocatedb - Verlagern einer Datenbank" auf Seite 206 beschrieben.

Hinweise zur Verwendung

Führen Sie die Operation `db2 connect to`*`aliasname_der_datenbank`* nicht vor dem Absetzen des Befehls `db2inidb`*`aliasname_der_datenbank`*`as mirror` aus. Bei dem Versuch, eine Verbindung zu einer geteilten Spiegeldatenbank herzustellen, bevor diese initialisiert wurde, werden die für die aktualisierende Recovery benötigten Protokolldateien gelöscht. Beim Verbindungsaufbau wird Ihre Datenbank wieder in den Status versetzt, den sie hatte, als sie ausgesetzt wurde. Wurde die Datenbank bei ihrer Aussetzung als konsistent markiert, geht das DB2-Datenbanksystem davon aus, dass keine Recovery nach Systemabsturz erforderlich ist, und löscht den Inhalt der Protokolldateien, damit sie später wieder verwendet werden können.

Wenn der Inhalt der Protokolldateien gelöscht wurde, wird bei dem Versuch einer aktualisierenden Recovery die Fehlernachricht SQL4970N zurückgegeben.

In Umgebungen mit partitionierten Datenbanken müssen Sie den Befehl **db2inidb** auf jeder Datenbankpartition ausführen, bevor Sie das Image der geteilten Spiegeldatenbank einer der Datenbankpartitionen verwenden können. **db2inidb** kann auf allen Datenbankpartitionen gleichzeitig ausgeführt werden, wenn der Befehl **db2_all** verwendet wird.

Wenn Sie jedoch die Option **RELOCATE USING** verwenden, können Sie den Befehl **db2_all** nicht verwenden, um **db2inidb** auf allen Partitionen gleichzeitig auszuführen. Für jede Partition muss eine eigene Konfigurationsdatei angegeben werden, die den Wert NODENUM der zu ändernden Datenbankpartition enthält. Wenn beispielsweise der Name der Datenbank geändert wird, sind alle Datenbankpartitionen von dieser Änderung betroffen, und der Befehl **db2relocatedb** muss auf jeder Datenbankpartition mit einer eigenen Konfigurationsdatei ausgeführt werden. Wenn Container versetzt werden, die zu einer einzelnen Datenbankpartition gehören, muss der Befehl **db2relocatedb** nur einmal auf dieser Datenbankpartition ausgeführt werden.

Wird der Parameter **RELOCATE USING** *konfigurationsdatei* angegeben und die Datenbank erfolgreich verlagert, dann wird die angegebene *konfigurationsdatei* in das Datenbankverzeichnis kopiert und in `db2path.cfg` umbenannt. Bei einer späteren Recovery nach Systemabsturz oder aktualisierenden Recovery wird diese Datei während der Verarbeitung von Protolldateien verwendet, um Containerpfade umzubenennen.

Beim Initialisieren einer Klondatenbank wird die angegebene *konfigurationsdatei* nach Beendigung einer Recovery nach Systemabsturz automatisch aus dem Datenbankverzeichnis entfernt.

Beim Initialisieren einer Bereitschafts- oder Spiegeldatenbank wird die angegebene *konfigurationsdatei* automatisch aus dem Datenbankverzeichnis entfernt, nachdem eine aktualisierende Recovery beendet oder abgebrochen wurde. Neue Containerpfade können zur Datei `db2path.cfg` hinzugefügt werden, nachdem der Befehl **db2inidb** ausgeführt wurde. Dies wird erforderlich, wenn CREATE- oder ALTER TABLESPACE-Operationen an der ursprünglichen Datenbank ausgeführt werden und in der Bereitschaftsdatenbank andere Pfade verwendet werden müssen.

Verwenden Sie beim Durchführen der Initialisierung einer geteilten Spiegeldatenbank aus einer HADR-Primär- oder Bereitschaftsdatenbank den Parameter **STANDBY**, wenn eine der folgenden Situationen vorliegt:

- Die neue Datenbank wird in einem HADR-Paar verwendet, und die HADR-Konfigurationseinstellungen des neues Paares stimmen nicht mit den Einstellungen des ursprünglichen Paares überein.
- Die Datenbank wird als eigenständige Datenbank initialisiert.

In DB2 pureScale-Umgebungen können Sie den Befehl **db2inidb** über jedes Member absetzen. Der Befehl muss nur einmal abgesetzt werden.

db2relocatedb - Verlagern einer Datenbank

Dieser Befehl benennt eine Datenbank um oder verlagert eine Datenbank oder einen Teil einer Datenbank (z. B. den Container und das Protokollverzeichnis), wie

in der durch den Benutzer bereitgestellten Konfigurationsdatei angegeben ist. Dieses Tool nimmt die erforderlichen Änderungen an den Unterstützungsdateien der DB2-Instanz und der Datenbank vor.

Die Zieldatenbank muss offline sein, damit der Befehl **db2relocatedb** zum Ändern der Steuerdateien und Metadaten der Zieldatenbank ausgeführt werden kann.

Die mit dem Befehl **db2relocatedb** an den Dateien und Steuerstrukturen einer Datenbank vorgenommenen Änderungen werden nicht protokolliert und sind daher nicht wiederherstellbar. Es empfiehlt sich, nach der Ausführung des Befehls für eine Datenbank ein Gesamtbackup auszuführen, insbesondere wenn die Datenbank wiederherstellbar ist und Protokolldateien beibehalten werden.

Berechtigung

Keine

Voraussetzung

Vor der Verwendung dieses Befehls müssen die Daten mithilfe des folgenden Befehls versetzt werden:

```
mv /home/db2inst1/db2inst1/NODE0000/X /home/db2inst1/db2inst1/NODE0000/Y
```

Hierbei gilt: *X* ist der alte Datenbankname, *Y* der der neue Datenbankname.

Dieser zusätzliche Schritt ist erforderlich, um sicherzustellen, dass der Befehl **db2relocatedb** ohne Fehlernachrichten ausgeführt wird. Dieser Schritt ist nur für DB2 ab Version 10.1 erforderlich.

Befehlssyntax

Befehlsparameter

-f *konfigurationsdateiname*

Gibt den Namen der Datei mit den Konfigurationsdaten an, die zum Verlagern der Datenbank benötigt werden. Dies kann ein relativer oder ein absoluter Dateiname sein. Die Konfigurationsdatei hat folgendes Format:

```
DB_NAME=alterName,neuerName
   DB_PATH=alterPfad,neuerPfad
   INSTANCE=alteInstanz,neueInstanz
   NODENUM=knotenNummer
   LOG_DIR=alterVerzPfad,neuerVerzPfad
   CONT_PATH=alterBehältPfad1,neuerBehältPfad1
   CONT_PATH=alterBehältPfad2,neuerBehältPfad2
...
   STORAGE_PATH=alterSpeicherpfad1,neuerSpeicherpfad1
   STORAGE_PATH=alterSpeicherpfad2,neuerSpeicherpfad2
...
FAILARCHIVE_PATH=neuerVerzPfad
LOGARCHMETH1=neuerVerzPfad
LOGARCHMETH2=neuerVerzPfad
MIRRORLOG_PATH=neuerVerzPfad
OVERFLOWLOG_PATH=neuerVerzPfad
...
```

Dabei gilt Folgendes:

DB_NAME
Gibt den Namen der Datenbank an, die verlagert wird. Wenn der Name der Datenbank geändert wird, müssen der alte und der neue Name angegeben werden. Dies ist ein erforderliches Feld.

DB_PATH
Gibt den ursprünglichen Pfad der Datenbank an, die verlagert wird. Wenn der Datenbankpfad geändert wird, müssen der alte und der neue Pfad angegeben werden. Dies ist ein erforderliches Feld.

INSTANCE
Gibt die Instanz an, in dem die Datenbank vorhanden ist. Wenn die Datenbank in eine neue Instanz versetzt wird, müssen die alte und die neue Instanz angegeben werden. Dies ist ein erforderliches Feld.

NODENUM
Gibt die Knotennummer für den zu ändernden Datenbankknoten an. Der Standardwert ist 0.

LOG_DIR
Gibt eine Änderung der Speicherposition des Protokollpfads an. Wenn der Protokollpfad geändert wird, müssen sowohl der alte als auch der neue Pfad angegeben werden. Diese Spezifikation ist optional, wenn sich der Protokollpfad unter dem Datenbankpfad befindet. In diesem Fall wird der Pfad automatisch aktualisiert.

CONT_PATH
Gibt eine Änderung der Speicherposition der Tabellenbereichscontainer an. Der alte und der neue Containerpfad müssen angegeben werden. Mehrere **CONT_PATH**-Zeilen können angegeben werden, wenn mehrere Containerpfadänderungen erforderlich sind. Diese Spezifikation ist optional, wenn sich der Containerpfad unter dem Datenbankpfad befindet. In diesem Fall werden die Pfade automatisch aktualisiert. Wenn Sie Änderungen an mehr als einem Container vornehmen, wobei derselbe alte Pfad durch einen gemeinsamen neuen Pfad ersetzt wird, können Sie einen einzigen Eintrag **CONT_PATH** verwenden. In diesem Fall kann ein Stern (*) als Platzhalterzeichen für den alten und neuen Pfad verwendet werden.

STORAGE_PATH
Gibt eine Änderung an der Speicherposition eines der Speicherpfade für die Datenbank an. Sowohl der alte als auch der neue Speicherpfad müssen angegeben werden. Mehrere **STORAGE_PATH**-Zeilen können angegeben werden, wenn mehrere Speicherpfadänderungen erforderlich sind.Sie können diesen Parameter angeben, um einen Speicherpfad in allen Speichergruppen zu ändern. Allerdings ist es nicht möglich, diesen Parameter zum Ändern der Speicherpfade für eine einzelne Speichergruppe anzugeben.

Anmerkung: Dieser Parameter kann nicht für eine Datenbank verwendet werden, die mit der Klausel AUTOMATIC STORAGE NO erstellt wurde. Eine Datenbank kann mit der Klausel AUTOMATIC STORAGE NO erstellt werden; die Klausel AUTOMATIC STORAGE ist jedoch veraltet und wird in zukünftigen Releases möglicherweise nicht mehr enthalten sein.

FAILARCHIVE_PATH
Gibt eine neue Speicherposition zum Archivieren der Protokolldateien an, wenn der Datenbankmanager die Protokolldateien weder an der primären noch an der sekundären Archivposition archivieren kann. Geben Sie dieses Feld nur an, wenn in der zu verlagernden Datenbank der Konfigurationsparameter **failarchpath** definiert ist.

LOGARCHMETH1
Gibt eine neue primäre Archivposition an. Geben Sie dieses Feld nur an, wenn in der zu verlagernden Datenbank der Konfigurationsparameter **logarchmeth1** definiert ist.

LOGARCHMETH2
Gibt eine neue sekundäre Archivposition an. Geben Sie dieses Feld nur an, wenn in der zu verlagernden Datenbank der Konfigurationsparameter **logarchmeth2** definiert ist.

MIRRORLOG_PATH
Gibt eine neue Speicherposition für den Spiegelprotokollpfad an. Diese Zeichenfolge muss auf einen Pfadnamen verweisen, der ein vollständig qualifizierter Pfadname und kein relativer Pfadname ist. Geben Sie dieses Feld nur an, wenn in der zu verlagernden Datenbank der Konfigurationsparameter **mirrorlogpath** definiert ist.

OVERFLOWLOG_PATH
Gibt eine neue Speicherposition an, an der die Protokolldateien gesucht werden sollen, die für eine ROLLFORWARD-Operation benötigt werden, an der die aktiven Protokolldateien gespeichert werden sollen, die aus dem Archiv abgerufen werden, und an der die Protokolldateien gesucht und gespeichert werden sollen, die für die API db2ReadLog benötigt werden. Geben Sie dieses Feld nur an, wenn in der zu verlagernden Datenbank der Konfigurationsparameter **overflowlogpath** definiert ist.

Leere Zeilen oder Zeilen, die mit einem Kommentarzeichen (#) beginnen, werden ignoriert.

Beispiele

Beispiel 1

Erstellen Sie die folgende Konfigurationsdatei, um den Namen der Datenbank TESTDB in der Instanz 'db2inst1', die sich im Pfad /home/db2inst1 befindet, in PRODDB zu ändern:

```
DB_NAME=TESTDB,PRODDB
DB_PATH=/home/db2inst1
INSTANCE=db2inst1
NODENUM=0
```

Nach dem Erstellen der Konfigurationsdatei müssen Sie alle Pfadangaben für den dynamischen Speicher so ändern, dass sie mit dem neuen Datenbanknamen übereinstimmen.

```
rename /home/db2inst1/db2inst1/TESTDB /home/db2inst1/db2inst1/PRODDB
```

Speichern Sie die Konfigurationsdatei als relocate.cfg, und verwenden Sie den folgenden Befehl, um die Änderungen an den Datenbankdateien vorzunehmen:

```
db2relocatedb -f relocate.cfg
```

Beispiel 2

Gehen Sie wie folgt vor, um die Datenbank DATAB1 aus der Instanz 'jsmith' im Pfad /dbpath in die Instanz 'prodinst' zu versetzen:

1. Versetzen Sie die Dateien im Verzeichnis /dbpath/jsmith in das Verzeichnis /dbpath/prodinst.
2. Verwenden Sie die folgende Konfigurationsdatei mit dem Befehl **db2relocatedb**, um die Änderungen an den Datenbankdateien vorzunehmen:

```
DB_NAME=DATAB1
DB_PATH=/dbpath
INSTANCE=jsmith,prodinst
NODENUM=0
```

Beispiel 3

Die Datenbank PRODDB ist in der Instanz 'inst1' im Pfad /databases/PRODDB vorhanden. Die Speicherposition von zwei Tabellenbereichscontainern muss wie folgt geändert werden:

- Der SMS-Container /data/SMS1 muss nach /DATA/NewSMS1 versetzt werden.
- Der DMS-Container /data/DMS1 muss nach /DATA/DMS1 versetzt werden.

Nachdem die physischen Verzeichnisse und Dateien an die neuen Speicherpositionen versetzt wurden, kann die folgende Konfigurationsdatei mit dem Befehl **db2relocatedb** verwendet werden, um die Änderungen an den Datenbankdateien vorzunehmen, damit sie die neuen Speicherpositionen erkennen:

```
DB_NAME=PRODDB
   DB_PATH=/databases/PRODDB
   INSTANCE=inst1
   NODENUM=0     CONT_PATH=/data/SMS1,/DATA/NewSMS1
   CONT_PATH=/data/DMS1,/DATA/DMS1
```

Beispiel 4

Die Datenbank TESTDB ist in der Instanz 'db2inst1' vorhanden und wurde im Pfad /databases/TESTDB erstellt. Danach wurden Tabellenbereiche mit den folgenden Containern erstellt:

```
TS1
   TS2_Cont0
   TS2_Cont1
   /databases/TESTDB/TS3_Cont0
   /databases/TESTDB/TS4/Cont0
   /Data/TS5_Cont0
   /dev/rTS5_Cont1
```

TESTDB soll auf ein neues System versetzt werden. Die Instanz auf dem neuen System ist 'newinst', und die Speicherposition der Datenbank ist /DB2.

Beim Versetzen der Datenbank müssen alle Dateien, die im Verzeichnis /databases/TESTDB/db2inst1 vorhanden sind, in das Verzeichnis /DB2/newinst versetzt werden. Das bedeutet, dass die ersten fünf Container als Teil des Versetzens verlagert werden. (Die ersten drei Container werden relativ zum Datenbankverzeichnis angegeben und die anderen beiden relativ zum Datenbankpfad.) Weil sich diese Container im Datenbankverzeichnis bzw. im Datenbankpfad befinden, müssen sie nicht in der Konfigurationsdatei aufgelistet werden. Wenn die restlichen beiden Container an andere Speicherpositionen auf dem neuen System versetzt werden sollen, müssen sie in der Konfigurationsdatei aufgelistet werden.

Nachdem die physischen Verzeichnisse und Dateien an ihre neuen Speicherpositionen versetzt wurden, kann die folgende Konfigurationsdatei mit dem Befehl **db2relocatedb** verwendet werden, um Änderungen an den Datenbankdateien vorzunehmen, damit sie die neuen Speicherpositionen erkennen:

```
DB_NAME=TESTDB
   DB_PATH=/databases/TESTDB,/DB2
   INSTANCE=db2inst1,newinst
   NODENUM=0     CONT_PATH=/Data/TS5_Cont0,/DB2/TESTDB/TS5_Cont0
   CONT_PATH=/dev/rTS5_Cont1,/dev/rTESTDB_TS5_Cont1
```

Beispiel 5

Die Datenbank TESTDB hat zwei Datenbankpartitionen auf den Datenbankpartitionsservern 10 und 20. Die Instanz ist 'servinst', und der Datenbankpfad auf beiden Datenbankpartitionsservern ist `/home/servinst`. Der Name der Datenbank wird in SERVDB und der Datenbankpfad auf beiden Datenbankpartitionsservern in `/databases` geändert. Außerdem wird das Protokollverzeichnis auf dem Datenbankpartitionsserver 20 von `/testdb_logdir` in `/servdb_logdir` geändert.

Weil an beiden Datenbankpartitionen Änderungen vorgenommen werden, muss für jede Datenbankpartition eine Konfigurationsdatei erstellt werden. Der Befehl **db2relocatedb** muss auf jedem Datenbankpartitionsserver mit der entsprechenden Konfigurationsdatei ausgeführt werden.

Auf Datenbankpartitionsserver 10 wird die folgende Konfigurationsdatei verwendet:

```
DB_NAME=TESTDB,SERVDB
   DB_PATH=/home/servinst,/databases
   INSTANCE=servinst
NODENUM=10
```

Auf Datenbankpartitionsserver 20 wird die folgende Konfigurationsdatei verwendet:

```
DB_NAME=TESTDB,SERVDB
   DB_PATH=/home/servinst,/databases
   INSTANCE=servinst
NODENUM=20
   LOG_DIR=/testdb_logdir,/servdb_logdir
```

Beispiel 6

Die Datenbank MAINDB ist in der Instanz `maininst` im Pfad `/home/maininst` vorhanden. Die Speicherposition von vier Tabellenbereichscontainern muss wie folgt geändert werden:

```
/maininst_files/allconts/C0 muss nach /MAINDB/C0 versetzt werden
/maininst_files/allconts/C1 muss nach /MAINDB/C1 versetzt werden
/maininst_files/allconts/C2 muss nach /MAINDB/C2 versetzt werden
/maininst_files/allconts/C3 muss nach /MAINDB/C3 versetzt werden
```

Nachdem die physischen Verzeichnisse und Dateien an die neuen Speicherpositionen versetzt wurden, kann die folgende Konfigurationsdatei mit dem Befehl **db2relocatedb** verwendet werden, um die Änderungen an den Datenbankdateien vorzunehmen, damit sie die neuen Speicherpositionen erkennen.

Eine ähnliche Änderung wird in allen Containern vorgenommen, d. h. `/maininst_files/allconts/` wird durch `/MAINDB/` ersetzt, sodass ein einziger Eintrag mit dem Platzhalterzeichen verwendet werden kann:

```
DB_NAME=MAINDB
   DB_PATH=/home/maininst
   INSTANCE=maininst
   NODENUM=0     CONT_PATH=/maininst_files/allconts/*, /MAINDB/*
```

Hinweise zur Verwendung

Wenn eine Instanz geändert wird, zu der eine Datenbank gehört, müssen Sie die folgenden Schritte ausführen, bevor Sie diesen Befehl ausführen, um sicherzustellen, dass Änderungen an der Instanz und an den Datenbankunterstützungsdateien vorgenommen werden:

- Wenn eine Datenbank in eine andere Instanz versetzt wird, erstellen Sie die neue Instanz. Die neue Instanz muss dasselbe Release-Level aufweisen wie die Instanz, zu der die Datenbank gehört.
- Wenn die neue Instanz einen anderen Eigentümer als die aktuelle Instanz aufweist, erteilen Sie dem neuen Instanzeigner die entsprechenden Zugriffsrechte.
- Kopieren Sie die Dateien und Einheiten der Datenbank, die kopiert wird, auf das System, auf dem sich die neue Instanz befindet. Falls erforderlich müssen die Pfadnamen angepasst werden. Sind jedoch bereits Datenbanken in dem Verzeichnis vorhanden, in das die Datenbankdateien versetzt werden, kann es vorkommen, dass die bestehende Datei `sqldbdir` versehentlich überschrieben wird und dadurch die Verweise auf die bestehenden Datenbanken entfernt werden. In diesem Szenario kann das Dienstprogramm **db2relocatedb** nicht verwendet werden. Eine umgeleitete Restore-Operation ist eine Alternative zu **db2relocatedb**.
- Ändern Sie die Berechtigungen der kopierten Dateien/Einheiten, damit der Instanzeigner Eigner dieser Dateien/Einheiten wird.

Wenn Sie eine Datenbank aus einem Datenbankpfad versetzen, in dem sich mehrere Datenbanken befinden, müssen Sie das Verzeichnis `sqldbdir` in diesem Datenbankpfad kopieren, nicht versetzen. Dieses Verzeichnis wird von DB2 weiterhin an der alten Position benötigt, um die nicht versetzten Datenbanken zu finden. Nach dem Kopieren des Verzeichnisses `sqldbdir` an die neue Speicherposition werden die nicht versetzten Datenbanken mit dem Befehl `LIST DB DIRECTORY ON` *neuerPfad* aufgelistet. Diese Verweise können nicht entfernt werden und es können keine neuen Datenbanken mit solchen Namen in demselben Pfad erstellt werden. In einem anderen Pfad können jedoch Datenbanken mit einem solchen Namen erstellt werden.

Mit dem Befehl **db2relocatedb** können keine vorhandene, von Benutzern erstellte Container für einen Tabellenbereich versetzt werden, der mit der Anweisung ALTER TABLESPACE MANAGED BY AUTOMATIC STORAGE für die Verwendung von dynamischem Speicher konvertiert wurde.

Wenn die Instanz geändert wird, muss der Befehl vom neuen Instanzeigner ausgeführt werden.

In einer Umgebung mit partitionierten Datenbanken muss dieses Tool für alle Datenbankpartitionen ausgeführt werden, auf denen Änderungen erforderlich sind. Für jede Datenbankpartition muss eine eigene Konfigurationsdatei angegeben werden, die den Wert `NODENUM` der zu ändernden Datenbankpartition enthält. Wenn beispielsweise der Name der Datenbank geändert wird, sind alle Datenbankpartitionen von dieser Änderung betroffen, und der Befehl **db2relocatedb** muss auf jeder Datenbankpartition mit einer eigenen Konfigurationsdatei ausgeführt werden. Wenn Container versetzt werden, die zu einer einzelnen Datenbankpartition gehören, muss der Befehl **db2relocatedb** nur einmal auf dieser Datenbankpartition ausgeführt werden.

Der Befehl **db2relocatedb** kann nicht zum Verlagern einer Datenbank verwendet werden, für die eine Ladeoperation aktiv ist, oder die auf die Beendigung eines Befehls **LOAD RESTART** oder **LOAD TERMINATE** wartet.

Einschränkung: In einer Umgebung mit partitionierten Datenbanken können Sie keinen vollständigen Knoten verlagern, wenn dieser Knoten eine von zwei oder mehr logischen Partitionen repräsentiert, die sich in derselben Einheit befinden.

db2look - DB2-Statistiktool und DDL-Extraktionstool

Extrahiert die DDL-Anweisungen (DDL = Data Definition Language), die zum Reproduzieren der Datenbankobjekte einer Produktionsdatenbank in einer Testdatenbank erforderlich sind. Der Befehl **db2look** generiert die DDL-Anweisungen nach Objekttyp. Beachten Sie, dass dieser Befehl außer benutzerdefinierte Funktionen und gespeicherte Prozeduren alle Objekte unter dem Schema SYSTOOLS ignoriert.

Oft ist es vorteilhaft, über ein Testsystem zu verfügen, das eine Untermenge der Daten eines Produktionssystems enthält. Jedoch sind die Zugriffspläne, die auf einem solchen Testsystem ausgewählt werden, nicht unbedingt dieselben wie die, die für das Produktionssystem ausgewählt würden. Mit dem Tool **db2look** können Sie jedoch eine Testdatenbank erstellt, deren Zugriffspläne den auf dem Produktionssystem verwendeten Plänen ähneln. Mit diesem Tool können Sie die UPDATE-Anweisungen generieren, die erforderlich sind, um die Katalogstatistiken für die Objekte einer Produktionsdatenbank in einer Testdatenbank zu replizieren. Außerdem können Sie mit diesem Tool die Befehle **UPDATE DATABASE CONFIGURATION**, **UPDATE DATABASE MANAGER CONFIGURATION** und **db2set**generieren, damit die Werte der auf das Abfrageoptimierungsprogramm bezogenen Konfigurationsparameter und der Registrierdatenbankvariablen der Testdatenbank denen einer Produktionsdatenbank entsprechen.

Die vom Befehl **db2look** generierten DDL-Anweisungen sollten überprüft werden, da sie möglicherweise nicht alle Merkmale der ursprünglichen SQL-Objekte reproduzieren. Bei Tabellenbereichen in Umgebungen mit partitionierter Datenbank sind die DDL-Anweisungen möglicherweise nicht vollständig, wenn manche Datenbankpartitionen nicht aktiv sind. Stellen Sie mit dem Befehl **ACTIVATE DATABASE** sicher, dass alle Datenbankpartitionen aktiv sind.

Berechtigung

Zugriffsrecht SELECT für die Systemkatalogtabellen.

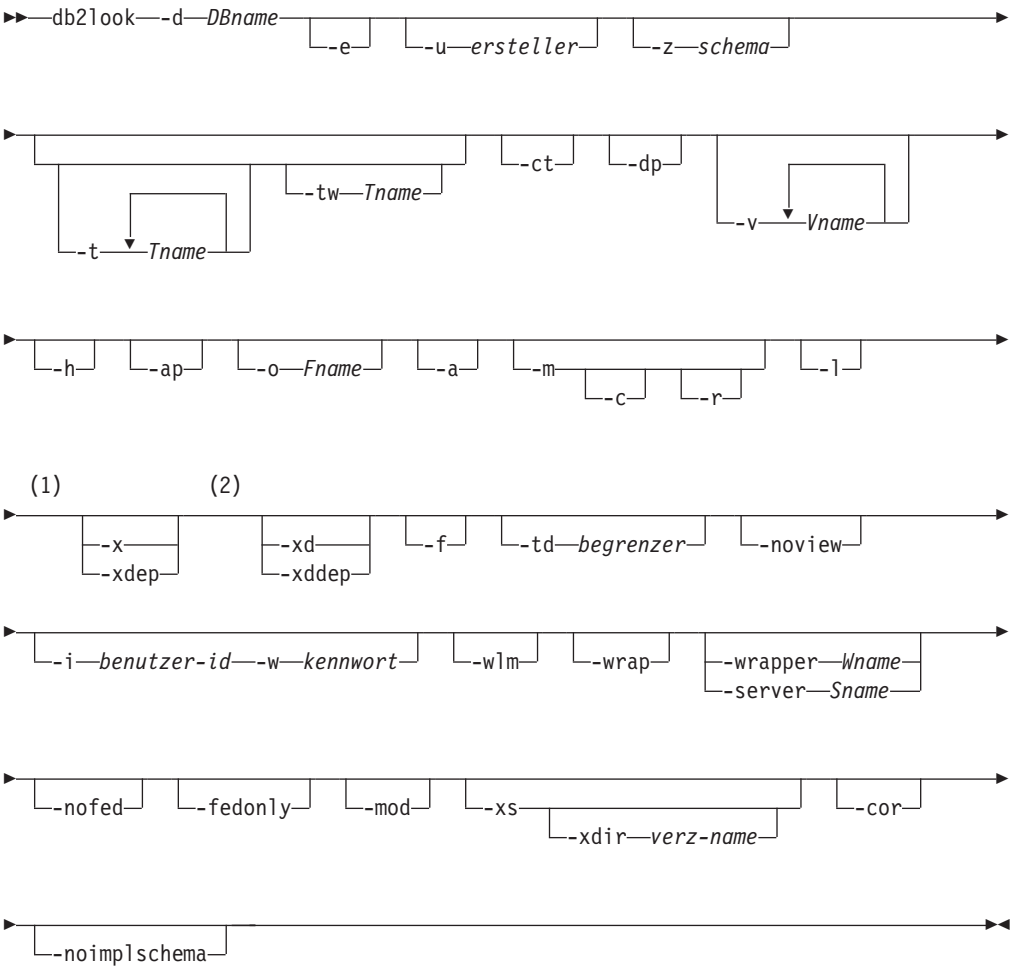
In manchen Fällen, z. B. bei der DDL-Generierung für Tabellenbereichscontainer, benötigen Sie eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL
- SYSMAINT
- SYSMON
- DBADM
- Zugriffsrecht EXECUTE für die Tabellenfunktion ADMIN_GET_STORAGE_PATHS

Erforderliche Verbindung

Keine

Befehlssyntax



Anmerkungen:

1 Sie können den Parameter **-x** und den Parameter **-xdep** nicht gleichzeitig angeben.

2 Sie können den Parameter **-xd** und den Parameter **-xddep** nicht gleichzeitig angeben.

Befehlsparameter

-d *DBname*

Aliasname der Produktionsdatenbank, die abgefragt werden soll. *DBname* kann der Name einer DB2 for Linux, UNIX and Windows- oder DB2 Version 9.1 für z/OS-Datenbank sein. Wenn *DBname* eine DB2 for z/OS-Datenbank ist, werden mit dem Befehl **db2look** die folgenden Anweisungen für OS/390- und z/OS-Objekte generiert:

- DDL-Anweisungen für Tabellen, Indizes, Sichten und benutzerdefinierte einzigartige Typen
- UPDATE-Statistikanweisungen für Tabellen, Spalten, Spaltenverteilungen und Indizes

Diese DDL- und UPDATE-Statistikanweisungen können auf eine DB2 for Linux, UNIX and Windows-Datenbank angewendet werden und nicht auf

eine DB2 for z/OS-Datenbank. Diese Anweisungen sind hilfreich, wenn Sie OS/390- und z/OS-Objekte extrahieren und in einer DB2 for Linux, UNIX and Windows-Datenbank erneut erstellen möchten.

-e Extrahiert DDL-Anweisungen für folgende Datenbankobjekte:

- Aliasnamen
- Prüfrichtlinien
- Prüfungen auf Integritätsbedingungen
- Funktionszuordnungen
- Funktionsschablonen
- Globale Variablen
- Indizes (einschließlich der partitionierten Indizes in partitionierten Tabellen)
- Indexspezifikationen
- MQTs (Materialized Query Tables)
- Kurznamen
- Primärschlüssel, referenzielle Integrität und Prüfung auf Integritätsbedingungen
- Referenzielle Integritätsbedingungen
- Rollen
- Schemata
- Sicherheitskennsätze
- Sicherheitskennsatzkomponenten
- Sicherheitsrichtlinien
- Sequenzen
- Server
- Gespeicherte Prozeduren
- Tabellen

  Anmerkung: Werte aus der Spalte STATISTICS_PROFILE in der Katalogtabelle SYSIBM.SYSTABLES sind nicht enthalten.
- Auslöser
- Gesicherte Kontexte
- Typenzuordnungen
- Benutzerzuordnungen
- Benutzerdefinierte einzigartige Typen
- Benutzerdefinierte Funktionen
- Benutzerdefinierte Methoden
- Benutzerdefinierte strukturierte Typen
- Benutzerdefinierte Umsetzungen
- Sichten
- Wrapper

Wenn Sie mithilfe von DDL-Anweisungen, die mit dem Befehl **db2look** erstellt wurden, eine benutzerdefinierte Funktion neu erstellen möchten, muss der Quellcode, auf den die Funktion verweist (z. B. die Klausel EXTERNAL NAME) verfügbar sein, damit die Funktion verwendet werden kann.

-u *ersteller*

Generiert DDL-Anweisungen für Objekte, die mit der angegebenen Ersteller-ID erstellt wurden. Begrenzt die Ausgabe auf Objekte, die mit der angegebenen Ersteller-ID erstellt wurden. Die Ausgabe enthält keine funktionsunfähigen Objekte. Verwenden Sie zum Anzeigen der funktionsunfähigen Objekte den Parameter **-a**. Wenn Sie den Parameter **-a** angeben, wird der Parameter **-u** ignoriert.

-z *schema*

Generiert DDL-Anweisungen für Objekte, die den angegebenen Schemanamen haben. Begrenzt die Ausgabe auf Objekte, die den angegebenen Schemanamen haben. Die Ausgabe enthält keine funktionsunfähigen Objekte. Verwenden Sie zum Anzeigen der funktionsunfähigen Objekte den Parameter **-a**. Wenn Sie den Parameter **-z** nicht angeben, werden Objekte mit allen Schemanamen extrahiert. Wenn Sie den Parameter **-a** angeben, wird der Parameter **-z** ignoriert. Dieser Parameter wird auch bei föderierten DDL-Anweisungen ignoriert.

-t *Tname1 Tname2 ... TnameN*

Generiert DDL-Anweisungen für die angegebenen Tabellen und die von ihnen abhängigen Objekte. Begrenzt die Ausgabe auf die in der Tabellenliste angegebenen Tabellen und generiert die DDL-Anweisungen für alle abhängigen Objekte aller benutzerdefinierten Tabellen. Die maximale Tabellenanzahl ist 30. Geben Sie die Liste wie folgt an:

- Trennen Sie an mit einem Leerzeichen.
- Schließen Sie Namen, bei denen die Groß-/Kleinschreibung zu beachten ist, sowie Doppelbytezeichensatz-Namen mit einem umgekehrten Schrägstrich (\) und mit Anführungszeichen (" ") ein (z. B. `\" MeineTabelle \"`.
- Schließen Sie Tabellennamen, die aus mehreren Wörtern bestehen, mit einem umgekehrten Schrägstrich und zwei Sets von Anführungszeichen ein (z. B. `"\"Meine Tabelle\""`), damit sie vom Befehlszeilenprozessor (CLP) nicht wie einzelne Wörter behandelt werden. Wenn Sie nur ein einziges Set von Anführungszeichen verwenden (z. B. `"Meine Tabelle"`), werden alle Wörter in Großbuchstaben umgewandelt und der Befehl **db2look** sucht nach einem Tabellennamen in Großbuchstaben (z. B. MEINE TABELLE).

Wenn Sie den Parameter **-t** mit dem Parameter **-l** angeben, werden partitionierte Tabellen unterstützt.

Mithilfe von zweiteiligen Tabellennamen im Format *schema.tabelle* können Sie einen Tabellennamen ohne Verwendung des Parameters **-z** *schema* vollständig qualifizieren. Verwenden Sie zweiteilige Tabellennamen, wenn die Tabelle abhängige Objekte besitzt, die ein anderes Schema als das Tabellenschema aufweisen, und Sie für die abhängigen Objekte DDL-Anweisungen generieren müssen. Wenn Sie das Schema mit dem Parameter **-z** *schema* angeben, schließt der Parameter abhängige Objekte aus, die nicht dasselbe übergeordnete Schema aufweisen, was die Generierung von DDL-Anweisungen für die abhängigen Objekte verhindert.

-tw *Tname*

Generiert DDL-Anweisungen für die Tabellen, deren Namen mit dem Muster übereinstimmen, das Sie mit *Tname* angeben, und generiert die DDL-Anweisungen für alle Objekte, die von diesen Tabellen abhängig sind. *Tname* darf nur einen einzigen Wert enthalten. Das Unterstreichungszeichen (_)

in *Tname* steht für ein beliebiges einzelnes Zeichen. Das Prozentzeichen (%) steht für eine Zeichenfolge aus null oder mehr Zeichen. Wenn **-tw** angegeben ist, wird die Option **-t** ignoriert.

Mithilfe von zweiteiligen Tabellennamen im Format *schema.tabelle* können Sie einen Tabellennamen ohne Verwendung des Parameters **-z** *schema* vollständig qualifizieren. Verwenden Sie zweiteilige Tabellennamen, wenn die Tabelle abhängige Objekte besitzt, die ein anderes Schema als das Tabellenschema aufweisen, und Sie für die abhängigen Objekte DDL-Anweisungen generieren müssen. Wenn Sie das Schema mit dem Parameter **-z** *schema* angeben, schließt der Parameter abhängige Objekte aus, die nicht dasselbe übergeordnete Schema aufweisen, was die Generierung von DDL-Anweisungen für die abhängigen Objekte verhindert.

-ct Generiert DDL-Anweisungen nach dem Zeitpunkt der Objekterstellung. Die DDL-Anweisungen der Objekte werden möglicherweise nicht in der richtigen Abhängigkeitsreihenfolge angezeigt. Bei Angabe des Parameters **-ct** unterstützt der Befehl **db2look** nur die folgenden zusätzlichen Parameter: **-e**, **-a**, **-u, -z**, **-t**, **-tw**, **-v**, **-l**, **-noview** und **-wlm**. Bei Verwendung des Parameters **-ct** mit den Parametern **-z** und **-t** generiert der Befehl **db2look** die erforderlichen UPDATE-Anweisungen zum Replizieren der Statistiken für Tabellen, Statistiksichten, Spalten und Indizes.

-dp Generiert vor einer Anweisung CREATE eine Anweisung DROP. Die Anweisung DROP funktioniert möglicherweise nicht, wenn ein Objekt von dem gelöschten Objekt abhängig ist. Sie können beispielsweise kein Schema löschen, wenn eine vorhandene Tabelle von diesem Schema abhängig ist, und Sie können keinen benutzerdefinierten Typ oder keine benutzerdefinierte Funktion löschen, wenn ein anderes Element (Typ, Funktion, Auslöser oder Tabelle) davon abhängig ist. Bei typisierten Tabellen wird die Anweisung DROP TABLE HIERARCHY nur für die Stammtabelle generiert. Für Indizes, Primär- und Fremdschlüssel sowie für Integritätsbedingungen wird keine Anweisung DROP generiert, da sie beim Löschen der Tabelle ebenfalls gelöscht werden. Eine Tabelle, die über das Attribut RESTRICT ON DROP verfügt, kann nicht gelöscht werden.

-v *Vname1 Vname2 ... VnameN*
Generiert DDL-Anweisungen für die angegebenen Sichten, jedoch nicht für die von ihnen abhängigen Objekte. Die maximale Anzahl für Sichten ist 30. Die Regeln für Tabellennamen, bei denen die Groß-/Kleinschreibung zu beachten ist, Doppelbytezeichensatz-Tabellennamen und Tabellennamen, die aus mehreren Wörtern bestehen, gelten auch für Sichtnamen. Wenn Sie den Parameter **-t** angeben, wird der Parameter **-v** ignoriert.

Sie können einen zweiteiligen Sichtnamen im Format *schema.sicht* verwenden, um eine Sicht vollständig zu qualifizieren.

-h Hilfetext anzeigen. Wenn Sie diesen Parameter angeben, werden alle übrigen Parameter ignoriert.

-ap Generiert die Anweisungen AUDIT USING, die erforderlich sind, um Prüfrichtlinien für andere Datenbankobjekte zuzuordnen.

-o *Fname*
Schreibt die Ausgabe in die Datei *Fname*. Wenn Sie keine Erweiterung angeben, wird die Erweiterung `.sql` verwendet. Wird dieser Parameter nicht angegeben, wird die Ausgabe in die Standardausgabe geschrieben.

-a Generiert DDL-Anweisungen für Objekte, die von einem beliebigen Benutzer erstellt wurden, einschließlich funktionsunfähiger Objekte. Wenn Sie

diesen Parameter beispielsweise mit dem Parameter **-e** angeben, werden DDL-Anweisungen für alle Objekte in der Datenbank extrahiert. Wenn Sie diesen Parameter mit dem Parameter **-m** angeben, werden UPDATE-Anweisungen für die Statistiken aller von Benutzern erstellten Tabellen und Indizes in der Datenbank extrahiert.

Wenn Sie weder **-u** noch **-a** als Parameter auswählen, wird die Umgebungsvariable USER verwendet. Auf UNIX-Betriebssystemen brauchen Sie diese Variable nicht explizit festzulegen. Auf Windows-Betriebssystemen gibt es jedoch keinen Standardwert für die Umgebungsvariable USER. Daher müssen Sie bei SYSTEM-Variablen eine Benutzervariable festlegen oder für die Sitzung den Befehl **set USER=***benutzername* absetzen.

-m Generiert die erforderlichen UPDATE-Anweisungen zum Replizieren der Statistiken für Tabellen, Statistiksichten, Spalten und Indizes. Die Verwendung des Parameters **-m** wird als Ausführung im Nachahmungsmodus (mimic) bezeichnet.

- **-c** Wenn Sie diese Option angeben, generiert der Befehl **db2look** keine COMMIT-, CONNECT- und CONNECT RESET-Anweisungen. Standardmäßig werden diese Anweisungen generiert. Diese Option wird ignoriert, außer Sie geben außerdem den Parameter **-m** oder **-e** an.
- **-r** Wenn Sie diese Option zusammen mit dem Parameter **-m** angeben, generiert der Befehl **db2look** den Befehl **RUNSTATS** nicht. Standardmäßig wird der Befehl **RUNSTATS** generiert.

Wichtig: Wenn Sie das Befehlsprozessorscript, das mit dem Befehl **db2look** mit dem Parameter **-m** erstellt wird, für eine andere Datenbank (z. B. zur Erreichung der Übereinstimmung der Katalogstatistiken der Testdatenbank mit denen des Produktionssystems) ausführen wollen, dann müssen beide Datenbanken mit dem gleichen codierten Zeichensatz und der gleichen Gebietsangabe arbeiten.

-l Generiert DDL-Anweisungen für folgende Datenbankobjekte:

- Benutzerdefinierte Tabellenbereiche
- Benutzerdefinierte Speichergruppen
- Benutzerdefinierte Datenbankpartitionsgruppen
- Benutzerdefinierte Pufferpools

-x Generiert DDL-Berechtigungsanweisungen, z. B. GRANT-Anweisungen.

Die folgenden Berechtigungen werden unterstützt:

- Spalten: UPDATE, REFERENCES
- Datenbanken: ACCESSCTRL, BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, DATAACCESS, DBADM, EXPLAIN, IMPLICIT_SCHEMA, LOAD, QUIESCE_CONNECT, SECADM, SQLADM, WLMADM
- Freistellungen
- Globale Variablen
- Indizes: CONTROL
- Pakete: CONTROL, BIND, EXECUTE
- Rollen
- Schemata: CREATEIN, DROPIN, ALTERIN
- Sicherheitskennsätze

- Reihenfolgen: USAGE, ALTER
- Gespeicherte Prozeduren: EXECUTE
- Tabellen: ALTER, SELECT, INSERT, DELETE, UPDATE, INDEX, REFERENCE, CONTROL
- Sichten: SELECT, INSERT, DELETE, UPDATE, CONTROL
- Benutzerdefinierte Funktionen (UDFs): EXECUTE
- Benutzerdefinierte Methoden: EXECUTE
- Tabellenbereiche: USE
- Workloads: USAGE

Anmerkung: Dieser Parameter generiert keine DDL-Berechtigungsanweisungen für abhängige Objekte, wenn er mit dem Parameter **-t** oder **-tw** verwendet wird. Verwenden Sie zum Generieren von DDL-Berechtigungsanweisungen für übergeordnete und abhängige Objekte den Parameter **-xdep**.

-xdep Generiert DDL-Berechtigungsanweisungen, z. B. GRANT-Anweisungen, für übergeordnete und abhängige Objekte. Dieser Parameter wird ignoriert, wenn der Parameter **-t** oder **-tw** nicht angegeben wird. Die folgenden Berechtigungen werden unterstützt:

- Spalten: UPDATE, REFERENCES
- Indizes: CONTROL
- Gespeicherte Prozeduren: EXECUTE
- Tabellen: ALTER, SELECT, INSERT, DELETE, UPDATE, INDEX, REFERENCE, CONTROL
- Tabellenbereiche: USE
- Benutzerdefinierte Funktionen (UDFs): EXECUTE
- Benutzerdefinierte Methoden: EXECUTE
- Sichten: SELECT, INSERT, DELETE, UPDATE, CONTROL

-xd Generiert DDL-Berechtigungsanweisungen, einschließlich DDL-Berechtigungsanweisungen für Objekte, deren Berechtigungen zum Zeitpunkt der Objekterstellung von SYSIBM erteilt wurden. Der Parameter generiert keine Authorisierungs-DLLs für Systemkatalogtabellen und Katalogsichten.

Anmerkung: Dieser Parameter generiert keine DDL-Berechtigungsanweisungen für abhängige Objekte, wenn er mit dem Parameter **-t** oder **-tw** verwendet wird. Verwenden Sie zum Generieren von DDL-Berechtigungsanweisungen für übergeordnete und abhängige Objekte den Parameter **-xddep**.

-xddep
Generiert alle DDL-Berechtigungsanweisungen für übergeordnete und abhängige Objekte, einschließlich DDL-Berechtigungsanweisungen für Objekte, deren Berechtigungen zum Zeitpunkt der Objekterstellung von SYSIBM erteilt wurden. Dieser Parameter wird ignoriert, wenn der Parameter **-t** oder **-tw** nicht angegeben wird.

-f Extrahiert die Konfigurationsparameter und Registrierdatenbankvariablen, die das Abfrageoptimierungsprogramm betreffen.

-td *begrenzer*
Gibt den Anweisungsbegrenzer für SQL-Anweisungen an, die mit dem Befehl **db2look** generiert werden. Der Standardbegrenzer ist ein Semikolon (;).

Verwenden Sie diesen Parameter, wenn Sie den Parameter **-e** angeben, weil die extrahierten Objekte möglicherweise Auslöser oder SQL-Routinen enthalten.

-noview
Gibt an, dass CREATE VIEW-DDL-Anweisungen nicht extrahiert werden.

-i *benutzer-id*
Gibt die Benutzer-ID an, mit der sich der Befehl **db2look** an einem fernen System anmeldet. Wenn Sie diesen Parameter und den Parameter **-w** angeben, kann der Befehl **db2look** für eine Datenbank auf einem fernen System ausgeführt werden. Die lokale und die ferne Datenbank müssen dieselbe DB2-Version verwenden.

-w *kennwort*
Gibt das Kennwort an, mit dem sich der Befehl **db2look** an einem fernen System anmeldet. Wenn Sie diesen Parameter und den Parameter **-i** angeben, kann der Befehl **db2look** für eine Datenbank auf einem fernen System ausgeführt werden. Die lokale und die ferne Datenbank müssen dieselbe DB2-Version verwenden.

-wlm Generiert eine WLM-spezifische DDL-Ausgabe, mit der CREATE- und ALTER-Anweisungen für folgende Elemente generiert werden können:

- Histogramme
- Serviceklassen
- Schwellenwerte
- WLM-Ereignismonitore
- Workloads
- Arbeitsaktionssets
- Arbeitsklassensets

-wrap Generiert verschlüsselte Versionen von DDL-Anweisungen für Routinen, Trigger, Sichten und PL/SQL-Pakete.

-wrapper *Wname*
Generiert DDL-Anweisungen für föderierte Objekte, die für den angegebenen Wrapper gelten. Die folgenden föderierten DDL-Anweisungen können generiert werden:

- CREATE FUNCTION ... AS TEMPLATE
- CREATE FUNCTION MAPPING
- CREATE INDEX SPECIFICATION
- CREATE NICKNAME
- CREATE SERVER
- CREATE TYPE MAPPING
- CREATE USER MAPPING
- CREATE WRAPPER
- GRANT (Zugriffsrechte für Kurznamen, Server, Indizes)

Wenn Sie keinen oder mehrere Wrappernamen angeben, wird ein Fehler zurückgegeben.

-server *Sname*
Generiert DDL-Anweisungen für föderierte Objekte, die für den angegebenen Server gelten. Die folgenden föderierten DDL-Anweisungen können generiert werden:

- CREATE FUNCTION ... AS TEMPLATE

- CREATE FUNCTION MAPPING
- CREATE INDEX SPECIFICATION
- CREATE NICKNAME
- CREATE SERVER
- CREATE TYPE MAPPING
- CREATE USER MAPPING
- CREATE WRAPPER
- GRANT (Zugriffsrechte für Kurznamen, Server, Indizes)

Wenn Sie keinen oder mehrere Servernamen angeben, wird ein Fehler zurückgegeben.

-nofed Gibt an, dass keine föderierten DDL-Anweisungen generiert werden. Wenn Sie diesen Parameter angeben, werden die Parameter **-wrapper** und **-server** ignoriert.

-fedonly
Gibt an, dass nur föderierte DDL-Anweisungen generiert werden.

-mod Generiert DDL-Anweisungen für die einzelnen Module und für alle in den einzelnen Modulen definierten Objekte.

-xs Exportiert alle Dateien, die zum Registrieren von XML-Schemata und DTDs in der Zieldatenbank erforderlich sind, und generiert entsprechende Befehle für ihre Registrierung. Die Gruppe der XSR-Objekte, die exportiert werden, wird durch die Parameter **-u**, **-z** und **-a** gesteuert.

-xdir *verz-name*
Exportiert XML-bezogene Dateien in den angegebenen Pfad. Wird dieser Parameter nicht angegeben, werden alle XML-bezogenen Dateien in das aktuelle Verzeichnis exportiert.

-cor Generiert DDL-Anweisungen mit der Klausel CREATE OR REPLACE unabhängig davon, ob die Anweisungen diese Klausel ursprünglich enthalten haben.

-noimplschema
Gibt an, dass Anweisungen des Typs CREATE SCHEMA DDL für implizit erstellte Schemas nicht generiert werden. Wenn Sie diesen Parameter festlegen, muss auch der Parameter **-e** angegeben werden.

Beispiele

Die folgenden Beispiele zeigen, wie der Befehl **db2look** zu verwenden ist:

- Generieren der DDL-Anweisungen für Objekte, die vom Benutzer `walid` in der Datenbank DEPARTMENT erstellt wurden. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -u walid -e -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für Objekte, die den Schemanamen `ianhe` aufweisen, der vom Benutzer `walid` in der Datenbank DEPARTMENT erstellt wurden. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -u walid -z ianhe -e -o db2look.sql
  ```

- Generieren der UPDATE-Anweisungen zum Replizieren der Statistiken für die Datenbankobjekte, die vom Benutzer `walid` in der Datenbank DEPARTMENT erstellt wurden. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -u walid -m -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für die vom Benutzer `walid` erstellten Objekte und der UPDATE-Anweisungen zum Replizieren der Statistikdaten zu den vom gleichen Benutzer erstellten Datenbankobjekten. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  +db2look -d department -u walid -e -m -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für die von allen Benutzern in der Datenbank DEPARTMENT erstellten Objekte. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -a -e -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für alle benutzerdefinierten Datenbankpartitionsgruppen, Pufferpools und Tabellenbereiche. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -l -o db2look.sql
  ```

- Generieren der UPDATE-Anweisungen für die auf das Optimierungsprogramm bezogenen Konfigurationsparameter der Datenbank und des Datenbankmanagers sowie der **db2set**-Befehle für auf das Optimierungsprogramm bezogene Registrierdatenbankvariablen in der Datenbank DEPARTMENT. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -f -o db2look.sql
  ```

- Generieren der **db2set**-Befehle für auf das Optimierungsprogramm bezogene Registrierdatenbankvariablen und der folgenden Anweisungen für die Datenbank DEPARTMENT:
  - DDL-Anweisungen für alle Datenbankobjekte
  - UPDATE-Anweisungen zum Replizieren der Statistiken für alle Tabellen und Indizes
  - GRANT-Berechtigungsanweisungen
  - UPDATE-Anweisungen für die auf das Optimierungsprogramm bezogenen Konfigurationsparameter der Datenbank und des Datenbankmanagers
  - **db2set**-Befehle für auf das Optimierungsprogramm bezogene Registrierdatenbankvariablen
  - DDL-Anweisungen für alle benutzerdefinierten Datenbankpartitionsgruppen, Pufferpools und Tabellenbereiche

  Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -a -e -m -l -x -f -o db2look.sql
  ```

- Generieren aller DDL-Anweisungen zur Autorisierung für alle Objekte in der Datenbank DEPARTMENT, einschließlich der vom ursprünglichen Ersteller erstellten Objekte. (In diesem Fall wurden die Autorisierungen zum Zeitpunkt der Objekterstellung von SYSIBM erteilt.) Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -xd -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für die von allen Benutzern in der Datenbank DEPARTMENT erstellten Objekte. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -a -e -td % -o db2look.sql
  ```

  Die Ausgabe kann anschließend vom Befehlszeilenprozessor gelesen werden:

  ```
  db2 -td% -f db2look.sql
  ```

- Generieren der DDL-Anweisungen für Objekte in der Datenbank DEPARTMENT, ausgenommen die CREATE VIEW-Anweisungen. Die Ausgabe wird an die Datei `db2look.sql` gesendet.

  ```
  db2look -d department -e -noview -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für Objekte in der Datenbank DEPARTMENT, die sich auf die angegebenen Tabellen beziehen. Die Ausgabe wird an die Datei db2look.sql gesendet.

  ```
  db2look -d department -e -t tab1 "\"Meine TaBellE2\"" -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für alle Objekte (föderierte und nicht föderierte) in der föderierten Datenbank FEDDEPART. Föderierte DDL-Anweisungen werden nur generiert, soweit sie für den angegebenen Wrapper (FEDWRAP) gelten. Die Ausgabe wird an die Standardausgabe gesendet.

  ```
  db2look -d feddepart -e -wrapper fedwrap
  ```

- Generieren einer Scriptdatei, die nur nicht föderierte DDL-Anweisungen enthält. Der folgende Systembefehl kann für eine föderierte Datenbank (FEDDEPART) ausgeführt werden und dennoch nur Ausgabedaten wie bei der Ausführung für eine nicht föderierte Datenbank erzeugen. Die Ausgabe wird an die Datei out.sql gesendet.

  ```
  db2look -d feddepart -e -nofed -o out
  ```

- Generieren der DDL-Anweisungen für Objekte in der Datenbank DEPARTMENT, die den Schemanamen walid aufweisen. Die erforderlichen Dateien zum Registrieren der enthaltenen XML-Schemata und DTDs werden in das aktuelle Verzeichnis exportiert. Die Ausgabe wird an die Datei db2look.sql gesendet.

  ```
  db2look -d department -z walid -e -xs -o db2look.sql
  ```

- Generieren der DDL-Anweisungen für die von allen Benutzern in der Datenbank DEPARTMENT erstellten Objekte. Die zum Registrieren der enthaltenen XML-Schemata und DTDs erforderlichen Dateien werden in das Verzeichnis /home/ofer/ofer/ exportiert. Die Ausgabe wird an die Standardausgabe gesendet.

  ```
  db2look -d department -a -e -xs -xdir /home/ofer/ofer/
  ```

- Generieren von ausschließlich WLM-spezifischen DDL-Anweisungen für die Datenbank DEPARTMENT:

  ```
  db2look -d department -wlm
  ```

- Generieren der DDL-Anweisungen für alle Objekte in der Datenbank DEPARTMENT:

  ```
  db2look -d department -wlm -e -l
  ```

- Generieren der DDL-Anweisungen für die übergeordnete Tabelle TAB1 im Schema TABLES und der abhängigen Sicht von TAB1 mit der Bezeichnung VIEW1 im Schema VIEWS der Datenbank DB1. Die Ausgabe wird an die Datei db2look.sql gesendet.

  ```
  db2look -d DB1 -t TABLES.TAB1 -e -o db2look.sql
  ```

- Generieren der DDL-Anweisungen und der DDL-Berechtigungsanweisungen für die übergeordnete Tabelle TAB1 im Schema TABLES und der abhängigen Sicht von TAB1 mit der Bezeichnung VIEW1 im Schema VIEWS der Datenbank DB1. Die Ausgabe wird an die Datei db2look.sql gesendet.

  ```
  db2look -d DB1 -t TABLES.TAB1 -e -xdep -o db2look.sql
  ```

- Generieren der RUNSTATS DDL-Anweisungen für die Tabelle TABLE1 im Abbildemodus. Die Ausgabe wird an die Datei db2look.sql gesendet. Nicht alle verfügbaren RUNSTATS-Klauseln des Befehls werden unterstützt.

  ```
  db2look -d DB1 -t TABLE1 -m  -e -o db2look.sql
  ```

Hinweise zur Verwendung

Auf Windows-Betriebssystemen müssen Sie den Befehl **db2look** über ein DB2-Befehlsfenster absetzen.

Standardmäßig verfügt der Instanzeigner über die Berechtigung EXECUTE für **db2look**-Pakete. Damit andere Benutzer den Befehl **db2look** ausführen können, muss der Instanzeigner die Berechtigung EXECUTE für **db2look**-Pakete erteilen. Um die **db2look**-Paketnamen zu ermitteln, kann der Befehl **db2bfd** folgendermaßen verwendet werden:

```
cd .../sqllib/bnd
db2bfd -b db2look.bnd
db2bfd -b db2lkfun.bnd
db2bfd -b db2lksp.bnd
```

Zum Erstellen von DDL-Anweisungen für föderierte Objekte müssen Sie in der Datenbankmanagerkonfiguration die Verwendung föderierter Systeme aktivieren. Nachdem die Scriptdatei von dem Befehl **db2look** generiert wurde, müssen Sie den Konfigurationsparameter **federated** auf YES setzen, bevor das Script ausgeführt wird. In einer föderierten Umgebung werden die folgenden Parameter des Befehls **db2look** unterstützt:

-ap
Generiert AUDIT USING-Anweisungen.

-e
Generiert DDL-Anweisungen für föderierte Objekte.

-f Extrahiert föderationsbezogene Informationen aus der Datenbankmanagerkonfiguration.

-m Extrahiert Statistiken für Kurznamen.

-x Generiert GRANT-Anweisungen zur Erteilung von Zugriffsrechten für föderierte Objekte.

-xd
Generiert DDL-Anweisungen, um Zugriffsrechte, die vom System erteilt wurden, zu föderierten Objekten hinzuzufügen.

-wlm
Generiert WLM-spezifische DDL-Anweisungen.

Wenn die Kurznamenspalte und die Spalte der fernen Tabelle unterschiedliche Datentypen aufweisen, wird mit dem Befehl **db2look** eine Anweisung ALTER COLUMN für die Kurznamenspalte generiert.

Sie müssen das Ausgabescript modifizieren, um die fernen Kennwörter für die CREATE USER MAPPING-Anweisungen hinzuzufügen.

Sie müssen das Ausgabescript des Befehls **db2look** modifizieren, indem Sie AUTHORIZATION und PASSWORD zu den verwendeten CREATE SERVER-Anweisungen hinzufügen, um eine Instanz der DB2-Produktfamilie als Datenquelle zu definieren.

Die Option **-tw** wird wie folgt verwendet:

- Zum Generieren der DDL-Anweisungen für Objekte in der Datenbank DEPARTMENT, die Tabellen zugeordnet sind, deren Namen mit abc beginnen, und zum Senden der Ausgabe an die Datei db2look.sql:

  ```
  db2look -d department -e -tw abc% -o db2look.sql
  ```

- Zum Generieren der DDL-Anweisungen für Objekte in der Datenbank DEPARTMENT, die Tabellen zugeordnet sind, deren Namen ein d als zweites Zeichen enthalten, und zum Senden der Ausgabe an die Datei db2look.sql:

  ```
  db2look -d department -e -tw _d% -o db2look.sql
  ```

- Der Befehl **db2look** verwendet das Prädikat LIKE, um zu bewerten, welche Tabellennamen mit dem im Argument *Tname* angegebenen Muster übereinstimmen. Wenn bei Verwendung des Prädikats LIKE das Zeichen _ oder % im Tabellennamen enthalten ist, muss unmittelbar vor diesem Zeichen der umgekehrte Schrägstrich angegeben werden. In diesem Fall kann weder das Zeichen _ noch das Zeichen % als Platzhalterzeichen in *Tname* verwendet werden. Geben Sie beispielsweise Folgendes ein, um die DDL-Anweisungen für Objekte in der Datenbank DEPARTMENT zu generieren, denen Tabellen zugeordnet sind, deren Namen weder als erstes noch als letztes Zeichen ein Prozentzeichen enthalten:

  ```
  db2look -d department -e -tw zeichenfolge\%string
  ```

- Tabellen- und Sichtnamen, bei denen Groß-/Kleinschreibung zu beachten ist, Doppelbytezeichensatz-Tabellen- und -Sichtnamen sowie Tabellen- und Sichtnamen, die aus mehreren Wörtern bestehen, müssen in einen umgekehrten Schrägstrich und Anführungszeichen eingeschlossen werden. Beispiel:

  ```
  \"Meine TabeLle\"
  ```

  Wenn ein Mehrbytezeichensatz- oder Doppelbytezeichensatz-Name nicht in das Begrenzungszeichen aus umgekehrtem Schrägstrich und Anführungszeichen eingeschlossen ist und dasselbe Byte enthält wie der Kleinbuchstabe, wird er in Großbuchstaben umgewandelt und der Befehl **db2look** sucht nach einem Datenbankobjekt mit dem umgewandelten Namen. Dies führt dazu, dass die DDL-Anweisung nicht extrahiert wird.
- Die Option **-tw** kann zusammen mit der Option **-x** (zum Generieren von GRANT-Zugriffsrechten) verwendet werden sowie mit der Option **-m** (zum Zurückgeben von Tabellen- und Spaltenstatistiken) und mit der Option **-l** (zum Generieren der DDL-Anweisungen für benutzerdefinierte Tabellenbereiche, Partitionsgruppen und Pufferpools). Wenn die Option **-t** zusammen mit der Option **-tw** angegeben wird, wird die Option **-t** (und das dazugehörige Argument *Tname*) ignoriert.
- Die Option **-tw** kann nicht zum Generieren der DDL-Anweisungen für Tabellen (und für deren zugehörige Objekte) verwendet werden, die sich in föderierten Datenquellen oder in DB2 Universal Database for z/OS and OS/390, in DB2 for i oder in DB2 Server for VSE & VM befinden.
- Die Option **-tw** wird nur über den Befehlszeilenprozessor unterstützt.

Wenn Sie versuchen, DDL-Anweisungen auf Systemen in einer Umgebung mit partitionierten Datenbanken zu generieren, wird anstelle der DDL-Anweisungen für Tabellenbereiche, die sich in inaktiven Datenbankpartitionen befinden, eine Warnung angezeigt. Um sicherzustellen, dass für alle Tabellenbereiche korrekte DDL-Anweisungen erstellt werden, müssen Sie alle Datenbankpartitionen aktivieren.

Sie können den Befehl **db2look** von einem DB2-Client an eine Datenbank absetzen, die dasselbe oder ein höheres Release aufweist als der Client; es ist jedoch nicht möglich, diesen Befehl von einem Client an eine Datenbank abzusetzen, die ein früheres Release aufweist als der Client. So können Sie den Befehl **db2look** beispielsweise von einem Client der Version 9.8 an eine Datenbank von Version 10.1 absetzen; es ist jedoch nicht möglich, den Befehl von einem Version 10.1-Client an eine Datenbank der Version 9.8 abzusetzen.

Wenn Sie das Dienstprogramm db2look aufrufen, generiert der Befehl **db2look** die DDL-Anweisungen für alle Objekte, die mithilfe einer nicht festgeschriebene Transaktion erstellt wurden.

Beim Extrahieren einer DDL-Anweisung für eine Sicherheitskennsatzkomponente des Typs 'array' generiert die extrahierte Anweisung möglicherweise keine Kompo-

nente, deren interne Darstellung (Codierung von Elementen in dem Array) mit der Darstellung dieser Komponente in der Datenbank übereinstimmt, für die Sie den Befehl **db2look** abgesetzt haben. Diese Diskrepanz kann auftreten, wenn Sie die Sicherheitskennsatzkomponente durch Hinzufügen von einem oder mehreren Elementen geändert haben. In diesem Fall sind für Daten, die aus einer Tabelle extrahiert und in eine andere, aus einer **db2look**-Ausgabe erstellte Tabelle versetzt werden, keine entsprechenden Sicherheitskennsatzwerte vorhanden, sodass die neue Tabelle möglicherweise nicht umfassend geschützt ist.

Zugehörige Informationen:

Namen für Kurznamenspalten und Indizes

Migrationsänderungen mit Auswirkungen auf Anwendungen

Vergleich zwischen den Dienstprogrammen INGEST, IMPORT und LOAD

In den folgenden Tabellen sind einige der wichtigsten Gemeinsamkeiten und Unterschiede zwischen dem Dienstprogramm INGEST, dem Dienstprogramm IMPORT und dem Dienstprogramm LOAD zusammengefasst.

Tabelle 17. Unterstützte Tabellentypen

Tabellentypen	INGEST	LOAD	IMPORT
Tabelle mit aufgehobener Zuordnung	nicht unterstützt	nicht unterstützt	nicht unterstützt
Globale temporäre Tabelle	nicht unterstützt	nicht unterstützt	nicht unterstützt
MDC-Tabelle (mehrdimensionales Clustering) oder ITC-Tabelle (Clustering anhand der Einfügungszeit)	unterstützt	unterstützt	unterstützt
Vom Benutzer verwaltete MQT-Tabelle (MQT - Materialized Query Table)	unterstützt	unterstützt	unterstützt
Kurzname	unterstützt	nicht unterstützt	unterstützt
Bereichsclustertabellen (RCT)	unterstützt	nicht unterstützt	unterstützt
Bereichspartitionierte Tabellen	unterstützt	unterstützt	unterstützt
Übersichtstabellen	unterstützt	unterstützt	unterstützt
Temporale Tabelle	unterstützt	unterstützt	unterstützt
Typisierte Tabellen	nicht unterstützt	nicht unterstützt	unterstützt
Nicht typisierte (reguläre) Tabelle	unterstützt	unterstützt	unterstützt
Aktualisierbare Sicht (außer typisierte Sicht)	unterstützt	nicht unterstützt	unterstützt

Tabelle 18. Unterstützte Datentypen

Tabellentypen	INGEST	LOAD	IMPORT
Numerisch: SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE, DECFLOAT	unterstützt	unterstützt	unterstützt
Zeichen: CHAR, VARCHAR, NCHAR, NVARCHAR und entsprechende FOR BIT DATA-Typen	unterstützt	unterstützt	unterstützt
Grafische: GRAPHIC, VARGRAPHIC	unterstützt	unterstützt	unterstützt
Datentypen 'long': LONG VARCHAR, LONG VARGRAPHIC	unterstützt	unterstützt	unterstützt
Datum/Zeit: DATE, TIME, TIMESTAMP, einschließlich TIMESTAMP(p)	unterstützt	unterstützt	unterstützt
DB2SECURITYLABEL	unterstützt	unterstützt	unterstützt
Große Objekte aus Dateien: BLOB, CLOB, DBCLOB, NCLOB	nicht unterstützt	unterstützt	unterstützt
Integrierte große Objekte	nicht unterstützt	unterstützt	unterstützt
XML-Daten aus Dateien	nicht unterstützt	unterstützt	unterstützt
Integrierte XML-Daten	nicht unterstützt	unterstützt	unterstützt
Einzigartiger Datentyp	unterstützt (falls er auf einem unterstützten integrierten Datentyp basiert)	unterstützt	unterstützt
Strukturierter Typ	nicht unterstützt	nicht unterstützt	unterstützt
Verweistyp	unterstützt	unterstützt	unterstützt

Tabelle 19. Unterstützte Eingabequellen

Eingabetyp	INGEST Erneut startbar?	LOAD Erneut startbar?	IMPORT Erneut startbar?
Cursor	nicht unterstützt nicht zutreffend	unterstützt ja	nicht unterstützt nicht zutreffend
Einheit	nicht unterstützt nicht zutreffend	unterstützt ja	nicht unterstützt nicht zutreffend
Datei	unterstützt ja	unterstützt ja	unterstützt ja
Pipe	unterstützt ja	unterstützt ja	nicht unterstützt nicht zutreffend

Tabelle 20. Unterstützte Eingabeformate

Tabellentypen	INGEST	LOAD	IMPORT
ASC (einschließlich binär)	unterstützt	unterstützt	unterstützt

Tabelle 20. Unterstützte Eingabeformate (Forts.)

Tabellentypen	INGEST	LOAD	IMPORT
Format DB2 z/OS UNLOAD	nicht unterstützt	nicht unterstützt	nicht unterstützt
DEL	unterstützt	unterstützt	unterstützt
IXF	nicht unterstützt	unterstützt	unterstützt

Es gibt eine Reihe weiterer wichtiger Unterschiede, die das Dienstprogramm INGEST von den Dienstprogrammen LOAD und IMPORT unterscheiden:

- Beim Dienstprogramm INGEST können die Eingabedatensätze zwischen den Feldern zusätzliche Felder enthalten, die Spalten entsprechen.
- Das Dienstprogramm INGEST unterstützt Aktualisierungs-, Lösch- und Zusammenführungsoperationen.
- Das Dienstprogramm INGEST unterstützt die Erstellung von Spaltenwerten aus Ausdrücken mit Feldwerten.
- Während der Ausführung des Dienstprogramms INGEST kann die Zieltabelle von anderen Anwendungen aktualisiert werden.

Dateiformate und Datentypen

Dateiformate der Dienstprogramme EXPORT/IMPORT/LOAD

Es werden vier Betriebssystemdateiformate beschrieben, die von den DB2-Dienstprogrammen EXPORT, IMPORT und LOAD unterstützt werden:

DEL ASCII-Format ohne universelle Zeilenbegrenzer. Dieses Format ermöglicht den Datenaustausch zwischen einer Vielzahl von Datenbankmanagern und Dateimanagern. Bei diesem allgemeinen Ansatz zum Speichern von Daten werden spezielle Zeichenbegrenzer verwendet, um Spaltenwerte zu trennen.

ASC ASCII-Format mit universellen Zeilenbegrenzern. Dieses Format dient zum Importieren oder Laden von Daten anderer Anwendungen, die Flachdateien mit ausgerichteten Spaltendaten erstellen.

PC/IXF
PC-Version des IXF-Formats (Integration Exchange Format). Dies ist das bevorzugte Format für den Austausch von Daten innerhalb des Datenbankmanagers. PC/IXF ist eine strukturierte Beschreibung einer Datenbanktabelle, die eine externe Darstellung der internen Tabelle enthält.

CURSOR
Ein Cursor, der für eine SQL-Abfrage deklariert wird. Dieser Dateityp wird nur durch das Dienstprogramm LOAD unterstützt.

Bei Verwendung des DEL- oder ASC-Datendateiformats müssen Sie die Tabelle, einschließlich der zugehörigen Spaltennamen und Datentypen, definieren, bevor Sie die Datei importieren. Die Datentypen in den Feldern der Betriebssystemdatei werden in den entsprechenden Datentyp in der Datenbanktabelle umgesetzt. Das Dienstprogramm IMPORT akzeptiert Daten mit geringfügigen Inkompatibilitätsproblemen, einschließlich Zeichendaten, die beim Import möglicherweise aufgefüllt bzw. abgeschnitten werden, und numerischer Daten, die in unterschiedliche Typen numerischer Felder importiert werden.

Bei Verwendung des Datendateiformats PC/IXF muss die Tabelle beim Starten der Importoperation noch nicht vorhanden sein. Der einzigartige Datentyp (UDT, User-Defined Distinct Typ) muss jedoch definiert sein. Andernfalls erhalten Sie eine Fehlernachricht zu einem Fehler durch einen nicht definierten Namen (SQL0204N). Ebenso werden UDTs beim Exportieren in das PC/IXF-Datendateiformat in der Ausgabedatei gespeichert.

Bei Verwendung des Dateityps CURSOR muss die Tabelle inklusive Spaltennamen und Datentypen definiert werden, bevor die Ladeoperation gestartet wird. Die Spaltentypen der SQL-Abfrage müssen mit den entsprechenden Spaltentypen in der Zieltabelle kompatibel sein. Es ist nicht erforderlich, dass der angegebene Cursor geöffnet ist, bevor die Ladeoperation gestartet wird. Das Dienstprogramm LOAD verarbeitet das gesamte Ergebnis der Abfrage, die dem angegebenen Cursor zugeordnet ist, und zwar unabhängig davon, ob der Cursor zum Abrufen von Zeilen verwendet wurde oder nicht.

Versetzen von Daten zwischen Plattformen - Hinweise zum Dateiformat

Kompatibilität ist beim Exportieren, Importieren oder Laden von Daten zwischen verschiedenen Plattformen wichtig. In den folgenden Abschnitten finden Sie Hinweise zu den PC/IXF- und DEL-Dateiformaten (ASCII-Format ohne universelle Zeilenbegrenzer) beim Versetzen von Daten zwischen verschiedenen Betriebssystemen.

PC/IXF-Dateiformat

PC/IXF ist das bevorzugte Dateiformat zum Versetzen von Daten zwischen Plattformen. PC/IXF-Dateien ermöglichen es dem Dienstprogramm LOAD oder IMPORT, numerische Daten, die normalerweise von der Maschine abhängig sind, maschinenunabhängig zu verarbeiten. Numerische Daten werden beispielsweise von der Intel-Architektur anders als von anderen Hardwarearchitekturen gespeichert und verwaltet.

Um die Kompatibilität der PC/IXF-Dateien für alle Produkte der DB2-Familie herzustellen, erstellt das Dienstprogramm EXPORT Dateien mit numerischen Daten im Intel-Format, während das Dienstprogramm IMPORT die Daten in diesem Format erwartet.

Abhängig von der Hardwareplattform konvertieren DB2-Produkte numerische Werte zwischen Intel- und Nicht-Intel-Formaten (mittels Bytefolgeumkehrung) sowohl während der Export- als auch der Importoperationen.

Auf dem Betriebssystem UNIX basierende Implementierungen von DB2-Datenbanken erstellen während des Exports keine mehrteiligen PC/IXF-Dateien. Der Import einer mehrteiligen PC/IXF-Datei ist dagegen zulässig, sofern die betreffende Datei von DB2 erstellt wurde. Wenn eine solche Datei importiert wird, müssen sich alle zugehörigen Teildateien im selben Verzeichnis befinden. Andernfalls meldet das Dienstprogramm einen Fehler.

Einteilige PC/IXF-Dateien, die mit dem DB2-Dienstprogramm EXPORT auf UNIX-Betriebssystemen erstellt werden, können von DB2-Datenbanken für Windows importiert werden.

DEL-Dateiformat (ASCII-Format ohne universelle Zeilenbegrenzer)

DEL-Dateien weisen betriebssystemspezifische Unterschiede auf, je nachdem, unter welchem Betriebssystem sie erstellt wurden. Es gibt folgende Unterschiede:

- Zeilentrennzeichen
  - Textdateien aus UNIX-Betriebssystemen verwenden ein Zeilenvorschubzeichen.
  - Textdateien aus anderen Betriebssystemen verwenden eine Folge aus Rücklauf- und Zeilenvorschubzeichen.
- Dateiendezeichen
  - Textdateien aus UNIX-Betriebssystemen verwenden kein Dateiendezeichen.
  - Textdateien aus anderen Betriebssystemen verwenden ein Dateiendezeichen (X'1A').

Da DEL-Exportdateien Textdateien sind, können sie von einem Betriebssystem auf ein anderes übertragen werden. Dateiübertragungsprogramme können betriebssystemabhängige Dateien unterschiedlich verarbeiten, wenn Sie die Dateien im Textmodus übertragen. Zeilentrennzeichen und Dateiendezeichen werden im Binärmodus nicht konvertiert.

Anmerkung: Wenn die Zeichendatenfelder Zeilentrennzeichen enthalten, werden diese ebenfalls während der Dateiübertragung konvertiert. Diese Konvertierung führt zu unvorhersehbaren Änderungen der Daten. Aus diesem Grund sollten Sie keine DEL-Exportdateien verwenden, um Daten zwischen Plattformen zu versetzen. Verwenden Sie stattdessen das PC/IXF-Dateiformat.

DEL-Dateiformat (ASCII-Format ohne universelle Zeilenbegrenzer)

Eine DEL-Datei (Delimited ASCII - ASCII-Format ohne universelle Zeilenbegrenzer) ist eine sequenzielle ASCII-Datei mit Zeilen- und Spaltenbegrenzungszeichen. Jede DEL-Datei ist ein Datenstrom aus ASCII-Zeichen, der sich aus Zellenwerten zusammensetzt, die zuerst nach Zeilen und dann nach Spalten geordnet sind. Die Zeilen im Datenstrom werden durch Zeilenbegrenzungszeichen getrennt. Die einzelnen Zellenwerte innerhalb einer jeden Zeile werden durch Spaltenbegrenzungszeichen getrennt.

Die folgende Tabelle beschreibt das Format von DEL-Dateien, die importiert oder als Ergebnis einer Exportaktion generiert werden können.

```
DEL-Datei ::= Zeile 1 Daten || Zeilenbegrenzer ||
              Zeile 2 Daten || Zeilenbegrenzer ||
             .
             .
             .
              Zeile n Daten || Wahlfreier Zeilenbegrenzer

Zeile i Daten ::= Zellenwert(i,1) || Spaltenbegrenzer ||
                  Zellenwert(i,2) || Spaltenbegrenzer ||
                .
                .
                .
                  Zellenwert(i,m)

Zeilenbegrenzer ::= ASCII-Zeilenvorschubfolge[a]

Spaltenbegrenzer ::= Standardwert: ASCII-Komma (,)[b]

Zellenwert(i,j) ::= Führende Leerzeichen
                    || ASCII-Darstellung eines numerischen Werts
```

```
                    (Integer, Dezimal oder Gleitkomma)
                  || Begrenzte Zeichenfolge
                  || Nicht begrenzte Zeichenfolge
                  || Nachgestellte Leerzeichen

Nicht begrenzte Zeichenfolge ::=   Ein Satz beliebiger Zeichen außer eines
                                   Zeilen- oder Spaltenbegrenzers

Begrenzte Zeichenfolge       ::= Ein Zeichenfolgebegrenzer ||
                                 Eine erweiterte Zeichenfolge ||
                                 Ein Zeichenfolgebegrenzer ||
                                 Nicht verwertbarer Rest

Nicht verwertbarer Rest ::= Ein Satz beliebiger Zeichen außer eines Zeilen-
                      oder Spaltenbegrenzers

Zeichenfolgebegrenzer      ::= Standardwert: Doppelte ASCII-
                               Anführungszeichen (")[c]

Erweiterte Zeichenfolge   ::= || Ein Satz beliebiger Zeichen außer
                                 eines Zeilen- oder Zeichenfolge-
                                 begrenzers, wenn der Änderungswert
                                 NODOUBLEDEL                                    angegeben ist
                              || Ein Satz beliebiger Zeichen außer
                                 eines Zeilen- oder Zeichenfolge-
                                 begrenzers, wenn die Zeichenfolge nicht
                                 Teil zweier aufeinander folgender Zeichen-
                                 folgebegrenzer ist
                              || Ein Satz beliebiger Zeichen außer
                                 eines Zeichenfolgebegrenzers, wenn
                                 der Zeichenfolgebegrenzer nicht Teil
                                 zweier aufeinander folgender Zeichen-
                                 folgebegrenzer ist und der Änderungswert
                                 angegeben ist

Dateieindezeichen     ::= Hex '1A' (nur Windows-Betriebssysteme)

ASCII-Darstellung eines numerischen Werts[d] ::= Wahlfreies Vorzeichen "+" oder "-"
    || 1 bis 31 Dezimalziffern mit wahlfreiem Dezimalzeichen vor,
       nach oder zwischen zwei Ziffern
    || Wahlfreier Exponent

Exponent ::= Zeichen "E" oder "e"
    || Wahlfreies Zeichen "+" oder "-"
    || 1 bis 3 Dezimalziffern ohne Dezimalzeichen

Dezimalziffer ::= Beliebiges Zeichen "0", "1", ... "9"

Dezimalzeichen ::= Standardwert: ASCII-Punkt (.)[e]
```

- [a] Als Satzbegrenzer wird das Zeilenvorschubzeichen, ASCII-Wert x0A, angenommen. Unter einem Windows-Betriebssystem generierte Daten können den 2-Byte-Standardwert für Rücklauf/Zeilenvorschub, 0x0D0A, verwenden. Daten in EBCDIC-Codepages müssen das EBCDIC-Zeichen für Zeilenvorschub (0x25) als Satzbegrenzer verwenden (EBCDIC-Daten können geladen werden, indem der Änderungswert `code page` für den Dateityp mit dem Befehl **LOAD** verwendet wird).
- [b] Der Spaltenbegrenzer kann mit dem Änderungswert `coldel` für den Dateityp angegeben werden.
- [c] Der Zeichenfolgebegrenzer kann mit dem Änderungswert `chardel` für den Dateityp angegeben werden.

  Anmerkung: Die Standardpriorität der Begrenzer ist:

  1. Satzbegrenzer

2. Zeichenbegrenzer
3. Spaltenbegrenzer

- [d] Wenn die ASCII-Darstellung eines numerischen Wertes einen Exponenten aufweist, handelt es sich um eine FLOAT-Konstante. Wenn sie ein Dezimalzeichen, jedoch keinen Exponenten aufweist, handelt es sich um eine DECIMAL-Konstante. Wenn sie kein Dezimalzeichen und keinen Exponenten aufweist, handelt es sich um eine INTEGER-Konstante.
- [e] Das Dezimalzeichen kann mit dem Änderungswert `decpt` für den Dateityp angegeben werden.

Vom Dienstprogramm EXPORT wird jedes Zeichenfolgebegrenzerbyte (der Standarwert ist ein doppeltes Anführungszeichen oder x22), das in Spaltendaten eingebettet ist, durch zwei Zeichenfolgebegrenzerbyte ersetzt (das Zeichen wird also verdoppelt). Dies ist erforderlich, damit die Parsingroutinen beim Import zwischen einem Zeichenfolgebegrenzerbyte, das Anfang und Ende einer Spalte definiert, und einem Zeichenfolgebegrenzerbyte unterscheiden können, das in den Spaltendaten enthalten ist. Dies ist bei der Verwendung einer exportierten DEL-Datei für einige andere Anwendungen als das Dienstprogramm EXPORT zu beachten. Dieselbe Zeichenverdoppelung erfolgt bei Zeichenfolgebegrenzern innerhalb von mit 'FOR BIT' markierten binären Spaltendaten.

DEL-Datentypbeschreibungen:

Tabelle 21. Gültige Datentypformate für das DEL-Dateiformat

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
BIGINT	Ganzzahlige Konstante (INTEGER) im Bereich von -9 223 372 036 854 775 808 bis 9 223 372 036 854 775 807.	ASCII-Darstellung eines numerischen Werts im Bereich von -9 223 372 036 854 775 808 bis 9 223 372 036 854 775 807. Dezimal- und Gleitkommazahlen werden ganzzahlig abgeschnitten.
BLOB, CLOB	Zeichendaten, die in Zeichenbegrenzer (z. B. doppelte Anführungszeichen) eingeschlossen sind.	Eine begrenzte oder nicht begrenzte Zeichenfolge. Die Zeichenfolge wird als Wert der Datenbankspalte verwendet.
BLOB_FILE, CLOB_FILE	Die Zeichendaten für jede BLOB-/CLOB-Spalte werden in einzelnen Dateien gespeichert, und der Dateiname ist in Zeichenbegrenzer eingeschlossen.	Der begrenzte oder nicht begrenzte Name der Datei, die die Daten enthält.
CHAR	Zeichendaten, die in Zeichenbegrenzer (z. B. doppelte Anführungszeichen) eingeschlossen sind.	Eine begrenzte oder nicht begrenzte Zeichenfolge. Die Zeichenfolge wird nach Bedarf abgeschnitten oder mit Leerzeichen aufgefüllt (X'20') und so an die Breite der Datenbankspalte angeglichen.

Tabelle 21. Gültige Datentypformate für das DEL-Dateiformat (Forts.)

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
DATE	*jjjjmmtt* (Jahr Monat Tag) ohne Spaltenbegrenzer. Beispiel: 19931029 Alternativ kann die Option DATESISO verwendet werden, um festzulegen, dass alle Datumswerte im ISO-Format exportiert werden sollen.	Eine begrenzte oder nicht begrenzte Zeichenfolge, die einen Datumswert in einem ISO-Format enthält, das mit dem Gebietscode der Zieldatenbank konsistent ist, oder eine nicht begrenzte Zeichenfolge im Format *jjjjmmtt*.
DBCLOB (nur DBCS)	Grafikdaten werden als begrenzte Zeichenfolge exportiert.	Eine begrenzte oder nicht begrenzte Zeichenfolge (Länge ist eine gerade Anzahl Byte). Die Zeichenfolge wird als Wert der Datenbankspalte verwendet.
DBCLOB_FILE (nur DBCS)	Die Zeichendaten für jede DBCLOB-Spalte werden in einzelnen Dateien gespeichert, und der Dateiname ist in Zeichenbegrenzer eingeschlossen.	Der begrenzte oder nicht begrenzte Name der Datei, die die Daten enthält.
DB2SECURITYLABEL	Spaltendaten werden als Rohdaten in Anführungszeichen (") exportiert. Verwenden Sie die Skalarfunktion SECLABEL_TO_CHAR in der Anweisung SELECT, um den Wert in das Zeichenfolgeformat des Sicherheitskennsatzes zu konvertieren.	Es wird standardmäßig davon ausgegangen, dass es sich beim Wert in der Datendatei um die tatsächlichen Byte handelt, die die interne Darstellung dieses Sicherheitskennsatzes bilden. Es wird davon ausgegangen, dass der Wert durch Anführungszeichen (" ") begrenzt ist.
DECIMAL	Eine DECIMAL-Konstante mit der Genauigkeit und der Anzahl der Kommastellen des Feldes, das exportiert wird. Mit dem Änderungswert `decplusblank` für den Dateityp kann angegeben werden, dass positiven Dezimalwerten ein Leerzeichen statt eines Pluszeichens (+) vorangestellt werden soll.	ASCII-Darstellung eines numerischen Werts, der nicht außerhalb des Bereichs der Datenbankspalte liegt, in die das Feld importiert wird. Wenn der Eingabewert mehr Ziffern nach dem Dezimalzeichen enthält, als in der Datenbankspalte Platz finden, werden die überzähligen Ziffern abgeschnitten.
FLOAT(long)	Eine FLOAT-Konstante im Bereich -10E307 bis 10E307.	ASCII-Darstellung eines numerischen Wertes im Bereich -10E307 bis 10E307.

Tabelle 21. Gültige Datentypformate für das DEL-Dateiformat (Forts.)

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
GRAPHIC (nur DBCS)	Grafikdaten werden als begrenzte Zeichenfolge exportiert.	Eine begrenzte oder nicht begrenzte Zeichenfolge (Länge ist eine gerade Anzahl Byte). Die Zeichenfolge wird nach Bedarf abgeschnitten oder mit Doppelbyteleerzeichen (X'8140') aufgefüllt und so an die Breite der Datenbankspalte angeglichen.
INTEGER	Ganzzahlige Konstante (INTEGER) im Bereich von -2 147 483 648 bis 2 147 483 647.	ASCII-Darstellung eines numerischen Werts im Bereich von -2 147 483 648 bis 2 147 483 647. Dezimal- und Gleitkommazahlen werden ganzzahlig abgeschnitten.
LONG VARCHAR	Zeichendaten, die in Zeichenbegrenzer (z. B. doppelte Anführungszeichen) eingeschlossen sind.	Eine begrenzte oder nicht begrenzte Zeichenfolge. Die Zeichenfolge wird als Wert der Datenbankspalte verwendet.
LONG VARGRAPHIC (nur DBCS)	Grafikdaten werden als begrenzte Zeichenfolge exportiert.	Eine begrenzte oder nicht begrenzte Zeichenfolge (Länge ist eine gerade Anzahl Byte). Die Zeichenfolge wird als Wert der Datenbankspalte verwendet.
SMALLINT	Ganzzahlige Konstante (INTEGER) im Bereich von -32 768 bis 32 767.	ASCII-Darstellung eines numerischen Werts im Bereich von -32 768 bis 32 767. Dezimal- und Gleitkommazahlen werden ganzzahlig abgeschnitten.
TIME	*ss.mm.ss* (Stunde Minuten Sekunden). Ein Zeitwert im ISO-Format, in Zeichenbegrenzer eingeschlossen. Beispiel: "09.39.43"	Eine begrenzte oder nicht begrenzte Zeichenfolge, die einen Zeitwert in einem Format enthält, das mit dem Gebietscode der Zieldatenbank konsistent ist.
TIMESTAMP	*jjjj-mm-tt-ss.mm.ss.nnnnnn* (Jahr Monat Tag Stunde Minuten Sekunden Mikrosekunden). Eine Zeichenfolge, die ein Datum und eine Uhrzeit darstellt, in Zeichenbegrenzer eingeschlossen.	Eine begrenzte oder nicht begrenzte Zeichenfolge, die einen Zeitmarkenwert enthält, dessen Speicherung in einer Datenbank zulässig ist.

Tabelle 21. Gültige Datentypformate für das DEL-Dateiformat (Forts.)

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
VARCHAR	Zeichendaten, die in Zeichenbegrenzer (z. B. doppelte Anführungszeichen) eingeschlossen sind.	Eine begrenzte oder nicht begrenzte Zeichenfolge. Die Zeichenfolge wird nach Bedarf abgeschnitten und so an die maximal zulässige Breite der Datenbankspalte angeglichen.
VARGRAPHIC (nur DBCS)	Grafikdaten werden als begrenzte Zeichenfolge exportiert.	Eine begrenzte oder nicht begrenzte Zeichenfolge (Länge ist eine gerade Anzahl Byte). Die Zeichenfolge wird nach Bedarf abgeschnitten und so an die maximal zulässige Breite der Datenbankspalte angeglichen.

Beispiel für DEL-Datei:
Es folgt ein Beispiel für eine DEL-Datei. Jede Zeile endet mit einer Zeilenvorschubfolge (unter den Windows-Betriebssystemen endet jede Zeile mit einer Folge aus einem Rücklauf- und einem Zeilenvorschubzeichen).

```
"Smith, Bob",4973,15.46
"Jones, Bill",12345,16.34
"Williams, Sam",452,193.78
```

Das folgende Beispiel zeigt die Verwendung von nicht begrenzten Zeichenfolgen. Das Spaltenbegrenzungszeichen wurde durch ein Semikolon ersetzt, weil die Zeichendaten ein Komma enthalten.

```
Smith, Bob;4973;15.46
Jones, Bill;12345;16.34
Williams, Sam;452;193.78
```

Anmerkung:

1. Ein Leerzeichen (X'20') ist als Begrenzer nie zulässig.
2. Leerzeichen vor dem ersten oder nach dem letzten Zeichen eines Zellenwertes werden beim Import gelöscht. Leerzeichen, die in einen Zellenwert eingebettet sind, werden nicht gelöscht.
3. Ein Punkt (.) ist kein gültiges Zeichenfolgebegrenzungszeichen, da dies zu Konflikten mit den Punkten in den Werten für Zeitmarken führen würde.
4. Bei reinem DBCS (Grafik), gemischtem DBCS und EUC sind Begrenzer auf den Bereich von x00 bis x3F (jeweils einschließlich) beschränkt.
5. Bei DEL-Daten, die in einer EBCDIC-Codepage angegeben werden, sind die Begrenzer nicht unbedingt mit den DBCS-Startzeichen und -Endezeichen identisch.
6. Unter den Windows-Betriebssystemen gibt das erste Dateiendezeichen (X'1A'), das sich nicht innerhalb von Zeichenbegrenzern befindet, das Dateiende an. Darauf folgende Daten werden nicht importiert.
7. Ein Nullwert wird durch das Fehlen eines Zellenwerts an einer Stelle, an der sich normalerweise ein Zellenwert befände, oder durch eine Folge aus Leerzeichen gekennzeichnet.

8. Da einige andere Produkte die Länge von Zeichenfeldern auf 254 oder 255 Byte begrenzen, gibt das Dienstprogramm EXPORT eine Warnung aus, wenn eine Zeichenspalte mit einer Länge von mehr als 254 Zeichen für den Export ausgewählt wird. Das Dienstprogramm IMPORT verarbeitet Felder, die so lang wie die längste LONG VARCHAR- und LONG VARGRAPHIC-Spalte sind.

Aspekte beim Versetzen von XML-Daten - Begrenzer:

Beim Versetzen von ASCII-Dateien (DEL) muss sichergestellt werden, dass die Daten beim Versetzen nicht versehentlich durch Fehler bei der Erkennung von Begrenzerzeichen geändert werden. Um derartige Fehler zu vermeiden, gibt DB2 verschiedene einschränkende Bedingungen vor und stellt eine Anzahl von Änderungswerten für den Dateityp bereit.

Einschränkungen für Begrenzer

Verschiedene Einschränkungen verhindern, dass das gewählte Begrenzerzeichen als Teil der zu versetzenden Daten behandelt wird. Zunächst gilt, dass Begrenzer sich gegenseitig ausschließen müssen. Zweitens dürfen Begrenzer keine binäre Nullen, Zeilenvorschubzeichen, Rücklaufzeichen oder Leerzeichen sein. Auch das Standarddezimalzeichen (.) darf nicht als Zeichenfolgebegrenzer verwendet werden. Und abschließend wäre zu nennen, dass das Pipe-Zeichen (`|`) in einer DBCS-Umgebung nicht als Zeichenbegrenzer unterstützt wird.

Die folgenden Zeichen werden durch eine ASCII- und EBCDIC-Codepage anders angegeben:

- Das DBCS-Endezeichen (0x0F) und das DBCS-Startzeichen (0x0E) können nicht als Begrenzer für eine EBCDIC-MBCS-Datendatei verwendet werden.
- Begrenzer für MBCS-, EUC- oder DBCS-Codepages dürfen nicht größer sein als 0x40. Hiervon ausgenommen ist das Standarddezimalzeichen für EBCDIC-MBCS-Daten (0x4b).
- Standardbegrenzer für Datendateien in ASCII-Codepages oder EBCDIC-MBCS-Codepages sind:
  - Zeichenfolgebegrenzer: " (0x22, doppeltes Anführungszeichen)
  - Spaltenbegrenzer: **,** (0x2c, Komma)
- Standardbegrenzer für Datendateien in EBCDIC-SBCS-Codepages sind:
  - Zeichenfolgebegrenzer: " (0x7F, doppeltes Anführungszeichen)
  - Spaltenbegrenzer: **,** (0x6B, Komma)
- Das Standarddezimalzeichen für ASCII-Datendateien ist 0x2e (Punkt).
- Das Standarddezimalzeichen für EBCDIC-Datendateien ist 0x4B (Punkt).
- Wenn sich die Codepage des Servers von der des Clients unterscheidet, sollte die hexadezimale Darstellung von Begrenzern, die von den Standardwerten abweichen, angegeben werden. Beispiel:

```
db2 load from ... modified by chardel0x0C coldelX1e ...
```

Probleme mit Begrenzern beim Versetzen von Daten

Doppelte Zeichenbegrenzer

Standardmäßig werden bei zeichenbasierten Feldern in DEL-Dateien alle Vorkommen von in einem Feld gefundenen Zeichenbegrenzern durch doppelte Zeichenbegrenzer dargestellt. Wird z. B. das doppelte Anführungszeichen als Zeichenbegrenzer verwendet, wird der exportierte Text `I am 6" tall.` im Ausgabetext in der DEL-Datei folgendermaßen wiedergegeben: `"I`

am 6"" tall.". Auf der anderen Seite gilt, dass der Eingabetext in einer DEL-Datei "What a ""nice"" day!" als What a "nice" day! importiert wird.

nodoubledel
Das Verhalten bei doppelten Zeichenbegrenzern kann für die Dienstprogramme IMPORT, EXPORT und LOAD durch Angabe des Änderungswerts für den Dateityp nodoubledel inaktiviert werden. Dabei ist jedoch zu bedenken, dass das Verhalten bei doppelten Zeichenbegrenzern eingesetzt wird, um Parsing-Fehler zu vermeiden. Wenn Sie den Änderungswert nodoubledel beim Export verwenden, werden vorhandene Zeichenbegrenzer in Zeichenfeldern nicht verdoppelt. Wenn Sie nodoubledel beim Import und beim Laden verwenden, werden doppelte Zeichenbegrenzer nicht als Literalvorkommen von Zeichenbegrenzern interpretiert.

nochardel
Wenn Sie den Änderungswert nochardel für den Dateityp beim Export verwenden, werden die Zeichenfelder nicht in Zeichenbegrenzer eingeschlossen. Wird nochardel beim Importieren oder Laden verwendet, werden die Zeichenbegrenzer nicht als Sonderzeichen, sondern als echte Daten interpretiert.

chardel
Es gibt weitere Änderungswerte für den Dateityp, die manuell angewendet werden können, um eine Verwechslung von Standardbegrenzern und Daten zu verhindern. Der Änderungswert chardel für den Dateityp gibt an, dass anstelle des Standardbegrenzers in Form von doppelten Anführungszeichen das einzelne Zeichen x als Zeichenfolgebegrenzer verwendet werden soll.

coldel
Gleichermaßen können Sie mit dem Änderungswert coldel, der angibt, dass das einzelne Zeichen x als Begrenzer für Spaltendaten verwendet werden soll, die standardmäßige Verwendung des Kommas als Spaltenbegrenzer verhindern.

delprioritychar
Ein weiterer beim Versetzen von DEL-Dateien zu beachtender Aspekt ist, dass der Vorrang der einzelnen Begrenzer eingehalten werden muss. Die Standardpriorität für Begrenzer lautet: Zeile, Zeichen, Spalte. Bei einigen Anwendungen ist jedoch folgende Priorität wichtig: Zeichen, Zeile, Spalte. Mit dieser Standardpriorität würde z. B. die DEL-Datendatei

```
"Vincent <zeilenbegrenzer> is a manager",<zeilenbegrenzer>
```

als eine Datei mit zwei Zeilen interpretiert (Vincent, und is a manager), da der <zeilenbegrenzer> Vorrang vor dem Zeichenbegrenzer (") hat. Bei Verwendung von delprioritychar hat der Zeichenbegrenzer (") Vorrang vor dem Zeilenbegrenzer (<zeilenbegrenzer>), sodass dieselbe DEL-Datei in diesem Fall richtig als Datei mit einer einzigen Zeile interpretiert wird: Vincent is a manager.

ASC-Dateiformat (ASCII-Format mit universellen Zeilenbegrenzern)

Das ASCII-Format mit universellen Zeilenbegrenzern, das die Dienstprogramme IMPORT und LOAD als ASC kennen, wird in zwei Varianten zur Verfügung gestellt: mit fester Länge und mit flexibler Länge. Beim ASC-Format mit fester Länge haben alle Datensätze eine feste Länge. Beim ASC-Format mit flexibler Länge sind die Datensätze durch einen Zeilenbegrenzer (immer ein Zeilenumbruch) begrenzt. Der Begriff *universell* bedeutet im Zusammenhang mit dem ASCII-Format mit universellen Zeilenbegrenzern, dass Spaltenwerte nicht durch Begrenzer voneinander getrennt werden.

Beim Importieren oder Laden von ASC-Daten wird durch den Änderungswert `reclen` für den Dateityp angegeben, dass die Datendatei im ASC-Format mit fester Länge vorliegt. Wird dieser Änderungswert nicht angegeben, bedeutet dies, dass die Datendatei im ASC-Format mit flexibler Länge vorliegt.

Das ASCII-Format mit universellen Zeilenbegrenzern kann für den Austausch von Daten mit beliebigen ASCII-Produkten verwendet werden, die Daten im Spaltenformat verarbeiten können (einschließlich Textverarbeitungsprogrammen). Jede ASC-Datei ist ein Datenstrom aus ASCII-Zeichen, der sich aus Datenwerten zusammensetzt, die in Zeilen und Spalten angeordnet sind. Die Zeilen innerhalb des Datenstroms werden durch Zeilenbegrenzungszeichen voneinander getrennt. Jede Spalte innerhalb einer Zeile wird durch ein Wertepaar bestehend aus Anfangsposition-Endposition (angegeben über **IMPORT**-Parameter) definiert. Jedes Paar stellt Speicherpositionen innerhalb einer Zeile dar, die als Bytepositionen angegeben werden. Die erste Position innerhalb einer Zeile ist die Byteposition 1. Das erste Element jedes Positionspaars ist das Byte in der Zeile, bei dem die Spalte beginnt, und das zweite Element ist das Byte, bei dem die Spalte endet. Die Spalten können sich möglicherweise überlappen. Jede Zeile in einer ASC-Datei hat die gleiche Spaltendefinition.

Eine ASC-Datei ist durch Folgendes definiert:

```
ASC-Datei ::= Zeile 1 Daten || Zeilenbegrenzer ||
                            Zeile 2 Daten || Zeilenbegrenzer ||
              .
              .
              .
                            Zeile n Daten

Zeile i Daten ::= ASCII-Zeichen || Zeilenbegrenzer

Zeilenbegrenzer ::= ASCII-Zeilenvorschubfolge[a]
```

- [a] Als Satzbegrenzer wird das Zeilenvorschubzeichen, ASCII-Wert x0A, angenommen. Unter einem Windows-Betriebssystem generierte Daten können den 2-Byte-Standardwert für Rücklauf/Zeilenvorschub, 0x0D0A, verwenden. Daten in EBCDIC-Codepages müssen das EBCDIC-Zeichen für Zeilenvorschub (0x25) als Satzbegrenzer verwenden (EBCDIC-Daten können geladen werden, indem der Änderungswert `codepage` für den Dateityp mit dem Befehl **LOAD** verwendet wird). Der Satzbegrenzer wird nie als Teil eines Feldes mit Daten gewertet.

ASC-Datentypbeschreibungen:

Tabelle 22. Gültige Datentypformate für das ASC-Dateiformat

Datentyp	Für das Dienstprogramm IMPORT zulässiges Format
BIGINT	Eine Konstante eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich von -9 223 372 036 854 775 808 bis 9 223 372 036 854 775 807 liegen. Dezimalzahlen werden ganzzahlig abgeschnitten. Ein Komma, Punkt oder Doppelpunkt gilt als Dezimalzeichen. Tausendertrennzeichen sind nicht zulässig. Die Anfangs- und Endposition müssen ein Feld festlegen, dessen Breite 50 Byte nicht überschreitet. Ganze Zahlen, Dezimalzahlen und die Mantissen von Gleitkommazahlen dürfen nicht mehr als 31 Stellen haben. Die Exponenten von Gleitkommazahlen dürfen nicht mehr als 3 Stellen haben.
BLOB/CLOB	Eine Zeichenfolge. Die Zeichenfolge wird nach Bedarf rechts abgeschnitten und so an die maximal zulässige Länge der Zielspalte angeglichen. Wenn die ASC-Option zum *Abschneiden von Leerzeichen* wirksam ist, werden abschließende Leerzeichen von der ursprünglichen oder der abgeschnittenen Zeichenfolge entfernt.
BLOB_FILE, CLOB_FILE, DBCLOB_FILE (nur DBCS)	Der begrenzte oder nicht begrenzte Name der Datei, die die Daten enthält.
CHAR	Eine Zeichenfolge. Die Zeichenfolge wird nach Bedarf rechts abgeschnitten oder mit Leerzeichen aufgefüllt und so an die Breite der Zielspalte angeglichen.
DATE	Eine Zeichenfolge, die einen Datumswert in einem Format enthält, das mit dem Gebietscode der Zieldatenbank konsistent ist. Die Anfangs- und Endposition müssen eine Feldbreite festlegen, die im Bereich der externen Darstellung eines Datums liegt.
DBCLOB (nur DBCS)	Eine Zeichenfolge mit einer geraden Anzahl Byte. Eine Zeichenfolge mit einer ungeraden Anzahl Byte ist ungültig und wird abgelehnt. Eine gültige Zeichenfolge wird nach Bedarf rechts abgeschnitten und so an die maximal zulässige Länge der Zielspalte angeglichen.
DECIMAL	Eine Konstante eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich der Datenbankspalte liegen, in die sie importiert werden. Wenn der Eingabewert mehr Ziffern nach dem Dezimalzeichen enthält, als in der Datenbankspalte Platz finden, werden die überzähligen Ziffern abgeschnitten. Ein Komma, Punkt oder Doppelpunkt gilt als Dezimalzeichen. Tausendertrennzeichen sind nicht zulässig. Die Anfangs- und Endposition müssen ein Feld festlegen, dessen Breite 50 Byte nicht überschreitet. Ganze Zahlen, Dezimalzahlen und die Mantissen von Gleitkommazahlen dürfen nicht mehr als 31 Stellen haben. Die Exponenten von Gleitkommazahlen dürfen nicht mehr als 3 Stellen haben.

Tabelle 22. Gültige Datentypformate für das ASC-Dateiformat (Forts.)

Datentyp	Für das Dienstprogramm IMPORT zulässiges Format
FLOAT(long)	Eine Konstante eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Alle Werte sind gültig. Ein Komma, Punkt oder Doppelpunkt gilt als Dezimalzeichen. Ein E wird unabhängig von der Groß-/Kleinschreibung als Anfang des Exponenten einer FLOAT-Konstante akzeptiert. Die Anfangs- und Endposition müssen ein Feld festlegen, dessen Breite 50 Byte nicht überschreitet. Ganze Zahlen, Dezimalzahlen und die Mantissen von Gleitkommazahlen dürfen nicht mehr als 31 Stellen haben. Die Exponenten von Gleitkommazahlen dürfen nicht mehr als 3 Stellen haben.
GRAPHIC (nur DBCS)	Eine Zeichenfolge mit einer geraden Anzahl Byte. Eine Zeichenfolge mit einer ungeraden Anzahl Byte ist ungültig und wird abgelehnt. Eine gültige Zeichenfolge wird nach Bedarf rechts abgeschnitten oder mit Doppelbyteleerzeichen (0x8140) aufgefüllt und so an die maximal zulässige Länge der Zielspalte angeglichen.
INTEGER	Eine Konstante eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich von -2 147 483 648 bis 2 147 483 647 liegen. Dezimalzahlen werden ganzzahlig abgeschnitten. Ein Komma, Punkt oder Doppelpunkt gilt als Dezimalzeichen. Tausendertrennzeichen sind nicht zulässig. Die Anfangs- und Endposition müssen ein Feld festlegen, dessen Breite 50 Byte nicht überschreitet. Ganze Zahlen, Dezimalzahlen und die Mantissen von Gleitkommazahlen dürfen nicht mehr als 31 Stellen haben. Die Exponenten von Gleitkommazahlen dürfen nicht mehr als 3 Stellen haben.
LONG VARCHAR	Eine Zeichenfolge. Die Zeichenfolge wird nach Bedarf rechts abgeschnitten und so an die maximal zulässige Länge der Zielspalte angeglichen. Wenn die ASC-Option zum *Abschneiden von Leerzeichen* wirksam ist, werden abschließende Leerzeichen von der ursprünglichen oder der abgeschnittenen Zeichenfolge entfernt.
LONG VARGRAPHIC (nur DBCS)	Eine Zeichenfolge mit einer geraden Anzahl Byte. Eine Zeichenfolge mit einer ungeraden Anzahl Byte ist ungültig und wird abgelehnt. Eine gültige Zeichenfolge wird nach Bedarf rechts abgeschnitten und so an die maximal zulässige Länge der Zielspalte angeglichen.
SMALLINT	Eine Konstante eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich von -32 768 bis 32 767 liegen. Dezimalzahlen werden ganzzahlig abgeschnitten. Ein Komma, Punkt oder Doppelpunkt gilt als Dezimalzeichen. Tausendertrennzeichen sind nicht zulässig. Die Anfangs- und Endposition müssen ein Feld festlegen, dessen Breite 50 Byte nicht überschreitet. Ganze Zahlen, Dezimalzahlen und die Mantissen von Gleitkommazahlen dürfen nicht mehr als 31 Stellen haben. Die Exponenten von Gleitkommazahlen dürfen nicht mehr als 3 Stellen haben.

Tabelle 22. Gültige Datentypformate für das ASC-Dateiformat (Forts.)

Datentyp	Für das Dienstprogramm IMPORT zulässiges Format
TIME	Eine Zeichenfolge, die einen Zeitwert in einem Format enthält, das mit dem Gebietscode der Zieldatenbank konsistent ist. Die Anfangs- und Endposition müssen eine Feldbreite festlegen, das im Bereich der externen Darstellung eines Zeitwerts liegt.
TIMESTAMP	Eine Zeichenfolge, die einen Zeitmarkenwert enthält, dessen Speicherung in einer Datenbank zulässig ist. Die Anfangs- und Endposition müssen eine Feldbreite festlegen, die im Bereich der externen Darstellung einer Zeitmarke liegt.
VARCHAR	Eine Zeichenfolge. Die Zeichenfolge wird nach Bedarf rechts abgeschnitten und so an die maximal zulässige Länge der Zielspalte angeglichen. Wenn die ASC-Option zum *Abschneiden von Leerzeichen* wirksam ist, werden abschließende Leerzeichen von der ursprünglichen oder der abgeschnittenen Zeichenfolge entfernt.
VARGRAPHIC (nur DBCS)	Eine Zeichenfolge mit einer geraden Anzahl Byte. Eine Zeichenfolge mit einer ungeraden Anzahl Byte ist ungültig und wird abgelehnt. Eine gültige Zeichenfolge wird nach Bedarf rechts abgeschnitten und so an die maximal zulässige Länge der Zielspalte angeglichen.

Beispiel für ASC-Datei:
Es folgt ein Beispiel für eine ASC-Datei. Jede Zeile endet mit einer Zeilenvorschubfolge (unter den Windows-Betriebssystemen endet jede Zeile mit einer Folge aus einem Rücklauf- und einem Zeilenvorschubzeichen).

```
Smith, Bob       4973     15.46
Jones, Suzanne   12345    16.34
Williams, Sam    452123  193.78
```

Anmerkung:

1. Bei ASC-Dateien wird davon ausgegangen, dass sie keine Spaltennamen enthalten.
2. Zeichenfolgen sind *nicht* in Begrenzer eingeschlossen. Der Datentyp einer Spalte in der ASC-Datei wird durch den Datentyp der Zielspalte in der Datenbanktabelle festgelegt.
3. NULL wird unter folgenden Bedingungen in einer Datenbankspalte, die Nullwerte enthalten darf, importiert:
   - Ein Feld mit Leerzeichen ist für eine numerische Datenbankspalte oder eine Datenbankspalte des Typs DATE, TIME oder TIMESTAMP bestimmt.
   - Es wird ein Feld ohne Anfangs- und Endpositionspaar angegeben.
   - Es wird ein Positionspaar mit Anfangs- und Endpositionen gleich null angegeben.
   - Eine Datenzeile ist zu kurz, um einen gültigen Wert für die Zielspalte zu enthalten.
   - Die LOAD-Option `NULL INDICATORS` wird verwendet, und die Nullanzeigerspalte enthält `N` (oder einen anderen vom Benutzer angegebenen Wert).
4. Wenn die Zielspalte keinen Nullwert enthalten darf, führt der Versuch, ein Feld mit Leerzeichen in eine numerische Spalte oder eine Spalte des Typs DATE, TIME oder TIMESTAMP zu importieren, zum Zurückweisen der Zeile.

5. Wenn die Eingabedaten nicht mit der Zielspalte kompatibel sind und diese Spalte Nullwerte enthalten darf, wird eine Null importiert oder die Zeile zurückgewiesen, je nachdem, wo der Fehler erkannt wird. Wenn die Spalte keine Nullwerte enthalten darf, wird die Zeile zurückgewiesen. Nachrichten mit Angaben zu den festgestellten Inkompatibilitäten werden in die Nachrichtendatei geschrieben.

PC-Version des IXF-Dateiformats

Die PC-Version des IXF-Dateiformats PC/IXF ist eine Anpassung der IXF-Datenaustauscharchitektur (Integration Exchange Format) für Datenbankmanager. Die IXF-Architektur ist speziell auf den Austausch von Strukturen und Daten relationaler Datenbanken ausgelegt. Die PC/IXF-Architektur ermöglicht es dem Datenbankmanager, eine Datenbank zu exportieren, ohne dass die Anforderungen und Eigenarten eines Zielprodukts bekannt sein müssen. Auch ein Produkt, in das eine PC/IXF-Datei importiert wird, muss nur die PC/IXF-Architektur beherrschen. Die Merkmale des Produkts, von dem die Datei exportiert wurde, sind nicht von Bedeutung. Die PC/IXF-Dateiarchitektur wahrt die Unabhängigkeit sowohl der exportierenden als auch der importierenden Datenbanksysteme.

Die IXF-Architektur stellt ein generisches Austauschformat für relationale Datenbanken dar, das eine Vielzahl von Typen relationaler Daten unterstützt, zu denen auch einige Typen zählen, die von bestimmten Produkten für relationale Datenbanken möglicherweise nicht unterstützt werden. Das PC/IXF-Dateiformat wahrt diese Flexibilität. Beispielsweise unterstützt die PC/IXF-Architektur sowohl SBCS-Datentypen (Einzelbytezeichenfolge) als auch DBCS-Datentypen (Doppelbytezeichenfolge). Nicht alle Implementierungen unterstützen alle PC/IXF-Datentypen. Selbst eingeschränkte Implementierungen erlauben jedoch die Erkennung und Disposition von nicht unterstützten Datentypen beim Import.

Im Allgemeinen besteht eine PC/IXF-Datei aus einer ununterbrochenen Folge von Datensätzen mit variabler Länge. Die Datei enthält die folgenden Datensatztypen in der gezeigten Reihenfolge:

- Einen Kopfdatensatz der Satzart H
- Einen Tabellensatz der Satzart T
- Mehrere Spaltendeskriptorsätze der Satzart C (ein Datensatz für jede Spalte in der Tabelle)
- Mehrere Datensätze der Satzart D (jede Zeile in der Tabelle wird durch einen oder mehrere D-Datensätze dargestellt)

Eine PC/IXF-Datei kann an einer beliebigen Stelle hinter dem H-Satz auch Anwendungssätze vom Satztyp A enthalten. Diese Datensätze sind in PC/IXF-Dateien zulässig, sodass eine Anwendung zusätzliche Daten, die im PC/IXF-Format nicht definiert sind, in eine PC/IXF-Datei einfügen kann. A-Datensätze werden von jedem Programm ignoriert, das eine PC/IXF-Datei liest und keine Informationen zum Datenformat und Inhalt besitzt, das bzw. der durch die Anwendungskennung im A-Datensatz impliziert wird.

Jeder Datensatz in einer PC/IXF-Datei beginnt mit einem Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große rechtsbündige Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Programme, die PC/IXF-Dateien lesen, müssen diese Satzlängen verwenden, um das Ende des aktuellen Datensatzes und den Anfang des nächsten Datensatzes zu bestimmen. H-, T- und C-Datensätze müssen groß genug für alle definierten Felder sein, und die Satzlängenfelder müssen selbstverständlich mit den Ist-Längen

übereinstimmen. Wenn jedoch zusätzliche Daten (z. B. ein *neues* Feld) am Ende eines dieser Datensätze angefügt werden, müssen vorhandene Programme, die PC/IXF-Dateien lesen, die zusätzlichen Daten ignorieren und lediglich eine Warnung generieren. Programme hingegen, die PC/IXF-Dateien schreiben, müssen H-, T- und C-Datensätze mit genau der für alle definierten Felder erforderlichen Länge schreiben.

Wenn eine PC/IXF-Datei Spalten mit LOB-Positionskennungen (so genannte LLS-Spalten) enthält, muss jede LLS-Spalte einen eigenen D-Datensatz haben. D-Datensätze werden automatisch durch das Dienstprogramm EXPORT erstellt. Werden die PC/IXF-Dateien jedoch mithilfe eines Fremdherstellertools erstellt, müssen Sie diese Datensätze manuell erstellen. Des Weiteren wird für jede LOB-Spalte in einer Tabelle (auch für LOB-Spalten mit einem Nullwert) eine LOB-Positionskennung benötigt. Für eine LOB-Spalte mit einem Nullwert müssen Sie eine LOB-Positionskennung mit einem LOB-Nullwert erstellen.

Der Eintrag des D-Datensatzes für jede XML-Spalte enthält zwei Byte Little Endian-Daten zur Angabe der Länge der XML-Datenkennung (XDS), gefolgt von der XDS selbst.

Beispiel: Die XDS

```
XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

wird von den folgenden Byte in einem D-Datensatz dargestellt:

```
0x3D 0x00 XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

PC/IXF-Dateisätze bestehen aus Feldern, die Zeichendaten enthalten. Die Dienstprogramme IMPORT und EXPORT interpretieren diese Zeichendaten mithilfe der CPGID der Zieldatenbank, wobei es zwei Ausnahmen gibt:

- Das IXFADATA-Feld von A-Datensätzen.

  Die Codepage-Umgebung von Zeichendaten in einem IXFADATA-Feld wird von der Anwendung festgelegt, die einen bestimmten A-Datensatz erstellt und verarbeitet. Das bedeutet, dass die Umgebung je nach Implementierung variiert.
- Das IXFDCOLS-Feld von D-Datensätzen.

  Die Codepage-Umgebung von Zeichendaten in einem IXFDCOLS-Feld ist eine Funktion der im C-Datensatz enthaltenen Informationen. Dieser Datensatz definiert eine bestimmte Spalte und ihre Daten.

Numerische Felder in H-, T- und C-Datensätzen sowie im Präfixteil von D- und A-Datensätzen müssen rechtsbündige Einzelbytezeichendarstellungen von ganzzahligen Werten sein, die mit führenden Nullen oder Leerzeichen aufgefüllt sind. Nullwerte müssen mit mindestens einem (rechtsbündigen) Nullzeichen, nicht mit Leerzeichen, angegeben werden. Wenn eines dieser numerischen Felder, z. B. IXFCLENG, dessen Länge vom Datentyp impliziert wird, nicht verwendet wird, muss es mit Leerzeichen aufgefüllt werden. Diese numerischen Felder sind:

```
IXFHRECL, IXFTRECL, IXFCRECL, IXFDRECL, IXFARECL,
IXFHHCNT, IXFHSBCP, IXFHDBCP, IXFTCCNT, IXFTNAML,
IXFCLENG, IXFCDRID, IXFCPOSN, IXFCNAML, IXFCTYPE,
IXFCSBCP, IXFCDBCP, IXFCNDIM, IXFCDSIZ, IXFDRID
```

Anmerkung: Das PC/IXF-Dateiformat des Datenbankmanagers ist nicht identisch mit dem Format von IBM System/370.

PC/IXF-Satztypen:
Es gibt fünf PC/IXF-Basissatztypen:

- Kopfdatensatz
- Tabellensatz
- Spaltendeskriptorsatz
- Datensatz
- Anwendungssatz

DB2 verwendet sieben Untertypen von Anwendungssätzen:
- Indexsatz
- Hierarchiesatz
- Satz für untergeordnete Tabelle
- Fortsetzungssatz
- Beendigungssatz
- Identitätssatz
- DB2 SQLCA

Jeder PC/IXF-Satztyp ist als eine Folge von Feldern definiert. Diese Felder sind erforderlich und müssen in der gezeigten Reihenfolge vorkommen.

KOPFDATEN

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFHRECL	06 BYTE	CHARACTER	Satzlänge
IXFHRECT	01 BYTE	CHARACTER	Satztyp = "H"
IXFHID	03 BYTE	CHARACTER	IXF-Kennung
IXFHVERS	04 BYTE	CHARACTER	IXF-Version
IXFHPROD	12 BYTE	CHARACTER	Produkt
IXFHDATE	08 BYTE	CHARACTER	Erstellungsdatum
IXFHTIME	06 BYTE	CHARACTER	Erstellungszeit
IXFHHCNT	05 BYTE	CHARACTER	Anzahl vorhergehender Datensätze
IXFHSBCP	05 BYTE	CHARACTER	Einzelbyte-Codepage
IXFHDBCP	05 BYTE	CHARACTER	Doppelbyte-Codepage
IXFHFIL1	02 BYTE	CHARACTER	reserviert

Die folgenden Felder sind in den Kopfdaten enthalten:

IXFHRECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der H-Datensatz muss lang genug sein für alle darin definierten Felder.

IXFHRECT
Der IXF-Satztyp, der für diesen Datensatz auf H gesetzt ist.

IXFHID
Die Dateiformatkennung, die für diese Datei auf IXF gesetzt ist.

IXFHVERS
Die Version des PC/IXF-Formats, die bei der Erstellung der Datei verwendet wurde, ist auf "0002" gesetzt.

IXFHPROD
Ein Feld, das von dem Programm, das die Datei erstellt, zu Identifikationszwecken verwendet werden kann. Wenn dieses Feld ausgefüllt wird, dienen die ersten sechs Byte dazu, das Produkt anzugeben, von dem die Datei erstellt wurde. Die letzten sechs Byte dienen dazu, die Version bzw. das

Release des Produkts anzugeben. Der Datenbankmanager verwendet dieses Feld, um das Vorhandensein von datenbankmanagerspezifischen Daten anzuzeigen.

IXFHDATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*.

IXFHTIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld ist wahlfrei und kann leergelassen werden.

IXFHHCNT
Die Anzahl der H-, T- und C-Datensätze in dieser Datei, die vor dem ersten Datensatz kommen. A-Datensätze sind bei diesem Wert nicht berücksichtigt.

IXFHSBCP
Einzelbyte-Codepage-Feld, das eine Einzelbytezeichendarstellung einer SBCS-CPGID oder "00000" enthält.

Das Dienstprogramm EXPORT setzt dieses Feld mit der SBCS-CPGID der exportierten Datenbanktabelle gleich. Beispiel: Wenn die SBCS-CPGID den Wert 850 hat, enthält dieses Feld "00850".

IXFHDBCP
Doppelbyte-Codepage-Feld, das eine Einzelbytezeichendarstellung einer DBCS-CPGID oder "00000" enthält.

Das Dienstprogramm EXPORT setzt dieses Feld mit der DBCS-CPGID der exportierten Datenbanktabelle gleich. Beispiel: Wenn die DBCS-CPGID den Wert 301 hat, enthält dieses Feld "00301".

IXFHFIL1
Zusatzfeld, das zwei Leerzeichen enthält. Dient als Entsprechung zu einem reservierten Feld in Host-IXF-Dateien.

TABELLENSATZ

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFTRECL	006 BYTE	CHARACTER	Satzlänge
IXFTRECT	001 BYTE	CHARACTER	Satztyp = "T"
IXFTNAML	003 BYTE	CHARACTER	Namenslänge
IXFTNAME	256 BYTE	CHARACTER	Name der Daten
IXFTQULL	003 BYTE	CHARACTER	Länge Qualifikationsmerkmal
IXFTQUAL	256 BYTE	CHARACTER	Qualifikationsmerkmal
IXFTSRC	012 BYTE	CHARACTER	Datenquelle
IXFTDATA	001 BYTE	CHARACTER	Datenkonvention = "C"
IXFTFORM	001 BYTE	CHARACTER	Datenformat = "M"
IXFTMFRM	005 BYTE	CHARACTER	Maschinenformat = "PC"
IXFTLOC	001 BYTE	CHARACTER	Datenspeicherposition = "I"
IXFTCCNT	005 BYTE	CHARACTER	"C"-Satzzählung
IXFTFIL1	002 BYTE	CHARACTER	reserviert
IXFTDESC	030 BYTE	CHARACTER	Datenbeschreibung
IXFTPKNM	257 BYTE	CHARACTER	Primärschlüsselname
IXFTDSPC	257 BYTE	CHARACTER	reserviert
IXFTISPC	257 BYTE	CHARACTER	reserviert
IXFTLSPC	257 BYTE	CHARACTER	reserviert

Die folgenden Felder sind im Tabellensatz enthalten:

IXFTRECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/

IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der T-Datensatz muss lang genug sein für alle darin definierten Felder.

IXFTRECT
Der IXF-Satztyp, der für diesen Datensatz auf T gesetzt ist.

IXFTNAML
Die Länge des Tabellennamens im IXFTNAME-Feld in Byte.

IXFTNAME
Der Name der Tabelle. Wenn jede Datei nur eine Tabelle enthält, dient dieses Feld nur zur Information. Dieses Feld wird vom Datenbankmanager beim Importieren von Daten nicht verwendet. Beim Schreiben einer PC/IXF-Datei schreibt der Datenbankmanager den DOS-Dateinamen (und möglicherweise Pfadinformationen) in dieses Feld.

IXFTQULL
Die Länge des Qualifikationsmerkmals für den Tabellennamen im IXFTQUAL-Feld in Byte.

IXFTQUAL
Qualifikationsmerkmal für den Tabellennamen, das den Ersteller einer Tabelle in einem relationalen System angibt. Dieses Feld dient nur zur Information. Wenn ein Programm, das eine Datei schreibt, keine Daten in dieses Feld schreiben muss, wird es vorzugsweise mit Leerzeichen gefüllt. Programme, die eine Datei lesen, können dieses Feld möglicherweise drucken oder anzeigen oder es in einem Informationsfeld speichern. Vom Inhalt dieses Feldes dürfen jedoch keine Berechnungen abhängen.

IXFTSRC
Wird verwendet, um die ursprüngliche Quelle der Daten anzugeben. Dieses Feld dient nur zur Information. Wenn ein Programm, das eine Datei schreibt, keine Daten in dieses Feld schreiben muss, wird es vorzugsweise mit Leerzeichen gefüllt. Programme, die eine Datei lesen, können dieses Feld möglicherweise drucken oder anzeigen oder es in einem Informationsfeld speichern. Vom Inhalt dieses Feldes dürfen jedoch keine Berechnungen abhängen.

IXFTDATA
Konvention, die zum Beschreiben der Daten verwendet wird. Dieses Feld muss für den Import und Export auf `C` gesetzt sein, um anzugeben, dass einzelne Spaltenattribute in den folgenden Spaltendeskriptorsätzen (C) beschrieben werden und dass Daten den PC/IXF-Konventionen entsprechen.

IXFTFORM
Konvention, die zum Speichern von numerischen Daten verwendet wird. Dieses Feld muss auf `M` gesetzt sein, um anzugeben, dass numerische Daten in den Datensätzen (D) in dem (internen) Maschinenformat gespeichert sind, das im IXFTMFRM-Feld angegeben ist.

IXFTMFRM
Das Format beliebiger Maschinendaten in der PC/IXF-Datei. Der Datenbankmanager liest oder schreibt Dateien nur, wenn dieses Feld auf `PC`*`bbb`* gesetzt ist, wobei *b* ein Leerzeichen ist und PC angibt, dass Daten in der PC/IXF-Datei in einem IBM PC-Maschinenformat vorliegen.

IXFTLOC
Die Speicherposition der Daten. Der Datenbankmanager unterstützt nur den Wert I, was bedeutet, dass sich die Daten in dieser Datei befinden.

IXFTCCNT
Die Anzahl von C-Datensätzen in dieser Tabelle. Hierbei handelt es sich um eine rechtsbündige Zeichendarstellung eines ganzzahligen Wertes.

IXFTFIL1
Zusatzfeld, das zwei Leerzeichen enthält. Dient als Entsprechung zu einem reservierten Feld in Host-IXF-Dateien.

IXFTDESC
Beschreibende Daten zur Tabelle. Dieses Feld dient nur zur Information. Wenn ein Programm, das eine Datei schreibt, keine Daten in dieses Feld schreiben muss, wird es vorzugsweise mit Leerzeichen gefüllt. Programme, die eine Datei lesen, können dieses Feld möglicherweise drucken oder anzeigen oder es in einem Informationsfeld speichern. Vom Inhalt dieses Feldes dürfen jedoch keine Berechnungen abhängen. Dieses Feld enthält `NOT NULL WITH DEFAULT`, wenn für die Spalte `NOT NULL WITH DEFAULT` angegeben war und der Tabellenname nicht aus einer Workstation-Datenbank stammt.

IXFTPKNM
Der Name des für die Tabelle definierten Primärschlüssels (sofern vorhanden). Der Name wird in Form einer auf null endenden Zeichenfolge gespeichert.

IXFTDSPC
Dieses Feld ist zur zukünftigen Verwendung reserviert.

IXFTISPC
Dieses Feld ist zur zukünftigen Verwendung reserviert.

IXFTLSPC
Dieses Feld ist zur zukünftigen Verwendung reserviert.

`SPALTENDESKRIPTORSATZ`

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFCRECL	006 BYTE	CHARACTER	Satzlänge
IXFTRECT	001 BYTE	CHARACTER	Satztyp = "C"
IXFCNAML	003 BYTE	CHARACTER	Länge des Spaltennamens
IXFCNAME	256 BYTE	CHARACTER	Spaltenname
IXFCNULL	001 BYTE	CHARACTER	Spalte lässt Nullen zu
IXFCDEF	001 BYTE	CHARACTER	Spalte hat Standardwerte
IXFCSLCT	001 BYTE	CHARACTER	Spaltenauswahlmarkierung
IXFCKPOS	002 BYTE	CHARACTER	Position im Primärschlüssel
IXFCCLAS	001 BYTE	CHARACTER	Datenklasse
IXFCTYPE	003 BYTE	CHARACTER	Datentyp
IXFCSBCP	005 BYTE	CHARACTER	Einzelbyte-Codepage
IXFCDBCP	005 BYTE	CHARACTER	Doppelbyte-Codepage
IXFCLENG	005 BYTE	CHARACTER	Spaltendatenlänge
IXFCDRID	003 BYTE	CHARACTER	Satzkennung "D"
IXFCPOSN	006 BYTE	CHARACTER	Spaltenposition
IXFCDESC	030 BYTE	CHARACTER	Spaltenbeschreibung
IXFCLOBL	020 BYTE	CHARACTER	Länge der LOB-Spalte
IXFCUDTL	003 BYTE	CHARACTER	Länge des UDT-Namens
IXFCUDTN	256 BYTE	CHARACTER	UDT-Name
IXFCDEFL	003 BYTE	CHARACTER	Länge des Standardwerts
IXFCDEFV	254 BYTE	CHARACTER	Standardwert
IXFCREF	001 BYTE	CHARACTER	Verweistyp
IXFCNDIM	002 BYTE	CHARACTER	Anzahl Dimensionen
IXFCDSIZ	variabel	CHARACTER	Größe jeder Dimension

Die folgenden Felder sind in Spaltendeskriptorsätzen enthalten:

IXFCRECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/

IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der C-Datensatz muss lang genug sein für alle darin definierten Felder.

IXFCRECT
Der IXF-Satztyp, der für diesen Datensatz auf C gesetzt ist.

IXFCNAML
Die Länge des Spaltennamens im IXFTNAME-Feld in Byte.

IXFCNAME
Der Name der Spalte.

IXFCNULL
Gibt an, ob Nullen in dieser Spalte zulässig sind. Gültige Werte sind `Y` oder `N`.

IXFCDEF
Gibt an, ob für dieses Feld ein Standardwert definiert ist. Gültige Werte sind `Y` oder `N`.

IXFCSLCT
Ein veraltetes Feld, das die Auswahl einer Untermenge von Spalten in der Tabelle ermöglichen sollte. Programme, die PC/IXF-Dateien schreiben, müssen in diesem Feld stets Y speichern. Programme, die PC/IXF-Dateien lesen, müssen dieses Feld ignorieren.

IXFCKPOS
Die Position der Spalte als Bestandteil des Primärschlüssels. Gültige Werte sind `01` bis `16` bzw. `N`, falls die Spalte nicht Bestandteil des Primärschlüssels ist.

IXFCCLAS
Die Klasse der Datentypen, die im IXFCTYPE-Feld verwendet werden sollen. Der Datenbankmanager unterstützt nur relationale Typen (`R`).

IXFCTYPE
Der Datentyp der Spalte.

IXFCSBCP
Enthält eine Einzelbytezeichendarstellung einer SBCS-CPGID. Dieses Feld legt die CPGID für Einzelbytezeichendaten fest, die im IXFDCOLS-Feld der D-Datensätze für diese Spalte vorkommen.

Die Semantik dieses Feldes variiert mit dem Datentyp für die Spalte (im IXFCTYPE-Feld angegeben).

- Für eine Zeichenfolgespalte sollte dieses Feld normalerweise einen Wert ungleich null enthalten, der mit dem Wert des IXFHSBCP-Feldes im H-Datensatz übereinstimmt. Andere Werte sind jedoch erlaubt. Wenn dieser Wert null ist, wird angenommen, dass die Spalte Bitfolgendaten enthält.
- Für eine numerische Spalte hat dieses Feld keine Bedeutung. Es wird vom Dienstprogramm EXPORT auf null gesetzt und vom Dienstprogramm IMPORT ignoriert.
- Für eine Datums- oder Uhrzeitspalte hat dieses Feld keine Bedeutung. Es wird vom Dienstprogramm EXPORT auf den Wert des Feldes IXFHSBCP gesetzt und vom Dienstprogramm IMPORT ignoriert.
- Für eine Grafikspalte muss dieses Feld null sein.

IXFCDBCP
Enthält eine Einzelbytezeichendarstellung einer DBCS-CPGID. Dieses Feld

legt die CPGID für Doppelbytezeichendaten fest, die im IXFDCOLS-Feld der D-Datensätze für diese Spalte vorkommen.

Die Semantik dieses Feldes variiert mit dem Datentyp für die Spalte (im IXFCTYPE-Feld angegeben).

- Für eine Zeichenfolgespalte muss dieses Feld null sein oder einen Wert enthalten, der mit dem Wert des IXFHDBCP-Feldes im H-Datensatz übereinstimmt. Andere Werte sind jedoch erlaubt. Wenn der Wert im IXFCSBCP-Feld null ist, muss der Wert in diesem Feld null sein.
- Für eine numerische Spalte hat dieses Feld keine Bedeutung. Es wird vom Dienstprogramm EXPORT auf null gesetzt und vom Dienstprogramm IMPORT ignoriert.
- Für eine Datums- oder Uhrzeitspalte hat dieses Feld keine Bedeutung. Es wird vom Dienstprogramm EXPORT auf null gesetzt und vom Dienstprogramm IMPORT ignoriert.
- Für eine Grafikspalte muss dieses Feld einen Wert haben, der mit dem Wert des IXFHDBCP-Feldes identisch ist.

IXFCLENG

Bietet Informationen zur Größe der beschriebenen Spalte. Für einige Datentypen wird dieses Feld nicht verwendet und muss Leerzeichen enthalten. Für andere Datentypen enthält dieses Feld die rechtsbündige Zeichendarstellung einer ganzen Zahl, die die Spaltenlänge festlegt. Für wieder andere Datentypen ist dieses Feld in zwei Teilfelder unterteilt: 3 Byte für die Genauigkeit und 2 Byte für die Anzahl der Kommastellen. Beide Teilfelder sind rechtsbündige Zeichendarstellungen von ganzen Zahlen.Ab Version 9.7 enthält dieses Feld beim Datentyp für Zeitmarken (TIMESTAMP) die rechtsbündige Zeichendarstellung einer ganzen Zahl (Integer), die die Genauigkeit der Zeitmarke angibt.

IXFCDRID

Die D-Satzkennung. Dieses Feld enthält die rechtsbündige Zeichendarstellung eines ganzzahligen Wertes. Es ist möglich, mehrere D-Datensätze für die einzelnen Datenzeilen in der PC/IXF-Datei zu verwenden. Dieses Feld gibt an, welcher D-Datensatz (einer von mehreren D-Datensätzen, die eine Datenzeile bilden) die Daten für die Spalte enthält. Der Wert Eins (z. B. `001`) gibt an, dass sich die Daten für eine Spalte im ersten D-Datensatz in einer Datenzeile befinden. Der erste C-Datensatz muss den IXFCDRID-Wert Eins haben. Alle nachfolgenden C-Datensätze müssen einen IXFCDRID-Wert haben, der mit dem Wert des vorherigen C-Datensatzes übereinstimmt oder um Eins größer ist.

IXFCPOSN

Der Wert dieses Feldes wird verwendet, um die Daten für die Spalte innerhalb eines der D-Datensätze zu lokalisieren, die eine Zeile der Tabellendaten darstellen. Es handelt sich um den Anfangspunkt der Daten für diese Spalte innerhalb des IXFDCOLS-Feldes des D-Datensatzes. Wenn die Spalte Nullwerte enthalten darf, zeigt IXFCPOSN auf den Nullanzeiger, andernfalls auf die Daten selbst. Wenn eine Spalte Daten mit variabler Länge enthält, beginnen die Daten selbst mit dem aktuellen Längenanzeiger. Der IXFCPOSN-Wert für das erste Byte im IXFDCOLS-Feld des D-Datensatzes ist eins (nicht null). Wenn sich eine Spalte in einem neuen D-Datensatz befindet, muss der Wert von IXFCPOSN eins sein. Andernfalls müssen die IXFCPOSN-Werte von Spalte zu Spalte so zunehmen, dass sich die Datenwerte nicht überschneiden.

IXFCDESC
Beschreibende Informationen zu der Spalte. Dieses Feld dient nur zur Information. Wenn ein Programm, das in eine Datei schreibt, keine Daten in dieses Feld schreiben muss, wird es vorzugsweise mit Leerzeichen gefüllt. Programme, die eine Datei lesen, können dieses Feld möglicherweise drucken oder anzeigen oder es in einem Informationsfeld speichern. Vom Inhalt dieses Feldes dürfen jedoch keine Berechnungen abhängen.

IXFCLOBL
Die Länge des in dieser Spalte definierten großen Objekts (LOB) in Byte. Handelt es sich nicht um eine LOB-Spalte, lautet der Wert in diesem Feld `000`.

IXFCUDTL
Die Länge des Namens des benutzerdefinierten Typs (UDT) im IXFCUDTN-Feld in Byte. Handelt es sich nicht um eine Spalte des Typs UDT, hat dieses Feld den Wert `000`.

IXFCUDTN
Der Name des benutzerdefinierten Typs, der als Datentyp für diese Spalte verwendet wird.

IXFCDEFL
Die Länge des Standardwerts im IXFCDEFV-Feld in Byte. Wenn diese Spalte keinen Standardwert hat, lautet der Wert in diesem Feld `000`.

IXFCDEFV
Gibt den Standardwert für diese Spalte an, sofern ein solcher definiert wurde.

IXFCREF
Wenn die Spalte Teil einer Hierarchie ist, gibt dieses Feld an, ob es sich bei der Spalte um eine Datenspalte (`D`) oder um eine Referenzspalte (`R`) handelt.

IXFCNDIM
Die Anzahl der Dimensionen in der Spalte. Matrizen werden in dieser Version von PC/IXF nicht unterstützt. Dieses Feld muss daher eine Zeichendarstellung eines ganzzahligen Nullwerts enthalten.

IXFCDSIZ
Die Größe bzw. der Bereich jeder Dimension. Die Länge dieses Feldes beträgt 5 Byte pro Dimension. Da Matrizen nicht unterstützt werden (d. h. die Anzahl der Dimensionen muss null sein), hat dieses Feld die Länge null und ist eigentlich gar nicht vorhanden.

```
DATENSATZ

     FELDNAME       LÄNGE     TYP         KOMMENTARE
  ----------     -------   ---------   -------------
     IXFDRECL       06 BYTE   CHARACTER   Satzlänge
     IXFDRECT       01 BYTE   CHARACTER   Satztyp = "D"
     IXFDRID        03 BYTE   CHARACTER   Satzkennung "D"
     IXFDFIL1       04 BYTE   CHARACTER   reserviert
     IXFDCOLS       variabel  variabel    Spaltendaten
```

Die folgenden Felder sind in Datensätzen enthalten:

IXFDRECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Jeder D-Datensatz muss lang genug

für alle signifikanten Daten für das aktuelle Vorkommen der letzten im Datensatz gespeicherten Datenspalte sein.

IXFDRECT
Der IXF-Satztyp, der für diesen Datensatz auf D gesetzt ist, um anzugeben, dass der Datensatz Datenwerte für die Tabelle enthält.

IXFDRID
Die Satzkennung, die einen bestimmten D-Datensatz in der Folge von mehreren D-Datensätzen angibt, die eine Datenzeile bilden. Für den ersten D-Datensatz in einer Datenzeile enthält dieses Feld den Wert Eins. Für den zweiten D-Datensatz in einer Datenzeile enthält dieses Feld den Wert Zwei usw. In jeder Datenzeile müssen alle D-Satzkennungen, die in den C-Datensätzen gesendet werden, auch vorhanden sein.

IXFDFIL1
Zusatzfeld, das vier Leerzeichen enthält. Dienst als Entsprechung zu reservierten Feldern in Host-IXF-Dateien und bietet Platz für ein mögliches DBCS-Startzeichen.

IXFDCOLS
Der Bereich für Spaltendaten. Der Datenbereich eines Datensatzes (D-Satzes) besteht aus mindestens einem Spalteneintrag. Es gibt einen Spalteneintrag für jeden Spaltendeskriptorsatz, der die gleiche D-Satzkennung hat wie der D-Satz. Im D-Satz wird die Anfangsposition der Spalteneinträge durch den IXFCPOSN-Wert in C-Sätzen angegeben.

Das Format der Spalteneingabedaten hängt davon ab, ob die Spalte Nullwerte enthalten darf:

- Wenn die Spalte Nullwerte enthalten darf (das IXFCNULL-Feld auf `Y` gesetzt ist), enthalten die Spalteneingabedaten einen Nullanzeiger. Wenn die Spalte nicht null ist, folgen auf den Anzeiger datentypspezifische Informationen, einschließlich des eigentlichen Datenbankwertes. Der Nullanzeiger ist ein 2 Byte großer Wert, der für 'nicht null' auf x'0000' und für 'null' auf x'FFFF' gesetzt wird.
- Wenn die Spalte keine Nullwerte enthalten darf, enthalten die Spalteneingabedaten nur datentypspezifische Informationen, einschließlich des tatsächlichen Datenbankwertes.

Für Datentypen mit variabler Länge schließen die datentypspezifischen Informationen einen aktuellen Längenanzeiger ein. Die aktuellen Längenanzeiger sind 2 Byte große ganze Zahlen in einem Format, das vom IXFTMFRM-Feld festgelegt wird.

Die Länge des Datenbereichs eines D-Datensatzes darf 32.771 Byte nicht überschreiten.

ANWENDUNGSSATZ

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	06 BYTE	CHARACTER	Satzlänge
IXFARECT	01 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	12 BYTE	CHARACTER	Anwendungskennzeichen
IXFADATA	variabel	variabel	anwendungsspezifische Daten

Die folgenden Felder sind in Anwendungssätzen enthalten:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/

IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der A-Datensatz muss lang genug sein für das gesamte IXFAPPID-Feld.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben, dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das die Anwendung angibt, die den A-Datensatz erstellt. Vom Datenbankmanager erstellte PC/IXF-Dateien können A-Datensätze enthalten, in denen die ersten sechs Zeichen dieses Feldes auf eine Konstante gesetzt sind, die den Datenbankmanager angibt, während die letzten 6 Zeichen das Release oder die Version des Datenbankmanagers oder einer anderen Anwendung angeben, der bzw. die den A-Datensatz schreibt.

IXFADATA
Dieses Feld enthält anwendungsabhängige Ergänzungsdaten, deren Format und Inhalt nur dem Programm bekannt ist, das den A-Datensatz erstellt, sowie anderen Anwendungen, die den A-Datensatz wahrscheinlich verarbeiten.

DB2-INDEXSATZ

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	006 BYTE	CHARACTER	Satzlänge
IXFARECT	001 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	012-BYTE	CHARACTER	Anwendungskennzeichen = 'DB2 02.00'
IXFAITYP	001-BYTE	CHARACTER	Anwendungsspezifischer Datentyp = 'I'
IXFADATE	008 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Daten
IXFATIME	006 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Zeit
IXFANDXL	002 BYTE	SHORT INT	Länge des Indexnamens
IXFANDXN	256 BYTE	CHARACTER	Name des Indexes
IXFANCL	002 BYTE	SHORT INT	Länge des Namens des Indexerstellers
IXFANCN	256 BYTE	CHARACTER	Name des Indexerstellers
IXFATABL	002 BYTE	SHORT INT	Länge des Tabellennamens
IXFATABN	256 BYTE	CHARACTER	Tabellenname
IXFATCL	002 BYTE	SHORT INT	Länge des Namens des Tabellenerstellers
IXFATCN	256 BYTE	CHARACTER	Name des Tabellenerstellers
IXFAUNIQ	001 BYTE	CHARACTER	Eindeutige Regel
IXFACCNT	002-BYTE	SHORT INT	Spaltenanzahl
IXFAREVS	001 BYTE	CHARACTER	Markierung für Rückwärtsdurchsuchen zulassen
IXFAIDXT	001-BYTE	CHARACTER	Indextyp
IXFAPCTF	002 BYTE	CHARACTER	Wert für PCTFREE
IXFAPCTU	002 BYTE	CHARACTER	Wert für MINPCTUSED (Prozentsatz des auf Indexseiten freizuhaltenden Minimums an verwendetem Speicherplatz)
IXFAEXTI	001 BYTE	CHARACTER	reserviert
IXFACNML	006-BYTE	SHORT INT	Länge des Namens der Spalten
IXFACOLN	variabel	CHARACTER	Name der Spalten im Index

Für jeden benutzerdefinierten Index wird 1 Datensatz diesen Typs angegeben. Die Position dieses Datensatzes befindet sich nach allen C-Datensätzen für die Tabelle. DB2-Indexsätze enthalten die folgenden Felder:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der A-Datensatz muss lang genug sein für das gesamte IXFAPPID-Feld.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben,

dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das DB2 als die Anwendung angibt, von der dieser A-Datensatz erstellt wird.

IXFAITYP
Gibt an, dass dies der Subtyp "I" der DB2-Anwendungssätze ist.

IXFADATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*. Dieses Feld muss denselben Wert wie das IXFHDATE-Feld aufweisen.

IXFATIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld muss denselben Wert wie das IXFHTIME-Feld aufweisen.

IXFANDXL
Die Länge des Indexnamens im IXFANDXN-Feld in Byte.

IXFANDXN
Der Name des Indexes.

IXFANCL
Die Länge des Namens des Indexerstellers im IXFANCN-Feld in Byte.

IXFANCN
Der Name des Indexerstellers.

IXFATABL
Die Länge des Tabellennamens im IXFATABN-Feld in Byte.

IXFATABN
Der Name der Tabelle.

IXFATCL
Die Länge des Namens des Tabellenerstellers im IXFATCN-Feld in Byte.

IXFATCN
Der Name des Tabellenerstellers.

IXFAUNIQ
Gibt den Indextyp an. Gültige Werte sind `P` (= Primärschlüssel), `U` (= eindeutiger Index) und `D` (= nicht eindeutiger Index).

IXFACCNT
Gibt die Anzahl der Spalten in der Indexdefinition an.

IXFAREVS
Gibt an, ob das Rückwärtsdurchsuchen dieses Indexes zulässig ist. Gültige Werte sind `Y` (= zulässig) und `N` (= nicht zulässig).

IXFAIDXT
Gibt den Indextyp an. Gültige Werte sind `R` für einen regulären Index und `C` für einen Clusterindex.

IXFAPCTF
Gibt den Prozentsatz der freizuhaltenden Indexseiten an. Gültige Werte liegen zwischen `-1` und `99`. Wenn der Wert `-1` oder null angegeben wird, wird der Systemstandardwert verwendet.

IXFAPCTU
Gibt den Mindestprozentsatz für den Speicherplatz an, der auf Indexseiten

freigehalten werden muss, bevor zwei Indexseiten zusammengeführt werden können. Gültige Werte liegen zwischen 00 und 99.

IXFAEXTI
Zur zukünftigen Verwendung reserviert.

IXFACNML
Die Länge der Spaltennamen im IXFACOLN-Feld in Byte.

IXFACOLN
Die Namen der Spalten, die Bestandteil dieses Indexes sind. Gültige Werte haben das Format *+name-name...*. Hierbei gibt + eine aufsteigende Sortierung der Spalte und - eine absteigende Sortierung der Spalte an.

DB2-HIERARCHIESATZ

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	006 BYTE	CHARACTER	Satzlänge
IXFARECT	001 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	012-BYTE	CHARACTER	Anwendungskennzeichen = 'DB2 02.00'
IXFAXTYP	001-BYTE	CHARACTER	Anwendungsspezifischer Datentyp = 'X'
IXFADATE	008 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Daten
IXFATIME	006 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Zeit
IXFAYCNT	010 BYTE	CHARACTER	"Y"-Satzzählung für diese Hierarchie
IXFAYSTR	010 BYTE	CHARACTER	Anfangsspalte dieser Hierarchie

Zur Beschreibung einer Hierarchie wird 1 Datensatz diesen Typs verwendet. Alle Sätze für untergeordnete Tabellen (siehe nachfolgende Liste) müssen unmittelbar auf den Hierarchiesatz folgen. Die Hierarchiesätze müssen auf die C-Datensätze für die Tabelle folgen. DB2-Hierarchiesätze enthalten die folgenden Felder:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der A-Datensatz muss lang genug sein für das gesamte IXFAPPID-Feld.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben, dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das DB2 als die Anwendung angibt, von der dieser A-Datensatz erstellt wird.

IXFAXTYP
Gibt an, dass dies der Subtyp "X" der DB2-Anwendungssätze ist.

IXFADATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*. Dieses Feld muss denselben Wert wie das IXFHDATE-Feld aufweisen.

IXFATIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld muss denselben Wert wie das IXFHTIME-Feld aufweisen.

IXFAYCNT
Gibt die Anzahl der Sätze für untergeordnete Tabellen an, die nach diesem Hierarchiesatz erwartet werden.

IXFAYSTR
Gibt den Index der Sätze für untergeordnete Tabellen am Anfang der ex-

portierten Daten an. Wurde der Export einer Hierarchie von einer untergeordneten Tabelle aus gestartet, die nicht die Stammtabelle ist, werden alle übergeordneten Tabellen dieser untergeordneten Tabelle exportiert. Die Position dieser untergeordneten Tabelle innerhalb der IXF-Datei wird ebenfalls in diesem Feld gespeichert. Der erste X-Datensatz stellt die Spalte mit einem Index 0 dar.

DB2-SATZ FÜR UNTERGEORDNETE TABELLE

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	006 BYTE	CHARACTER	Satzlänge
IXFARECT	001 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	012-BYTE	CHARACTER	Anwendungskennzeichen = 'DB2 02.00'
IXFAYTYP	001-BYTE	CHARACTER	Anwendungsspezifischer Datentyp = 'Y'
IXFADATE	008 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Daten
IXFATIME	006 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Zeit
IXFASCHL	003 BYTE	CHARACTER	Länge des Typschemennamens
IXFASCHN	256 BYTE	CHARACTER	Typschemenname
IXFATYPL	003 BYTE	CHARACTER	Länge des Typnamens
IXFATYPN	256 BYTE	CHARACTER	Typname
IXFATABL	003 BYTE	CHARACTER	Länge des Tabellennamens
IXFATABN	256 BYTE	CHARACTER	Tabellenname
IXFAPNDX	010 BYTE	CHARACTER	Index der untergeordneten Tabelle für übergeordnete Tabelle
IXFASNDX	005-BYTE	CHARACTER	Beginn Spaltenindex der aktuellen Tabelle
IXFAENDX	005-BYTE	CHARACTER	Ende Spaltenindex der aktuellen Tabelle

Zur Beschreibung einer untergeordneten Tabelle als Teil einer Hierarchie wird 1 Datensatz diesen Typs verwendet. Alle Sätze für untergeordnete Tabellen, die zu einer Hierarchie gehören, müssen zusammen gespeichert werden und unmittelbar auf den entsprechenden Hierarchiesatz folgen. Eine untergeordnete Tabelle besteht aus einer oder mehreren Spalten. Jede Spalte wird in einem Spaltensatz beschrieben. Jede Spalte in einer untergeordneten Tabelle muss durch eine Reihe von aufeinander folgenden C-Datensätzen beschrieben werden. DB2-Sätze für untergeordnete Tabelle enthalten die folgenden Felder:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der A-Datensatz muss lang genug sein für das gesamte IXFAPPID-Feld.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben, dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das DB2 als die Anwendung angibt, von der dieser A-Datensatz erstellt wird.

IXFAYTYP
Gibt an, dass dies der Subtyp "Y" der DB2-Anwendungssätze ist.

IXFADATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*. Dieses Feld muss denselben Wert wie das IXFHDATE-Feld aufweisen.

IXFATIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld muss denselben Wert wie das IXFHTIME-Feld aufweisen.

IXFASCHL
Die Länge des Schemennamens für die untergeordnete Tabelle im IXFASCHN-Feld in Byte.

IXFASCHN
Der Name des Schemas für die untergeordnete Tabelle.

IXFATYPL
Die Länge des Namens der untergeordneten Tabelle im IXFATYPN-Feld in Byte.

IXFATYPN
Der Name der untergeordneten Tabelle.

IXFATABL
Die Länge des Tabellennamens im IXFATABN-Feld in Byte.

IXFATABN
Der Name der Tabelle.

IXFAPNDX
Der Eintragsindex für die untergeordnete Tabelle der übergeordneten Tabelle. Bildet diese untergeordnete Tabelle den Stamm einer Hierarchie, enthält dieses Feld den Wert -1.

IXFASNDX
Startindex der Spaltensätze, aus denen diese untergeordnete Tabelle besteht.

IXFAENDX
Endindex der Spaltensätze, aus denen diese untergeordnete Tabelle besteht.

DB2-FORTSETZUNGSSATZ

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	006 BYTE	CHARACTER	Satzlänge
IXFARECT	001 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	012-BYTE	CHARACTER	Anwendungskennzeichen = 'DB2 02.00'
IXFACTYP	001 BYTE	CHARACTER	Anwendungsspezifischer Datentyp = "C"
IXFADATE	008 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Daten
IXFATIME	006 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Zeit
IXFALAST	002 BYTE	SHORT INT	Datenträgernummer der ersten Diskette
IXFATHIS	002 BYTE	SHORT INT	Datenträgernummer dieser Diskette
IXFANEXT	002 BYTE	SHORT INT	Datenträgernummer der nächsten Diskette

Dieser Datensatz befindet sich am Ende jeder Datei, die Bestandteil einer auf mehrere Datenträger verteilten IXF-Datei ist, sofern es sich bei dem aktuellen Datenträger nicht um den letzten Datenträger handelt. Außerdem ist dieser Datensatz am Beginn jeder Datei zu finden, die Bestandteil einer auf mehrere Datenträger verteilten IXF-Datei ist, sofern es sich bei dem aktuellen Datenträger nicht um den ersten Datenträger handelt. Der Zweck dieses Datensatzes ist die Einhaltung der Dateireihenfolge. DB2-Fortsetzungssätze enthalten die folgenden Felder:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der A-Datensatz muss lang genug sein für das gesamte IXFAPPID-Feld.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben, dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden

von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das DB2 als die Anwendung angibt, von der dieser A-Datensatz erstellt wird.

IXFACTYP
Gibt an, dass dies der Subtyp "C" der DB2-Anwendungssätze ist.

IXFADATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*. Dieses Feld muss denselben Wert wie das IXFHDATE-Feld aufweisen.

IXFATIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld muss denselben Wert wie das IXFHTIME-Feld aufweisen.

IXFALAST
Dieses Feld ist ein Feld für Binärdaten im Little-Endian-Format. Der Wert sollte um 1 kleiner als der Wert im IXFATHIS-Feld sein.

IXFATHIS
Dieses Feld ist ein Feld für Binärdaten im Little-Endian-Format. Der Wert in diesem Feld sollte auf Fortsetzungsdatenträgern ebenfalls fortlaufend sein. Der erste Datenträger hat den Wert 1.

IXFANEXT
Dieses Feld ist ein Feld für Binärdaten im Little-Endian-Format. Der Wert sollte um 1 größer als der Wert im IXFATHIS-Feld sein, es sei denn, der Datensatz befindet sich am Dateianfang. In diesem Fall sollte der Wert 0 lauten.

DB2-BEENDIGUNGSSATZ

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	006 BYTE	CHARACTER	Satzlänge
IXFARECT	001 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	012-BYTE	CHARACTER	Anwendungskennzeichen = 'DB2 02.00'
IXFAETYP	001-BYTE	CHARACTER	Anwendungsspezifischer Datentyp = 'E'
IXFADATE	008 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Daten
IXFATIME	006 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Zeit

Dieser Datensatz ist das Dateiendezeichen, das sich am Ende einer IXF-Datei befindet. DB2-Beendigungssätze enthalten die folgenden Felder:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der A-Datensatz muss lang genug sein für das gesamte IXFAPPID-Feld.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben, dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das DB2 als die Anwendung angibt, von der dieser A-Datensatz erstellt wird.

IXFAETYP
Gibt an, dass dies der Subtyp "E" der DB2-Anwendungssätze ist.

IXFADATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*. Dieses Feld muss denselben Wert wie das IXFHDATE-Feld aufweisen.

IXFATIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld muss denselben Wert wie das IXFHTIME-Feld aufweisen.

DB2-IDENTITÄTSSATZ

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	06 BYTE	CHARACTER	Satzlänge
IXFARECT	01 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	12 BYTE	CHARACTER	Anwendungskennzeichen
IXFATYPE	01 BYTE	CHARACTER	anwendungsspezifischer Satztyp = "S"
IXFADATE	08 BYTE	CHARACTER	Erstellungsdatum des Anwendungssatzes
IXFATIME	06 BYTE	CHARACTER	Erstellungszeit des Anwendungssatzes
IXFACOLN	06 BYTE	CHARACTER	Spaltennummer der Identitätsspalte
IXFAITYP	01 BYTE	CHARACTER	GENERATED ALWAYS ("Y" oder "N")
IXFASTRT	33 BYTE	CHARACTER	START AT-Wert für Identitätsspalte
IXFAINCR	33 BYTE	CHARACTER	INCREMENT BY-Wert für Identitätsspalte
IXFACACH	10 BYTE	CHARACTER	CACHE-Wert für Identitätsspalte
IXFAMINV	33-BYTE	CHARACTER	MINVALUE-Wert für Identitätsspalte
IXFAMAXV	33-BYTE	CHARACTER	MAXVALUE-Wert für Identitätsspalte
IXFACYCL	01-BYTE	CHARACTER	CYCLE-Wert für Identitätsspalte ('Y' oder 'N')
IXFAORDR	01-BYTE	CHARACTER	ORDER-Wert für Identitätsspalte ('Y' oder 'N')
IXFARMRL	03-BYTE	CHARACTER	Länge der Anmerkung für Identitätsspalte
IXFARMRK	254-BYTE	CHARACTER	Wert der Anmerkung für Identitätsspalte

DB2-Identitätssätze enthalten die folgenden Felder:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Der A-Datensatz muss lang genug sein für das gesamte IXFAPPID-Feld.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben, dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das DB2 als die Anwendung angibt, von der dieser A-Datensatz erstellt wird.

IXFATYPE
Gibt den anwendungsspezifischen Satztyp an. Dieses Feld sollte immer den Wert "S" haben.

IXFADATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*. Dieses Feld muss denselben Wert wie das IXFHDATE-Feld aufweisen.

IXFATIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld muss denselben Wert wie das IXFHTIME-Feld aufweisen.

IXFACOLN
Die Spaltennummer der Identitätsspalte in der Tabelle.

IXFAITYP
Der Typ der Identitätsspalte. Der Wert "Y" gibt an, dass die Identitätsspalte als GENERATED ALWAYS definiert ist, d. h. in jedem Fall generiert wird. Alle anderen Werte werden so interpretiert, dass die Spalte als GENERATED BY DEFAULT definiert ist.

IXFASTRT
Der START AT-Wert für die Identitätsspalte, der zum Zeitpunkt der Tabellenerstellung an die Anweisung CREATE TABLE übergeben wurde.

IXFAINCR
Der INCREMENT BY-Wert für die Identitätsspalte, der zum Zeitpunkt der Tabellenerstellung an die Anweisung CREATE TABLE übergeben wurde.

IXFACACH
Der CACHE-Wert für die Identitätsspalte, der zum Zeitpunkt der Tabellenerstellung an die Anweisung CREATE TABLE übergeben wurde. Der Wert "1" entspricht der Option NO CACHE.

IXFAMINV
Der MINVALUE-Wert für die Identitätsspalte, der zum Zeitpunkt der Tabellenerstellung an die Anweisung CREATE TABLE übergeben wurde.

IXFAMAXV
Der MAXVALUE-Wert für die Identitätsspalte, der zum Zeitpunkt der Tabellenerstellung an die Anweisung CREATE TABLE übergeben wurde.

IXFACYCL
Der CYCLE-Wert für die Identitätsspalte, der zum Zeitpunkt der Tabellenerstellung an die Anweisung CREATE TABLE übergeben wurde. Der Wert "Y" entspricht der Option CYCLE. Alle anderen Werte entsprechen NO CYCLE.

IXFAORDR
Der ORDER-Wert für die Identitätsspalte, der zum Zeitpunkt der Tabellenerstellung an die Anweisung CREATE TABLE übergeben wurde. Der Wert "Y" entspricht der Option ORDER. Alle anderen Werte entsprechen NO ORDER.

IXFARMRL
Die Länge (in Byte) der Anmerkung im Feld IXFARMRK.

IXFARMRK
Dies ist die vom Benutzer eingegebene Anmerkung, die der Identitätsspalte zugeordnet ist. Dieses Feld dient nur zur Information. Dieses Feld wird vom Datenbankmanager beim Importieren von Daten nicht verwendet.

DB2 SQLCA RECORD

FELDNAME	LÄNGE	TYP	KOMMENTARE
IXFARECL	006 BYTE	CHARACTER	Satzlänge
IXFARECT	001 BYTE	CHARACTER	Satztyp = "A"
IXFAPPID	012-BYTE	CHARACTER	Anwendungskennzeichen = 'DB2 02.00'
IXFAITYP	001-BYTE	CHARACTER	Anwendungsspezifischer Datentyp = 'A'
IXFADATE	008 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Daten
IXFATIME	006 BYTE	CHARACTER	aus dem H-Datensatz geschriebene Zeit
IXFASLCA	136-BYTE	Variable	sqlca - SQL-Kommunikationsbereich

Ein Datensatz dieses Typs wird dazu verwendet, anzugeben, dass die IXF-Datei nicht zur erneuten Erstellung der Tabelle bei einer nachfolgenden Importoperation verwendet werden kann. Die mit IXFASLCA zurückgegebene Nachricht und der Ursachencode liefern weitere Informationen hierzu.

DB2-SQLCA-Sätze enthalten die folgenden Felder:

IXFARECL
Der Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des Teils des PC/IXF-Datensatzes in Byte angibt, der auf den Satzlängenanzeiger folgt, d. h. die gesamte Satzgröße minus 6 Byte. Jeder A-Datensatz muss lang genug sein, um mindestens das gesamte `IXFAPPID`-Feld zu umfassen.

IXFARECT
Der IXF-Satztyp, der für diesen Datensatz auf A gesetzt ist, um anzugeben, dass es sich um einen Anwendungssatz handelt. Diese Datensätze werden von Programmen ignoriert, die keine Kenntnis des Inhalts und Formats der Daten haben, die vom Anwendungskennzeichen impliziert werden.

IXFAPPID
Das Anwendungskennzeichen, das DB2 als die Anwendung angibt, von der dieser A-Datensatz erstellt wird.

IXFAITYP
Gibt an, dass dies der Subtyp "A" der DB2-Anwendungssätze ist.

IXFADATE
Das Datum, an dem die Datei geschrieben wurde, im Format *jjjjmmtt*. Dieses Feld muss denselben Wert wie das `IXFHDATE`-Feld aufweisen.

IXFATIME
Die Uhrzeit, zu der die Datei geschrieben wurde, im Format *ssmmss*. Dieses Feld muss denselben Wert wie das `IXFHTIME`-Feld aufweisen.

IXFASLCA
Der SQL-Kommunikationsbereich mit der Warnung SQL27984W sowie einem Ursachencode, der eine Erläuterung dazu liefert, warum die IXF-Datei nicht alle Informationen enthält, die der Befehl **IMPORT** zum erneuten Erstellen der Tabelle benötigt.

PC/IXF-Datentypen:

Tabelle 23. PC/IXF-Datentypen

Name	IXFCTYPE-Wert	Beschreibung
BIGINT	492	Eine 8 Byte große ganze Zahl in dem von IXFTMFRM festgelegten Format. Dieser Wert stellt eine ganze Zahl im Bereich zwischen -9 223 372 036 854 775 808 und 9 223 372 036 854 775 807 dar. IXFCSBCP und IXFCDBCP sind nicht signifikant und müssen null sein. IXFCLENG wird nicht verwendet und muss Leerzeichen enthalten.

Tabelle 23. PC/IXF-Datentypen (Forts.)

Name	IXFCTYPE-Wert	Beschreibung
BLOB, CLOB	404, 408	Eine Zeichenfolge variabler Länge. Die maximale Länge der Zeichenfolge ist im IXFCLENG-Feld des Spaltendeskriptorsatzes enthalten, und sie darf 32 767 Byte nicht überschreiten. Der Zeichenfolge selbst wird ein aktueller Längenanzeiger vorangestellt, bei dem es sich um eine 4 Byte große ganze Zahl handelt, die die Länge der Zeichenfolge in Byte angibt. Die Zeichenfolge liegt in der Codepage vor, die in IXFCSBCP angegeben ist. Folgendes gilt nur für BLOBs: Wenn IXFCSBCP null ist, besteht die Zeichenfolge aus Bitdaten und darf von keinem Umsetzungsprogramm umgesetzt werden. Folgendes gilt nur für CLOBs: Wenn IXFCDBCP ungleich null ist, kann die Zeichenfolge auch Doppelbytezeichen in der Codepage enthalten, die in IXFCDBCP angegeben ist.
BLOB_LOCATION_ SPECIFIER und DBCLOB_ LOCATION_ SPECIFIER	960, 964, 968	Ein Feld fester Länge, das nicht größer als 255 Byte sein kann. Die LOB-Positionskennung (LLS) liegt in der Codepage vor, die in IXFCSBCP angegeben ist. Wenn IXFCSBCP null ist, besteht die LOB-Positionskennung aus Bitdaten und darf von keinem Umsetzungsprogramm umgesetzt werden. Wenn IXFCDBCP ungleich null ist, kann die Zeichenfolge auch Doppelbytezeichen in der Codepage enthalten, die in IXFCDBCP angegeben ist. Da die Länge der LOB-Positionskennung in IXFCLENG gespeichert ist, geht die tatsächliche Länge des ursprünglichen großen Objekts verloren. PC/IXF-Dateien mit Spalten dieses Typs dürfen nicht zur erneuten Erstellung des LOB-Feldes verwendet werden, weil das LOB-Feld mit der Länge der LOB-Positionskennung erstellt wird.

Tabelle 23. PC/IXF-Datentypen (Forts.)

Name	IXFCTYPE-Wert	Beschreibung
BLOB_FILE, CLOB_FILE, DBCLOB_FILE	916, 920, 924	Ein Feld fester Länge, das eine SQLFILE-Struktur enthält, bei der die Felder *name_length* und *name* ausgefüllt sind. Die Länge der Struktur ist im IXFCLENG-Feld des Spaltendeskriptorsatzes enthalten und darf 255 Byte nicht überschreiten. Der Dateiname liegt in der Codepage vor, die in IXFCSBCP angegeben ist. Wenn IXFCDBCP ungleich null ist, kann der Dateiname auch Doppelbytezeichen in der Codepage enthalten, die in IXFCDBCP angegeben ist. Wenn IXFCSBCP null ist, besteht der Dateiname aus Bitdaten und darf von keinem Umsetzungsprogramm umgesetzt werden. Da die Länge der Struktur in IXFCLENG gespeichert ist, geht die eigentliche Länge des ursprünglichen LOBs verloren. IXF-Dateien mit Spalten des Typs BLOB_FILE, CLOB_FILE oder DBCLOB_FILE dürfen nicht zur erneuten Erstellung des LOB-Feldes verwendet werden, da das LOB-Feld mit der Länge *sql_lobfile_len* erstellt wird.
CHAR	452	Eine Zeichenfolge mit fester Länge. Die Länge der Zeichenfolge ist im IXFCLENG-Feld des Spaltendeskriptorsatzes enthalten, und sie darf 254 Byte nicht überschreiten. Die Zeichenfolge liegt in der Codepage vor, die in IXFCSBCP angegeben ist. Wenn IXFCDBCP ungleich null ist, kann die Zeichenfolge auch Doppelbytezeichen in der Codepage enthalten, die in IXFCDBCP angegeben ist. Wenn IXFCSBCP null ist, besteht die Zeichenfolge aus Bitdaten und darf von keinem Umsetzungsprogramm umgesetzt werden.
DATE	384	Ein Zeitpunkt gemäß gregorianischem Kalender. Jedes Datum ist eine 10 Byte große Zeichenfolge im ISO-Format: *jjjj-mm-tt*. Der Wertebereich für das Jahr liegt zwischen `0001` und `9999`. Der Wertebereich für den Monat liegt zwischen `01` und `12`. Der Bereich für den Tag ist `01` bis *n*, wobei *n* vom Monat abhängt. Dabei gelten die üblichen Regeln für die Tage eines Monats und Schaltjahre. Führende Nullen können bei keinem Teil weggelassen werden. IXFCLENG wird nicht verwendet und muss Leerzeichen enthalten. Gültige Zeichen für DATE sind in allen PC-ASCII-Codepages gleich. Daher sind IXFCSBCP und IXFCDBCP nicht signifikant und müssen null sein.

Tabelle 23. PC/IXF-Datentypen (Forts.)

Name	IXFCTYPE-Wert	Beschreibung
DBCLOB	412	Eine Zeichenfolge variabler Länge aus Doppelbytezeichen. Das IXFCLENG-Feld im Spaltendeskriptorsatz gibt die maximale Anzahl von Doppelbytezeichen in der Zeichenfolge an, die 16 383 nicht überschreiten darf. Der Zeichenfolge selbst wird ein aktueller Längenanzeiger vorangestellt, bei dem es sich um eine 4 Byte große ganze Zahl handelt, die die Länge der Zeichenfolge in Doppelbytezeichen angibt (d. h. der Wert dieser ganzen Zahl ist die halbe Länge der Zeichenfolge in Byte). Die Zeichenfolge liegt in der DBCS-Codepage vor, wie mit IXFCDBCP im C-Datensatz angegeben. Da die Zeichenfolge nur Doppelbytezeichen enthält, muss IXFCSBCP null sein. Es gibt keine DBCS-Startzeichen oder DBCS-Endezeichen.
DECIMAL	484	Eine gepackte Dezimalzahl mit der Genauigkeit P (wie durch die ersten drei Byte von IXFCLENG im Spaltendeskriptorsatz festgelegt) und der Anzahl S der Kommastellen (wie durch die letzten beiden Byte von IXFCLENG festgelegt). Die Länge einer gepackten Dezimalzahl in Byte ist (P+2)/2. Die Genauigkeit muss eine ungerade Zahl zwischen 1 und 31 (jeweils einschließlich) sein. Die gepackte Dezimalzahl liegt im internen Format vor, das mit IXFTMFRM angegeben wird, während die gepackte Dezimalzahl für den PC so definiert ist, dass sie mit der gepackten Dezimalzahl für das IBM System/370 identisch ist. IXFCSBCP und IXFCDBCP sind nicht signifikant und müssen null sein.
DECFLOAT	996	Bei einem dezimalen Gleitkommawert handelt es sich um eine Zahl mit Dezimalzeichen gemäß IEEE-Standard 754r. Die Position des Dezimalzeichens wird jeweils im dezimalen Gleitkommawert gespeichert. Dezimale Gleitkommazahlen liegen im Bereich der Zahlen mit einer Dezimalstellengenauigkeit von 16 oder 34 Stellen, und der Exponentenbereich liegt bei 10-383 bis 10+384 bzw. entsprechend bei 10-6143 bis 10+6144. Die Speicherlänge des 16-stelligen Werts beträgt 8 Byte, die Speicherlänge des 34-stelligen Werts 16 Byte.

Tabelle 23. PC/IXF-Datentypen (Forts.)

Name	IXFCTYPE-Wert	Beschreibung
FLOATING POINT	480	Entweder eine lange (8 Byte) oder kurze (4 Byte) Gleitkommazahl. Dies hängt davon ab, ob IXFCLENG auf 8 oder 4 gesetzt ist. Die Daten liegen im internen Maschinenformat vor, wie mit IXFTMFRM angegeben. IXFCSBCP und IXFCDBCP sind nicht signifikant und müssen null sein. 4 Byte große Gleitkommazahlen werden vom Datenbankmanager nicht unterstützt.
GRAPHIC	468	Eine Zeichenfolge fester Länge aus Doppelbytezeichen. Das IXFCLENG-Feld im Spaltendeskriptorsatz gibt die Anzahl von Doppelbytezeichen in der Zeichenfolge an und darf maximal den Wert 127 enthalten. Die eigentliche Länge der Zeichenfolge ist das Doppelte des Werts des IXFCLENG-Feldes in Byte. Die Zeichenfolge liegt in der DBCS-Codepage vor, wie mit IXFCDBCP im C-Datensatz angegeben. Da die Zeichenfolge nur Doppelbytezeichen enthält, muss IXFCSBCP null sein. Es gibt keine DBCS-Startzeichen oder DBCS-Endezeichen.
INTEGER	496	Eine 4 Byte große ganze Zahl in dem von IXFTMFRM festgelegten Format. Dieser Wert stellt eine ganze Zahl im Bereich zwischen -2 147 483 648 und +2 147 483 647 dar. IXFCSBCP und IXFCDBCP sind nicht signifikant und müssen null sein. IXFCLENG wird nicht verwendet und muss Leerzeichen enthalten.
LONGVARCHAR	456	Eine Zeichenfolge variabler Länge. Die maximale Länge der Zeichenfolge ist im IXFCLENG-Feld des Spaltendeskriptorsatzes enthalten, und sie darf 32 767 Byte nicht überschreiten. Der Zeichenfolge selbst wird ein aktueller Längenanzeiger vorangestellt, bei dem es sich um eine 2 Byte große ganze Zahl handelt, die die Länge der Zeichenfolge in Byte angibt. Die Zeichenfolge liegt in der Codepage vor, die in IXFCSBCP angegeben ist. Wenn IXFCDBCP ungleich null ist, kann die Zeichenfolge auch Doppelbytezeichen in der Codepage enthalten, die in IXFCDBCP angegeben ist. Wenn IXFCSBCP null ist, besteht die Zeichenfolge aus Bitdaten und darf von keinem Umsetzungsprogramm umgesetzt werden.

Tabelle 23. PC/IXF-Datentypen (Forts.)

Name	IXFCTYPE-Wert	Beschreibung
LONG VARGRAPHIC	472	Eine Zeichenfolge variabler Länge aus Doppelbytezeichen. Das IXFCLENG-Feld im Spaltendeskriptorsatz gibt die maximale Anzahl von Doppelbytezeichen für die Zeichenfolge an, die 16 383 nicht überschreiten darf. Der Zeichenfolge selbst wird ein aktueller Längenanzeiger vorangestellt, bei dem es sich um eine 2 Byte große ganze Zahl handelt, die die Länge der Zeichenfolge in Doppelbytezeichen angibt (d. h. der Wert dieser ganzen Zahl ist die halbe Länge der Zeichenfolge in Byte). Die Zeichenfolge liegt in der DBCS-Codepage vor, wie mit IXFCDBCP im C-Datensatz angegeben. Da die Zeichenfolge nur Doppelbytezeichen enthält, muss IXFCSBCP null sein. Es gibt keine DBCS-Startzeichen oder DBCS-Endezeichen.
SMALLINT	500	Eine 2 Byte große ganze Zahl in dem von IXFTMFRM festgelegten Format. Dieser Wert stellt eine ganze Zahl zwischen -32 768 und +32 767 dar. IXFCSBCP und IXFCDBCP sind nicht signifikant und müssen null sein. IXFCLENG wird nicht verwendet und muss Leerzeichen enthalten.
TIME	388	Ein Zeitpunkt gemäß 24-Stunden-Zeiteinteilung. Jede Zeitangabe ist eine 8 Byte große Zeichenfolge im ISO-Format: *ss.mm.ss*. Der Wertebereich für die Stunde liegt zwischen `00` und `24` und der Wertebereich für Minuten und Sekunden zwischen `00` und `59`. Bei Stunde `24` sind die Minuten- und Sekundenangaben gleich `00`. Der kleinste Zeitwert ist `00.00.00`, der größte `24.00.00`. Führende Nullen können bei keinem Teil weggelassen werden. IXFCLENG wird nicht verwendet und muss Leerzeichen enthalten. Gültige Zeichen für TIME sind in allen PC-ASCII-Codepages gleich. Daher sind IXFCSBCP und IXFCDBCP nicht signifikant und müssen null sein.

Tabelle 23. PC/IXF-Datentypen (Forts.)

Name	IXFCTYPE-Wert	Beschreibung
TIMESTAMP	392	Das Datum und die Uhrzeit mit der Genauigkeit von Bruchsekunden. Jede Zeitmarke ist eine Zeichenfolge im Format *jjjj-mm-tt-ss.mm.ss.nnnnnn* (Jahr Monat Tag Stunde Minuten Sekunden Bruchsekunden). Ab Version 9.7 ist die Genauigkeit für Zeitmarken im Feld IXFCLENG des Spaltendeskriptorsatzes enthalten und darf 12 nicht überschreiten. Vor Version 9.7 wird IXFCLENG nicht verwendet und muss Leerzeichen enthalten. Gültige Zeichen für TIMESTAMP sind in allen PC-ASCII-Codepages gleich. Daher sind IXFCSBCP und IXFCDBCP nicht signifikant und müssen null sein.
VARCHAR	448	Eine Zeichenfolge variabler Länge. Die maximale Länge der Zeichenfolge in Byte ist im IXFCLENG-Feld des Spaltendeskriptorsatzes enthalten und darf 254 Byte nicht überschreiten. Der Zeichenfolge selbst wird ein aktueller Längenanzeiger vorangestellt, bei dem es sich um eine 2 Byte große ganze Zahl handelt, die die Länge der Zeichenfolge in Byte angibt. Die Zeichenfolge liegt in der Codepage vor, die in IXFCSBCP angegeben ist. Wenn IXFCDBCP ungleich null ist, kann die Zeichenfolge auch Doppelbytezeichen in der Codepage enthalten, die in IXFCDBCP angegeben ist. Wenn IXFCSBCP null ist, besteht die Zeichenfolge aus Bitdaten und darf von keinem Umsetzungsprogramm umgesetzt werden.
VARGRAPHIC	464	Eine Zeichenfolge variabler Länge aus Doppelbytezeichen. Das IXFCLENG-Feld im Spaltendeskriptorsatz gibt die maximale Anzahl von Doppelbytezeichen in der Zeichenfolge an, die 127 nicht überschreiten darf. Der Zeichenfolge selbst wird ein aktueller Längenanzeiger vorangestellt, bei dem es sich um eine 2 Byte große ganze Zahl handelt, die die Länge der Zeichenfolge in Doppelbytezeichen angibt (d. h. der Wert dieser ganzen Zahl ist die halbe Länge der Zeichenfolge in Byte). Die Zeichenfolge liegt in der DBCS-Codepage vor, wie mit IXFCDBCP im C-Datensatz angegeben. Da die Zeichenfolge nur Doppelbytezeichen enthält, muss IXFCSBCP null sein. Es gibt keine DBCS-Startzeichen oder DBCS-Endezeichen.

Nicht alle Kombinationen von IXFCSBCP- und IXFCDBCP-Werten für die PC/IXF-Zeichen- oder -Grafikspalten sind gültig. Eine PC/IXF-Zeichen- oder -Grafikspalte mit einer ungültigen Kombination aus IXFCSBCP und IXFCDBCP ist ein ungültiger Datentyp.

Tabelle 24. Gültige PC/IXF-Datentypen

PC/IXF-Datentyp	Gültige Paare (IXFCSBCP,IXFCDBCP)	Ungültige Paare (IXFCSBCP,IXFCDBCP)
CHAR, VARCHAR oder LONG VARCHAR	(0,0), (x,0) oder (x,y)[1]	(0,y)[1]
BLOB	(0,0)	(x,0), (0,y) oder (x,y)[1]
CLOB	(x,0), (x,y)[1]	(0,0), (0,y)[1]
GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC oder DBCLOB	(0,y)[1]	(0,0), (x,0) oder (x,y)[1]

Anmerkung: [1] x und y sind nicht 0.

PC/IXF-Datentypbeschreibungen:

Tabelle 25. Gültige Datentypformate für das PC/IXF-Dateiformat

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
BIGINT	Eine BIGINT-Spalte wird erstellt, die mit der Datenbankspalte identisch ist.	Eine Spalte eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich von -9 223 372 036 854 775 808 bis 9 223 372 036 854 775 807 liegen.
BLOB	Eine PC/IXF-Spalte des Typs BLOB wird erstellt. Die maximale Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs CHAR, VARCHAR, LONG VARCHAR, BLOB, BLOB_FILE oder BLOB_LOCATION_SPECIFIER ist in folgenden Fällen zulässig: • Die Datenbankspalte weist die Markierung FOR BIT DATA auf. • Der Einzelbyte-Codepage-Wert der PC/IXF-Spalte stimmt mit der SBCS-CPGID der Datenbankspalte überein, und der Doppelbyte-Codepage-Wert der PC/IXF-Spalte entspricht Null oder der DBCS-CPGID der Datenbankspalte. Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC BLOB ist ebenfalls zulässig. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge mit der maximalen Länge der Datenbankspalte kompatibel sein.

Tabelle 25. Gültige Datentypformate für das PC/IXF-Dateiformat (Forts.)

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
CHAR	Eine PC/IXF-Spalte des Typs CHAR wird erstellt. Die Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den PC/IXF-Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs CHAR, VARCHAR oder LONG VARCHAR ist in folgenden Fällen zulässig: • Die Datenbankspalte weist die Markierung FOR BIT DATA auf. • Der Einzelbyte-Codepage-Wert der PC/IXF-Spalte stimmt mit der SBCS-CPGID der Datenbankspalte überein, und der Doppelbyte-Codepage-Wert der PC/IXF-Spalte entspricht Null oder der DBCS-CPGID der Datenbankspalte. Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC ist ebenfalls zulässig, wenn die Datenbankspalte mit FOR BIT DATA markiert ist. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge in jedem Fall mit der Länge der Datenbankspalte kompatibel sein. Die Daten werden bei Bedarf auf der rechten Seite mit Einzelbyteleerzeichen (x'20') aufgefüllt.
CLOB	Eine PC/IXF-Spalte des Typs CLOB wird erstellt. Die maximale Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs CHAR, VARCHAR, LONG VARCHAR, CLOB, CLOB_FILE oder CLOB_LOCATION_SPECIFIER ist zulässig, wenn der Einzelbyte-Codepage-Wert der PC/IXF-Spalte mit der SBCS-CPGID der Datenbankspalte identisch ist und der Doppelbyte-Codepage-Wert der PC/IXF-Spalte entweder Null ist oder der DBCS-CPGID der Datenbankspalte entspricht. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge mit der maximalen Länge der Datenbankspalte kompatibel sein.
DATE	Eine DATE-Spalte wird erstellt, die mit der Datenbankspalte identisch ist.	Eine PC/IXF-Spalte des Typs DATE ist die übliche Eingabe. Das Dienstprogramm IMPORT versucht auch, Spalten in einem beliebigen Zeichentyp anzunehmen, sofern sie keine inkompatiblen Längen aufweisen. Die Zeichenspalte in der PC/IXF-Datei muss Datumswerte in einem Format enthalten, das mit dem Gebietscode der Zieldatenbank konsistent ist.

Tabelle 25. Gültige Datentypformate für das PC/IXF-Dateiformat (Forts.)

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
DBCLOB	Eine PC/IXF-Spalte des Typs DBCLOB wird erstellt. Die maximale Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, DBCLOB_FILE oder DBCLOB_LOCATION_SPECIFIER ist zulässig, wenn der Doppelbyte-Codepage-Wert der PC/IXF-Spalte mit dem der Datenbankspalte übereinstimmt. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge mit der maximalen Länge der Datenbankspalte kompatibel sein.
DECIMAL	Eine DECIMAL-Spalte, die mit der Datenbankspalte identisch ist, wird erstellt. Die Genauigkeit und die Anzahl der Kommastellen werden im Spaltendeskriptorsatz gespeichert.	Eine Spalte eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich der DECIMAL-Spalte liegen, in die sie importiert werden.
DECFLOAT	Eine DECFLOAT-Spalte, die mit der Datenbankspalte identisch ist, wird erstellt. Die Anzahl der Kommastellen in der Spalte werden im Spaltendeskriptorsatz gespeichert.	Spalten des Typs SMALLINT, INTEGER, BIGINT (nur in DECFLOAT(34)), DECIMAL, FLOAT, REAL, DOUBLE oder DECFLOAT(16) (nur in DECFLOAT(34)) werden akzeptiert. Andere numerische Spaltentypen sind für DECFLOAT gültig, übersteigt die Anzahl der Kommastellen jedoch die Dezimalgenauigkeit der Zielspalte, wird der betreffende Wert gerundet.
FLOAT	Eine FLOAT-Spalte wird erstellt, die mit der Datenbankspalte identisch ist.	Eine Spalte eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Alle Werte liegen innerhalb des Bereichs.
GRAPHIC (nur DBCS)	Eine PC/IXF-Spalte des Typs GRAPHIC wird erstellt. Die Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC ist zulässig, wenn der Doppelbyte-Codepage-Wert der PC/IXF-Spalte mit dem der Datenbankspalte übereinstimmt. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge mit der Länge der Datenbankspalte kompatibel sein. Die Daten werden bei Bedarf auf der rechten Seite mit Doppelbyteleerzeichen (x'8140') aufgefüllt.
INTEGER	Eine Spalte des Typs INTEGER wird erstellt, die mit der Datenbankspalte identisch ist.	Eine Spalte eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich von -2 147 483 648 bis 2 147 483 647 liegen.

Tabelle 25. Gültige Datentypformate für das PC/IXF-Dateiformat (Forts.)

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
LONG VARCHAR	Eine PC/IXF-Spalte des Typs LONG VARCHAR wird erstellt. Die maximale Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs CHAR, VARCHAR oder LONG VARCHAR ist in folgenden Fällen zulässig: • Die Datenbankspalte weist die Markierung FOR BIT DATA auf. • Der Einzelbyte-Codepage-Wert der PC/IXF-Spalte stimmt mit der SBCS-CPGID der Datenbankspalte überein, und der Doppelbyte-Codepage-Wert der PC/IXF-Spalte entspricht Null oder der DBCS-CPGID der Datenbankspalte. Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC ist ebenfalls zulässig, wenn die Datenbankspalte mit FOR BIT DATA markiert ist. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge in jedem Fall mit der maximalen Länge der Datenbankspalte kompatibel sein.
LONG VARGRAPHIC (nur DBCS)	Eine PC/IXF-Spalte des Typs LONG VARGRAPHIC wird erstellt. Die maximale Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC ist zulässig, wenn der Doppelbyte-Codepage-Wert der PC/IXF-Spalte mit dem der Datenbankspalte übereinstimmt. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge mit der maximalen Länge der Datenbankspalte kompatibel sein.
SMALLINT	Eine Spalte des Typs SMALLINT wird erstellt, die mit der Datenbankspalte identisch ist.	Eine Spalte eines beliebigen numerischen Typs (SMALLINT, INTEGER, BIGINT, DECIMAL oder FLOAT) wird akzeptiert. Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich von -32 768 bis 32 767 liegen.
TIME	Eine Spalte des Typs TIME wird erstellt, die mit der Datenbankspalte identisch ist.	Eine PC/IXF-Spalte des Typs TIME ist die übliche Eingabe. Das Dienstprogramm IMPORT versucht auch, Spalten in einem beliebigen Zeichentyp anzunehmen, sofern sie keine inkompatiblen Längen aufweisen. Die Zeichenspalte in der PC/IXF-Datei muss Zeitwerte in einem Format enthalten, das mit dem Gebietscode der Zieldatenbank konsistent ist.

Tabelle 25. Gültige Datentypformate für das PC/IXF-Dateiformat (Forts.)

Datentyp	Format in vom Dienstprogramm EXPORT erstellten Dateien	Für das Dienstprogramm IMPORT zulässiges Format
TIMESTAMP	Eine Spalte des Typs TIMESTAMP wird erstellt, die mit der Datenbankspalte identisch ist.	Eine PC/IXF-Spalte des Typs TIMESTAMP ist die übliche Eingabe. Das Dienstprogramm IMPORT versucht auch, Spalten in einem beliebigen Zeichentyp anzunehmen, sofern sie keine inkompatiblen Längen aufweisen. Die Zeichenspalte in der PC/IXF-Datei muss Daten im Eingabeformat für Zeitmarken enthalten.
VARCHAR	Wenn die maximale Länge der Datenbankspalte = 254 ist, wird eine PC/IXF-Spalte des Typs VARCHAR erstellt. Wenn die maximale Länge der Datenbankspalte > 254 ist, wird eine PC/IXF-Spalte des Typs LONG VARCHAR erstellt. Die maximale Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs CHAR, VARCHAR oder LONG VARCHAR ist in folgenden Fällen zulässig: • Die Datenbankspalte weist die Markierung FOR BIT DATA auf. • Der Einzelbyte-Codepage-Wert der PC/IXF-Spalte stimmt mit der SBCS-CPGID der Datenbankspalte überein, und der Doppelbyte-Codepage-Wert der PC/IXF-Spalte entspricht Null oder der DBCS-CPGID der Datenbankspalte. Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC ist ebenfalls zulässig, wenn die Datenbankspalte mit FOR BIT DATA markiert ist. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge in jedem Fall mit der maximalen Länge der Datenbankspalte kompatibel sein.
VARGRAPHIC (nur DBCS)	Wenn die maximale Länge der Datenbankspalte = 127 ist, wird eine PC/IXF-Spalte des Typs VARGRAPHIC erstellt. Wenn die maximale Länge der Datenbankspalte > 127 ist, wird eine PC/IXF-Spalte des Typs LONG VARGRAPHIC erstellt. Die maximale Länge der Datenbankspalte, der SBCS-CPGID-Wert und der DBCS-CPGID-Wert werden in den Spaltendeskriptorsatz kopiert.	Eine PC/IXF-Spalte des Typs GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC ist zulässig, wenn der Doppelbyte-Codepage-Wert der PC/IXF-Spalte mit dem der Datenbankspalte übereinstimmt. Wenn die PC/IXF-Spalte eine feste Länge besitzt, muss ihre Länge mit der maximalen Länge der Datenbankspalte kompatibel sein.

Allgemeine Regeln für den Import von PC/IXF-Dateien in Datenbanken:

Das Dienstprogramm IMPORT des Datenbankmanagers wendet beim Importieren einer PC/IXF-Datei in einer SBCS- oder DBCS-Umgebung die folgenden allgemeinen Regeln an:

- Das Dienstprogramm IMPORT akzeptiert nur Dateien im PC/IXF-Format (IXFHID = "IXF"). IXF-Dateien anderer Formate können nicht importiert werden.
- Das Dienstprogramm IMPORT weist eine PC/IXF-Datei mit mehr als 1024 Spalten zurück.
- Ist der Umfang von Kennungen bei einem Export in das IXF-Format größer, als es das IXF-Format unterstützt, ist die Exportoperation zwar erfolgreich, die bei diesem Export entstandene Datendatei kann jedoch nicht für nachfolgende Importoperationen im Modus `CREATE` verwendet werden. Es wird die Nachricht SQL27984W zurückgegeben.

  Anmerkung: Die Optionen `CREATE` und `REPLACE_CREATE` des Befehls IMPORT werden nicht weiter unterstützt und in zukünftigen Releases möglicherweise entfernt.
- Der Wert von IXFHSBCP im PC/IXF-H-Datensatz muss mit der SBCS-CPGID übereinstimmen, oder es muss eine Konvertierungstabelle zwischen IXFHSBCP/IXFHDBCP und der SBCS/DBCS-CPGID der Zieldatenbank geben. Der Wert von IXFHDBCP muss entweder "00000" sein oder dem Wert der DBCS-CPGID der Zieldatenbank entsprechen. Wenn eine dieser Bedingungen nicht erfüllt ist, weist das Dienstprogramm IMPORT die PC/IXF-Datei zurück, sofern nicht die Option `FORCEIN` angegeben ist.
- Ungültige Datentypen - neue Tabellen

  Der Import einer PC/IXF-Datei in eine *neue* Tabelle wird durch die Schlüsselwörter `CREATE` oder `REPLACE_CREATE` des Befehls IMPORT angegeben. Wenn eine PC/IXF-Spalte mit einem ungültigen Datentyp für den Import in eine neue Tabelle ausgewählt wird, wird das Dienstprogramm IMPORT beendet. Die gesamte PC/IXF-Datei wird zurückgewiesen, es wird keine Tabelle erstellt, und es werden keine Daten importiert.
- Ungültige Datentypen - vorhandene Tabellen

  Der Import einer PC/IXF-Datei in eine *vorhandene* Tabelle wird durch die Schlüsselwörter `INSERT`, `INSERT_UPDATE`, `REPLACE` oder `REPLACE_CREATE` des Befehls **`IMPORT`** angegeben. Wenn eine PC/IXF-Spalte mit einem ungültigen Datentyp für den Import in eine vorhandene Tabelle ausgewählt wird, ist eine von zwei Aktionen möglich:
  - Wenn die Zieltabellenspalte Nullwerte enthalten darf, werden alle Werte der ungültigen PC/IXF-Spalte ignoriert, und die Tabellenspaltenwerte werden auf NULL gesetzt.
  - Wenn die Zieltabellenspalte keine Nullwerte enthalten darf, wird das Dienstprogramm IMPORT beendet. Die gesamte PC/IXF-Datei wird zurückgewiesen, und es werden keine Daten importiert. Die vorhandene Tabelle bleibt unverändert.
- Beim Import in eine neue Tabelle bewirken PC/IXF-Spalten, die Nullwerte enthalten dürfen, die Generierung von Datenbankspalten, die Nullwerte enthalten dürfen. PC/IXF-Spalten, die keine Nullwerte enthalten dürfen, bewirken die Generierung von Datenbankspalten, die keine Nullwerte enthalten dürfen.
- Eine PC/IXF-Spalte, die keine Nullwerte enthalten darf, kann in eine Datenbankspalte, die Nullwerte enthalten darf, importiert werden.
- Eine PC/IXF-Spalte, die Nullwerte enthalten darf, kann in eine Datenbankspalte, die keine Nullwerte enthalten darf, importiert werden. Wenn ein NULL-Wert in der PC/IXF-Spalte festgestellt wird, weist das Dienstprogramm IMPORT die Werte aller Spalten in der PC/IXF-Zeile zurück, die den NULL-Wert enthält (die

ganze Zeile wird zurückgewiesen), und die Verarbeitung wird bei der nächsten PC/IXF-Zeile fortgesetzt. Das bedeutet, dass keine Daten aus einer PC/IXF-Zeile, die einen NULL-Wert enthält, importiert werden, wenn eine Zieltabellenspalte keine Nullwerte enthalten darf.

- Inkompatible Spalten - neue Tabelle

  Wenn beim Import in eine *neue* Datenbanktabelle eine PC/IXF-Spalte ausgewählt wird, die mit der Zieldatenbankspalte inkompatibel ist, wird das Dienstprogramm IMPORT beendet. Die gesamte PC/IXF-Datei wird zurückgewiesen, es wird keine Tabelle erstellt, und es werden keine Daten importiert.

  Anmerkung: Die **IMPORT**-Option FORCEIN erweitert den Bereich kompatibler Spalten.

- Inkompatible Spalten - vorhandene Tabelle

  Wenn beim Import in eine *vorhandene* Datenbanktabelle eine PC/IXF-Spalte ausgewählt wird, die zur Zieldatenbankspalte inkompatibel ist, ist eine von zwei Aktionen möglich:

  – Wenn die Zieltabellenspalte Nullwerte enthalten darf, werden alle Werte der PC/IXF-Spalte ignoriert, und die Tabellenspaltenwerte werden auf NULL gesetzt.
  – Wenn die Zieltabellenspalte keine Nullwerte enthalten darf, wird das Dienstprogramm IMPORT beendet. Die gesamte PC/IXF-Datei wird zurückgewiesen, und es werden keine Daten importiert. Die vorhandene Tabelle bleibt unverändert.

  Anmerkung: Die **IMPORT**-Option FORCEIN erweitert den Bereich kompatibler Spalten.

- Ungültige Werte

  Wenn beim Import ein PC/IXF-Spaltenwert festgestellt wird, der für die Zieldatenbankspalte ungültig ist, weist das Dienstprogramm IMPORT die Werte aller Spalten in der PC/IXF-Zeile mit dem ungültigen Wert zurück, d. h. die gesamte Zeile wird zurückgewiesen, und die Verarbeitung wird bei der nächste PC/IXF-Zeile fortgesetzt.

Datentypbezogene Regeln für den Import von PC/IXF-Dateien in Datenbanken:

- Eine gültige numerische PC/IXF-Spalte kann in jede beliebige kompatible numerische Datenbankspalte importiert werden. PC/IXF-Spalten, die 4 Byte große Gleitkommadaten enthalten, werden nicht importiert, weil dies ein ungültiger Datentyp ist.
- Datum-/Uhrzeitspalten der Datenbank können Werte aus entsprechenden PC/IXF-Datums-/-Uhrzeitspalten (DATE, TIME und TIMESTAMP) sowie aus anderen PC/IXF-Zeichenspalten (CHAR, VARCHAR und LONG VARCHAR) akzeptieren, wobei jedoch die Einschränkungen hinsichtlich der Spaltenlänge und der Wertekompatibilität gelten.
- Eine gültige PC/IXF-Zeichenspalte (CHAR, VARCHAR oder LONG VARCHAR) kann immer in eine *vorhandene* Datenbankzeichenspalte mit der Markierung FOR BIT DATA importiert werden. Ansonsten gilt Folgendes:

  – IXFCSBCP und die SBCS-CPGID müssen übereinstimmen.
  – Es muss eine Konvertierungstabelle für IXFCSBCP/IXFCDBCP und SBCS/DBCS vorhanden sein.
  – Ein Satz muss ausschließlich aus Nullen bestehen (FOR BIT DATA).

  Wenn IXFCSBCP nicht null ist, muss der Wert von IXFCDBCP entweder null sein oder mit der DBCS-CPGID der Zieldatenbankspalte übereinstimmen.

Wenn eine dieser Bedingungen nicht erfüllt ist, sind die PC/IXF-Spalten und die Datenbankspalten inkompatibel.

Beim Importieren einer gültigen PC/IXF-Zeichenspalte in eine *neue* Datenbanktabelle muss der Wert von IXFCSBCP null sein oder mit der SBCS-CPGID der Datenbank übereinstimmen, oder es ist eine Konvertierungstabelle erforderlich. Wenn IXFCSBCP null ist, muss auch IXFCDBCP null sein (andernfalls hat die PC/IXF-Spalte einen ungültigen Datentyp). Von IMPORT wird eine Zeichenspalte mit der Markierung FOR BIT DATA in der neuen Tabelle erstellt. Wenn IXFCSBCP nicht null ist und mit der SBCS-CPGID der Datenbank übereinstimmt, muss der Wert von IXFCDBCP entweder null sein oder mit der DBCS-CPGID der Datenbank übereinstimmen. In diesem Fall wird vom Dienstprogramm eine Zeichenspalte in der neuen Tabelle erstellt, wobei die SBCS- und DBCS-CPGID-Werte mit denen der Datenbank übereinstimmen. Wenn diese Bedingungen nicht erfüllt sind, sind die PC/IXF-Spalten und die Datenbankspalten inkompatibel.

Die Option FORCEIN kann verwendet werden, um Codepage-Gleichheitsüberprüfungen zu überschreiben. Eine PC/IXF-Zeichenspalte mit IXFCSBCP gleich null und IXFCDBCP ungleich null ist jedoch ein ungültiger Datentyp, der nicht importiert werden kann, auch dann nicht, wenn FORCEIN angegeben ist.

- Eine gültige PC/IXF-Grafikspalte (GRAPHIC, VARGRAPHIC oder LONG VARGRAPHIC) kann immer in eine *vorhandene* Datenbankzeichenspalte mit der Markierung FOR BIT DATA importiert werden, ist jedoch mit allen anderen Datenbankspalten inkompatibel. Die Option FORCEIN kann verwendet werden, um die Einschränkung abzuschwächen. Eine PC/IXF-Grafikspalte mit IXFCSBCP ungleich null und IXFCDBCP gleich null ist jedoch ein ungültiger Datentyp, der nicht importiert werden kann, auch dann nicht, wenn FORCEIN angegeben ist.

  Beim Import einer gültigen PC/IXF-Grafikspalte in eine Datenbankgrafikspalte muss der Wert von IXFCDBCP mit der DBCS-CPGID der Zieldatenbankspalte übereinstimmen (das heißt, die Doppelbyte-Codepages der beiden Spalten müssen übereinstimmen).
- Wenn beim Import einer PC/IXF-Datei in eine vorhandene Datenbanktabelle eine Zeichenfolgespalte mit fester Länge (CHAR oder GRAPHIC) ausgewählt wird, deren Länge die maximale Länge der Zielspalte überschreitet, sind die Spalten inkompatibel.
- Wenn beim Import einer PC/IXF-Datei in eine vorhandene Datenbanktabelle eine Zeichenfolgespalte mit variabler Länge (VARCHAR, LONG VARCHAR, VARGRAPHIC oder LONG VARGRAPHIC) ausgewählt wird, deren Länge die maximale Länge der Zielspalte überschreitet, *sind* die Spalten kompatibel. Einzelne Werte werden anhand der Kompatibilitätsregeln verarbeitet, die für die INSERT-Anweisung des Datenbankmanagers gelten. PC/IXF-Werte, die zu lang für die Zieldatenbankspalte sind, sind ungültig.
- PC/IXF-Werte, die in eine *Zeichen*spalte der Datenbank mit fester Länge (d. h. eine CHAR-Spalte) importiert werden, werden nach Bedarf rechts mit Einzelbyteleerzeichen (0x20) aufgefüllt, um Werte zu erhalten, deren Länge mit der der Datenbankspalte übereinstimmt. PC/IXF-Werte, die in eine *Grafik*spalte der Datenbank mit fester Länge (d. h. eine GRAPHIC-Spalte) importiert werden, werden nach Bedarf rechts mit Doppelbyteleerzeichen (0x8140) aufgefüllt, um Werte zu erhalten, deren Länge mit der der Datenbankspalte übereinstimmt.
- Da PC/IXF-Spalten des Typs VARCHAR maximal 254 Byte lang sein dürfen, muss eine VARCHAR-Spalte der Datenbank mit der maximalen Länge *n* (wobei 254 *n* 4001 gilt) in eine PC/IXF-Spalte LONG VARCHAR mit der maximalen Länge *n* exportiert werden.
- Obwohl PC/IXF-Spalten des Typs LONG VARCHAR maximal 32 767 Byte lang sein dürfen und LONG VARCHAR-Spalten der Datenbank maximal 32 700 Byte

lang sein dürfen, sind PC/IXF-Spalten des Typs LONG VARCHAR, die länger sind als 32 700 Byte (aber kürzer als 32 768 Byte), dennoch gültig und können in LONG VARCHAR-Spalten der Datenbank importiert werden. Allerdings können hierbei Daten verloren gehen.

- Da PC/IXF-Spalten des Typs VARGRAPHIC maximal 127 Byte lang sein dürfen, muss eine VARGRAPHIC-Spalte der Datenbank mit der maximalen Länge *n* (wobei 127 *n* 2001 gilt) in eine PC/IXF-Spalte LONG VARGRAPHIC der maximalen Länge *n* exportiert werden.
- Obwohl PC/IXF-Spalten des Typs LONG VARGRAPHIC maximal 16 383 Byte lang sein dürfen und LONG VARGRAPHIC-Spalten der Datenbank maximal 16 350 Byte lang sein dürfen, sind PC/IXF-Spalten des Typs LONG VARGRAPHIC, die länger sind als 16 350 Byte (aber kürzer als 16 384 Byte), dennoch gültig und können in LONG VARGRAPHIC-Spalten der Datenbank importiert werden. Allerdings können hierbei Daten verloren gehen.

Tabelle 26 und Tabelle 27 bieten eine Übersicht über den Import von PC/IXF-Dateien in neue oder vorhandene Datenbanktabellen ohne Verwendung der Option FORCEIN.

Tabelle 26. Übersicht über den Import von PC/IXF-Dateien ohne Verwendung der Option FORCEIN - numerische Datentypen

PC/IXF-SPALTENDATENTYP	SPALTENDATENTYP DER DATENBANK					
	SMALL INT	**INT**	**BIGINT**	**DEC**	**DFP**	**FLT**
-SMALLINT	N					
	V	V	V	V[a]	V	V
-INTEGER		N				
	V[a]	V	V	V[a]	V	V
-BIGINT			N			
	V[a]	V[a]	V	V[a]	V	V
-DECIMAL				N		
	V[a]	V[a]	V[a]	V[a]	V	V
-DECFLOAT					N	
	V[a]	V[a]	V[a]	V[a]	V	V[a]
-FLOAT						N
	V[a]	V[a]	V[a]	V[a]	V	V

[a] Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich für den numerischen Zieldatentyp liegen.

Tabelle 27. Übersicht über den Import von PC/IXF-Dateien ohne Verwendung der Option FORCEIN - Zeichendatentypen, grafische Datentypen und Datentypen für Uhrzeit/Datum

PC/IXF-SPALTENDATENTYP	SPALTENDATENTYP DER DATENBANK						
	(0,0)	**(SBCS, 0)[d]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
-(0,0)	N						
	V				V[c]	V[c]	V[c]
-(SBCS,0)		N	N				
	V	V	V		V[c]	V[c]	V[c]

Tabelle 27. Übersicht über den Import von PC/IXF-Dateien ohne Verwendung der Option FORCEIN - Zeichendatentypen, grafische Datentypen und Datentypen für Uhrzeit/Datum (Forts.)

PC/IXF-SPALTENDATENTYP	SPALTENDATENTYP DER DATENBANK						
	(0,0)	(SBCS, 0)[d]	(SBCS, DBCS)[b]	GRAPH[b]	DATE	TIME	TIME STAMP
-(SBCS, DBCS)			N		V[c]	V[c]	V[c]
	V		V				
-GRAPHIC				N			
	V			V			
-DATE					N		
					V		
-TIME						N	
						V	
-TIME STAMP							N
							V

[b] Datentyp ist nur in DBCS-Umgebungen verfügbar.
[c] Einzelne Werte werden zurückgewiesen, wenn sie keine gültigen Datums- oder Uhrzeitwerte darstellen.
[d] Datentyp ist in DBCS-Umgebungen nicht verfügbar.

Anmerkung:

1. Die Tabelle ist eine Matrix aller gültigen PC/IXF- und Datenbankmanagerdatentypen. Wenn eine PC/IXF-Spalte in eine Datenbankspalte importiert werden kann, wird in der Matrixzelle am Schnittpunkt der Matrixzeile des PC/IXF-Datentyps und der Matrixspalte des Datenbankmanager-Datentyps ein Buchstabe angezeigt. Ein "N" gibt an, dass das Dienstprogramm eine neue Datenbanktabelle erstellt (es wird eine Datenbankspalte des angegebenen Datentyps erstellt). Ein "V" gibt an, dass das Dienstprogramm Daten in eine vorhandene Datenbanktabelle importiert (eine Datenbankspalte des angegebenen Datentyps ist ein gültiges Ziel).
2. Zeichenfolgedatentypen werden anhand von Codepage-Attributen unterschieden. Diese Attribute werden als geordnetes Paar (SBCS,DBCS) angezeigt, wobei Folgendes gilt:
   - SBCS ist entweder null oder gibt einen Wert ungleich null eines Einzelbyte-Codepage-Attributs des Zeichendatentyps an.
   - DBCS ist entweder null oder gibt einen Wert ungleich null eines Doppelbyte-Codepage-Attributs des Zeichendatentyps an.
3. Wenn die Tabelle angibt, dass eine PC/IXF-Zeichenspalte in eine Datenbankzeichenspalte importiert werden kann, entsprechen die Werte der jeweiligen Codepage-Attributepaare den Regeln, die für die Gleichheit von Codepages gelten.

Unterschiede zwischen PC/IXF und Version 0 System/370 IXF:
Im Folgenden werden die Unterschiede zwischen PC/IXF, das vom Datenbankmanager verwendet wird, und Version 0 System/370 IXF, das von mehreren Host-Datenbankprodukten verwendet wird, beschrieben:

- PC/IXF-Dateien orientieren sich an ASCII und nicht an EBCDIC. PC/IXF-Dateien haben eine erheblich erweiterte Codepage-Kennzeichnung, einschließlich neuer Codepage-Kennungen im H-Datensatz und der Verwendung von Codepage-Werten in den Spaltendeskriptorsätzen. Es gibt auch einen Mechanismus zur Markierung von Spalten mit Zeichendaten als FOR BIT DATA. FOR BIT DATA-

Spalten sind von besonderer Bedeutung, weil Umsetzungen, die ein PC/IXF-Dateiformat in ein anderes IXF- oder Datenbankdateiformat oder umgekehrt konvertieren, keine Codepage-Umsetzung der Werte durchführen können, die in FOR BIT DATA-Spalten enthalten sind.

- Es ist nur das Maschinendatenformat zulässig, d. h. das Feld IXFTFORM muss immer den Wert `M` enthalten. Weiterhin müssen die Maschinendaten in PC-Formaten vorliegen. Das bedeutet, dass das Feld IXFTMFRM den Wert `PC` enthalten muss. Das bedeutet, dass ganze Zahlen, Gleitkommazahlen und Dezimalzahlen in Datenteilen von PC/IXF-Datensätzen PC-Formate aufweisen müssen.
- Anwendungssätze (A) sind nach dem H-Datensatz in einer PC/IXF-Datei zulässig. Sie werden nicht gezählt, wenn der Wert des Feldes IXFHHCNT berechnet wird.
- Jeder PC/IXF-Datensatz beginnt mit einem Satzlängenanzeiger. Es handelt sich dabei um eine 6 Byte große Zeichendarstellung eines ganzzahligen Werts, die die Länge des PC/IXF-Datensatzes in Byte enthält (ohne Einschluss des Satzlängenanzeigers), d. h. die gesamte Satzlänge minus 6 Byte. Der Zweck des Satzlängenfeldes besteht darin, PC-Programmen die Erkennung von Satzbegrenzungen zu ermöglichen.
- PC/IXF unterstützt keine Version 0 IXF X-Sätze (jedoch D-Satzkennungen), um die kompakte Speicherung von Daten mit variabler Länge zu erleichtern und eine komplexe Verarbeitung zu vermeiden, wenn ein Feld in mehrere Sätze aufgeteilt ist. Wenn ein Feld mit variabler Länge oder ein Feld, das Nullwerte enthalten darf, das letzte Feld in einem D-Datensatz ist, ist es nicht nötig, die gesamte maximale Länge des Feldes in die PC/IXF-Datei zu schreiben.

Option FORCEIN:

Der Änderungswert `forcein` für den Dateityp ermöglicht den Import einer PC/IXF-Datei trotz Codepage-Unterschieden zwischen Daten in der PC/IXF-Datei und der Zieldatenbank. Sie bietet zusätzliche Flexibilität bei der Definition kompatibler Spalten.

Allgemeine Semantikregeln für forcein

Für die Verwendung des Änderungswerts für den Dateityp `forcein` in einer SBCS- oder DBCS-Umgebung gilt die folgende allgemeine Semantik:

- Der Änderungswert `forcein` für den Dateityp sollte mit Bedacht verwendet werden. Es ist zumeist ratsam, zu versuchen, einen Import ohne Aktivierung dieser Option durchzuführen. Aufgrund der generischen Beschaffenheit der PC/IXF-Datenaustauscharchitektur können einige PC/IXF-Dateien jedoch möglicherweise Datentypen oder Werte enthalten, die nicht ohne Eingriff importiert werden können.
- Der Import mit `forcein` in eine *neue* Tabelle kann unter Umständen zu einem anderen Ergebnis führen als der Import in eine vorhandene Tabelle. Eine vorhandene Tabelle enthält vordefinierte Zieldatentypen für jeden PC/IXF-Datentyp.
- Wenn LOB-Daten mit dem Änderungswert `lobsinfile` für den Dateityp exportiert und die Dateien auf einen anderen Client mit einer anderen Codepage versetzt werden, werden die CLOBS und DBCLOBS in den separaten Dateien anders als andere Daten beim Importieren oder Laden in eine Datenbank nicht in die Client-Codepage konvertiert.

Codepage-Semantikregeln für forcein

Für die Verwendung des Änderungswerts `forcein` für den Dateityp in einer SBCS- oder DBCS-Umgebung gilt die folgende Codepage-Semantik:

- Der Änderungswert `forcein` für den Dateityp inaktiviert alle Codepage-Vergleiche des Dienstprogramms IMPORT.

  Diese Regel gilt für Codepage-Vergleiche auf Spalten- und auch auf Dateiebene, wenn in eine neue oder vorhandene Datenbanktabelle importiert wird. Auf der Spaltenebene (z. B. Datentyp) gilt diese Regel nur für die folgenden Datenbankmanager- und PC/IXF-Datentypen: Zeichen (CHAR, VARCHAR und LONG VARCHAR) und Grafik (GRAPHIC, VARGRAPHIC und LONG VARGRAPHIC). Die Einschränkung ergibt sich daraus, dass die Codepage-Attribute anderer Datentypen für die Auswertung der Datentypwerte nicht von Bedeutung sind.
- `forcein` inaktiviert nicht die Prüfung der Codepage-Attribute zur Ermittlung der Datentypen.

  Beispiel: Der Datenbankmanager erlaubt die Deklaration einer CHAR-Spalte mit dem Attribut FOR BIT DATA. Eine solche Deklaration setzt sowohl die SBCS-CPGID als auch die DBCS-CPGID der Spalte auf null. Es ist der Nullwert dieser CPGIDs, der die Spaltenwerte als Bitfolgen (statt als Zeichenfolgen) angibt.
- `forcein` impliziert keine Codepage-Umsetzung.

  Werte mit Datentypen, auf die sich der Änderungswert `forcein` für den Dateityp auswirkt, werden "unverändert" kopiert. Es werden keine Codepunktzuordnungen angewendet, um eine Änderung der Codepage-Umgebungen zu berücksichtigen. Das Auffüllen des importierten Werts mit Leerzeichen kann im Falle von Zielspalten mit fester Länge erforderlich sein.
- Wenn Daten mit `forcein` in eine *vorhandene* Tabelle importiert werden, gilt Folgendes:
  - Der Codepage-Wert der Zieldatenbanktabelle und -spalten ist immer vorrangig.
  - Der Codepage-Wert der PC/IXF-Datei und -Spalten wird ignoriert.

  Diese Regel gilt unabhängig davon, ob `forcein` verwendet wird. Der Datenbankmanager lässt nach der Erstellung der Datenbank keine Änderungen an einem Codepage-Wert einer Datenbank oder Spalte zu.
- Beim Importieren in eine *neue* Tabelle unter Verwendung von `forcein` gilt Folgendes:
  - Der Codepage-Wert der Zieldatenbank ist immer vorrangig.
  - PC/IXF-Zeichenspalten mit IXFCSBCP = IXFCDBCP = 0 generieren Tabellenspalten mit dem Attribut FOR BIT DATA.
  - Alle anderen PC/IXF-Zeichenspalten generieren Tabellenzeichenspalten mit SBCS- und DBCS-CPGID-Werten, die mit denen der Datenbank identisch sind.
  - PC/IXF-Grafikspalten generieren Tabellengrafikspalten mit der SBCS-CPGID "undefiniert" und einer DBCS-CPGID, die mit der der Datenbank identisch ist (nur DBCS-Umgebung).

Angenommen, Sie haben eine PC/IXF-Spalte des Typs CHAR mit IXFCSBCP = "00897" und IXFCDBCP = "00301". Diese Spalte soll in eine Datenbankspalte des Typs CHAR mit SBCS-CPGID = "00850" und DBCS-CPGID = "00000" importiert werden. Ohne `forcein` wird das Dienstprogramm beendet, und es werden keine Daten importiert, oder aber die PC/IXF-Spaltenwerte werden ignoriert und die Datenbankspalte enthält NULL-Werte (wenn die Datenbankspalte Nullwerte enthalten darf). Mit `forcein` wird das Dienstprogramm fortgesetzt, und Codepage-Inkompa-

tibilitäten werden ignoriert. Wenn keine weiteren Datentypinkompatibilitäten (wie beispielsweise die Länge) auftreten, werden die Werte der PC/IXF-Spalte "unverändert" importiert und stehen zur Auswertung unter der Codepage-Umgebung der Datenbank zur Verfügung.

An den folgenden Tabellen können Sie Folgendes ablesen:

- Die Codepage-Attribute einer Spalte, die in einer *neuen* Datenbanktabelle erstellt wird, wenn aus der PC/IXF-Datei ein Datentyp mit angegebenen Codepage-Attributen importiert wird.
- Dass das Dienstprogramm IMPORT die PC/IXF-Datentypen zurückweist, wenn sie ungültig oder inkompatibel sind.

Tabelle 28. Übersicht über die Codepage-Semantik des Dienstprogramms IMPORT (neue Tabelle) für SBCS

CODEPAGE-ATTRIBUTE DES PC/IXF-DATENTYPS	CODEPAGE-ATTRIBUTE DER DATENBANKTABELLENSPALTE	
	Ohne `forcein`	**Mit `forcein`**
(0,0)	(0,0)	(0,0)
(a,0)	(a,0)	(a,0)
(x,0)	zurückweisen	(a,0)
(x,y)	zurückweisen	(a,0)
(a,y)	zurückweisen	(a,0)
(0,y)	zurückweisen	(0,0)

Anmerkung:

1. In dieser Tabelle wird davon ausgegangen, dass es keine Konvertierungstabelle zwischen a und x gibt. Gäbe es eine solche, würden die Elemente 3 und 4 auch ohne Verwendung von `forcein` funktionieren.
2. Siehe Anmerkungen zu Tabelle 29.

Tabelle 29. Übersicht über die Codepage-Semantik des Dienstprogramms IMPORT (neue Tabelle) für DBCS

CODEPAGE-ATTRIBUTE DES PC/IXF-DATENTYPS	CODEPAGE-ATTRIBUTE DER DATENBANKTABELLENSPALTE	
	Ohne `forcein`	**Mit `forcein`**
(0,0)	(0,0)	(0,0)
(a,0)	(a,b)	(a,b)
(x,0)	zurückweisen	(a,b)
(a,b)	(a,b)	(a,b)
(x,y)	zurückweisen	(a,b)
(a,y)	zurückweisen	(a,b)
(x,b)	zurückweisen	(a,b)
(0,b)	(-,b)	(-,b)
(0,y)	zurückweisen	(-,b)

Anmerkung:

1. In dieser Tabelle wird davon ausgegangen, dass es keine Konvertierungstabelle zwischen a und x gibt.

2. Codepage-Attribute eines PC/IXF-Datentyps werden als geordnetes Paar gezeigt, wobei x für einen Einzelbyte-Codepage-Wert ungleich null und y für einen Doppelbyte-Codepage-Wert ungleich null steht. Das Zeichen '-' steht für einen undefinierten Codepage-Wert.
3. Die Verwendung verschiedener Buchstaben in verschiedenen Codepage-Attributpaaren ist willkürlich. Verschiedene Buchstaben implizieren verschiedene Werte. Beispiel: Wenn ein PC/IXF-Datentyp als (x,y) gezeigt wird und die Datenbankspalte als (a,y), entspricht x nicht a, die PC/IXF-Datei und die Datenbank haben jedoch den gleichen Doppelbyte-Codepage-Wert y.
4. Nur Zeichen- und Grafikdatentypen sind von der `forcein`-Codepage-Semantik betroffen.
5. Es wird angenommen, dass die Datenbank, die die neue Tabelle enthält, die Codepage-Attribute (a,0) hat. Demzufolge müssen alle Zeichenspalten in der neuen Tabelle die Codepage-Attribute (0,0) oder (a,0) haben.

   In einer DBCS-Umgebung wird angenommen, dass die Datenbank, die die neue Tabelle enthält, die Codepage-Attribute (a,b) hat. Demzufolge müssen alle Grafikspalten in der neuen Tabelle die Codepage-Attribute (-,b) und alle Zeichenspalten die Codepage-Attribute (a,b) haben. Die SBCS-CPGID wird als '-' gezeigt, weil sie für Grafikdatentypen nicht definiert ist.
6. Der Datentyp des Ergebnisses wird durch die im Abschnitt "Datentypsemantik für `forcein`" beschriebenen Regeln bestimmt.
7. Das Ergebnis `zurückweisen` ist eine Folge der Regeln für ungültige oder inkompatible Datentypen.

An den folgenden Tabellen können Sie Folgendes ablesen:

- Dass das Dienstprogramm IMPORT PC/IXF-Datentypen mit verschiedenen Codepage-Attributen in einer *vorhandenen* Tabellenspalte (der *Ziel*spalte) mit den angegebenen Codepage-Attributen akzeptiert.
- Dass das Dienstprogramm IMPORT den Import eines PC/IXF-Datentyps mit bestimmten Codepage-Attributen in eine *vorhandene* Tabellenspalte mit den gezeigten Codepage-Attributen nicht zulässt. Das Dienstprogramm weist die PC/IXF-Datentypen zurück, wenn sie ungültig oder inkompatibel sind.

Tabelle 30. Übersicht über die Codepage-Semantik des Dienstprogramms IMPORT (vorhandene Tabelle) für SBCS

CODEPAGE-ATTRIBUTE DES PC/IXF-DATENTYPS	CODEPAGE-ATTRIBUTE DER ZIELDATENBANKSPALTE	ERGEBNISSE DES IMPORTS	
		Ohne `forcein`	Mit `forcein`
(0,0)	(0,0)	akzeptieren	akzeptieren
(a,0)	(0,0)	akzeptieren	akzeptieren
(x,0)	(0,0)	akzeptieren	akzeptieren
(x,y)	(0,0)	akzeptieren	akzeptieren
(a,y)	(0,0)	akzeptieren	akzeptieren
(0,y)	(0,0)	akzeptieren	akzeptieren
(0,0)	(a,0)	Null oder zurückweisen	akzeptieren
(a,0)	(a,0)	akzeptieren	akzeptieren
(x,0)	(a,0)	Null oder zurückweisen	akzeptieren

Tabelle 30. Übersicht über die Codepage-Semantik des Dienstprogramms IMPORT (vorhandene Tabelle) für SBCS (Forts.)

CODEPAGE-ATTRIBUTE DES PC/IXF-DATENTYPS	CODEPAGE-ATTRIBUTE DER ZIELDATENBANKSPALTE	ERGEBNISSE DES IMPORTS	
		Ohne `forcein`	Mit `forcein`
(x,y)	(a,0)	Null oder zurückweisen	akzeptieren
(a,y)	(a,0)	Null oder zurückweisen	akzeptieren
(0,y)	(a,0)	Null oder zurückweisen	Null oder zurückweisen

Anmerkung:

- In dieser Tabelle wird davon ausgegangen, dass es keine Konvertierungstabelle zwischen a und x gibt.
- Siehe Anmerkungen zu Tabelle 28 auf Seite 279.
- Das Ergebnis `Null oder zurückweisen` ist eine Folge der Regeln für ungültige oder inkompatible Datentypen.

Tabelle 31. Übersicht über die Codepage-Semantik des Dienstprogramms IMPORT (vorhandene Tabelle) für DBCS

CODEPAGE-ATTRIBUTE DES PC/IXF-DATENTYPS	CODEPAGE-ATTRIBUTE DER ZIELDATENBANKSPALTE	ERGEBNISSE DES IMPORTS	
		Ohne `forcein`	Mit `forcein`
(0,0)	(0,0)	akzeptieren	akzeptieren
(a,0)	(0,0)	akzeptieren	akzeptieren
(x,0)	(0,0)	akzeptieren	akzeptieren
(a,b)	(0,0)	akzeptieren	akzeptieren
(x,y)	(0,0)	akzeptieren	akzeptieren
(a,y)	(0,0)	akzeptieren	akzeptieren
(x,b)	(0,0)	akzeptieren	akzeptieren
(0,b)	(0,0)	akzeptieren	akzeptieren
(0,y)	(0,0)	akzeptieren	akzeptieren
(0,0)	(a,b)	Null oder zurückweisen	akzeptieren
(a,0)	(a,b)	akzeptieren	akzeptieren
(x,0)	(a,b)	Null oder zurückweisen	akzeptieren
(a,b)	(a,b)	akzeptieren	akzeptieren
(x,y)	(a,b)	Null oder zurückweisen	akzeptieren
(a,y)	(a,b)	Null oder zurückweisen	akzeptieren
(x,b)	(a,b)	Null oder zurückweisen	akzeptieren

Tabelle 31. Übersicht über die Codepage-Semantik des Dienstprogramms IMPORT (vorhandene Tabelle) für DBCS (Forts.)

CODEPAGE-ATTRIBUTE DES PC/IXF-DATENTYPS	CODEPAGE-ATTRIBUTE DER ZIELDATENBANKSPALTE	ERGEBNISSE DES IMPORTS	
		Ohne `forcein`	Mit `forcein`
(0,b)	(a,b)	Null oder zurückweisen	Null oder zurückweisen
(0,y)	(a,b)	Null oder zurückweisen	Null oder zurückweisen
(0,0)	(-,b)	Null oder zurückweisen	akzeptieren
(a,0)	(-,b)	Null oder zurückweisen	Null oder zurückweisen
(x,0)	(-,b)	Null oder zurückweisen	Null oder zurückweisen
(a,b)	(-,b)	Null oder zurückweisen	Null oder zurückweisen
(x,y)	(-,b)	Null oder zurückweisen	Null oder zurückweisen
(a,y)	(-,b)	Null oder zurückweisen	Null oder zurückweisen
(x,b)	(-,b)	Null oder zurückweisen	Null oder zurückweisen
(0,b)	(-,b)	akzeptieren	akzeptieren
(0,y)	(-,b)	Null oder zurückweisen	akzeptieren

Anmerkung:

- In dieser Tabelle wird davon ausgegangen, dass es keine Konvertierungstabelle zwischen a und x gibt.
- Siehe Anmerkungen zu Tabelle 28 auf Seite 279.
- Das Ergebnis `Null oder zurückweisen` ist eine Folge der Regeln für ungültige oder inkompatible Datentypen.

Datentypsemantikregeln für forcein

Der Änderungswert `forcein` für den Dateityp ermöglicht den Import bestimmter PC/IXF-Spalten in Zieldatenbankspalten mit ungleichen oder anderweitig inkompatiblen Datentypen. Für die Verwendung von `forcein` in einer SBCS- oder DBCS-Umgebung gilt die folgende Datentypsemantik (sofern nicht anders angegeben):

- In SBCS-Umgebungen ermöglicht `forcein` folgende Importvarianten:
  - Ein PC/IXF-Datentyp BIT (IXFCSBCP = 0 = IXFCDBCP für eine PC/IXF-Zeichenspalte) in eine Datenbankzeichenspalte (SBCS-CPGID ungleich null und DBCS-CPGID gleich null); nur vorhandene Tabellen
  - Ein PC/IXF-Datentyp MIXED (IXFCSBCP und IXFCDBCP ungleich null) in eine Datenbankzeichenspalte; neue und vorhandene Tabellen
  - Ein PC/IXF-Datentyp GRAPHIC in eine Datenbankspalte mit dem Attribut FOR BIT DATA (SBCS-CPGID = 0 = DBCS-CPGID); nur neue Tabellen (dies ist für vorhandene Tabellen immer zulässig)

- Mit dem Änderungswert `forcein` für den Dateityp wird der Bereich gültiger PC/IXF-Datentypen nicht erweitert.

  PC/IXF-Spalten mit Datentypen, die nicht als gültige PC/IXF-Datentypen definiert sind, sind mit oder ohne `forcein` für den Import ungültig.
- In DBCS-Umgebungen ermöglicht `forcein` folgende Importvarianten:
  - Ein PC/IXF-Datentyp BIT in eine Datenbankzeichenspalte
  - Ein PC/IXF-Datentyp BIT in eine Grafikspalte der Datenbank. Wenn die PC/IXF-Spalte des Typs BIT eine feste Länge besitzt, muss die Länge ein gerader Wert sein. Eine PC/IXF-Spalte des Typs BIT mit fester Länge und einem ungeraden Längenwert ist mit einer Grafikspalte der Datenbank nicht kompatibel. Eine PC/IXF-Spalte des Typs BIT mit variabler Länge *ist* kompatibel, unabhängig davon, ob der Längenwert gerade oder ungerade ist, obwohl ein ungerader Längenwert aus einer Spalte mit variabler Länge ein ungültiger Wert für den Import in eine Grafikspalte der Datenbank ist.
  - Ein PC/IXF-Datentyp MIXED in eine Datenbankzeichenspalte

Tabelle 32 bietet eine Übersicht des Imports von PC/IXF-Dateien in neue oder vorhandene Datenbanktabellen unter Angabe von `forcein`.

Tabelle 32. Übersicht des Imports von PC/IXF-Dateien mit `forcein`

PC/IXF-SPALTEN-DATENTYP	SPALTENDATENTYP DER DATENBANK											
	SMALL INT	**INT**	**BIGINT**	**DEC**	**FLT**	**(0,0)**	**(SBCS, 0)[e]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
-SMALLINT	N											
	V	V	V	V[a]	V							
-INTEGER		N										
	V[a]	V	V	V[a]	V							
-BIGINT			N									
	V[a]	V[a]	V	V[a]	V							
-DECIMAL				N								
	V[a]	V[a]	V[a]	V[a]	V							
-FLOAT					N							
	V[a]	V[a]	V[a]	V[a]	V							
-(0,0)												
						N						
-(SBCS,0)						V	V mit F	V mit F	V mit F	V[c]	V[c]	V[c]
							N	N				
-(SBCS, DBCS)						V	V	V		V[c]	V[c]	V[c]
							N mit F[d]	N		V[c]	V[c]	V[c]
						V	V mit F	V				
-GRAPHIC						N mit F[d]			N			
						V			V			
-DATE										N		
										V		
-TIME											N	
											V	
-TIME STAMP												N
												V

[a] Einzelne Werte werden zurückgewiesen, wenn sie nicht im Bereich für den numerischen Zieldatentyp liegen.
[b] Datentyp ist nur in DBCS-Umgebungen verfügbar.
[c] Einzelne Werte werden zurückgewiesen, wenn sie keine gültigen Datums- oder Uhrzeitwerte darstellen.
[d] Gilt nur, wenn der PC/IXF-Quellendatentyp nicht von der Zieldatenbank unterstützt wird.
[e] Datentyp ist in DBCS-Umgebungen nicht verfügbar.

Anmerkung: Wenn eine a PC/IXF-Spalte nur mit der `forcein` in eine Datenbankspalte importiert werden kann, wird die Zeichenfolge "mit F" zusammen mit "N" oder "V" gezeigt. Ein "N" gibt an, dass das Dienstprogramm eine neue Datenbanktabelle erstellt. Ein "V" gibt an, dass das Dienstprogramm Daten in eine vorhandene Datenbanktabelle importiert. Der Änderungswert `forcein` für den Dateityp wirkt sich nur auf die Kompatibilität von Zeichen- und Grafikdatentypen aus.

Aspekte von Unicode beim Versetzen von Daten

Die Dienstprogramme EXPORT, IMPORT und LOAD werden nicht unterstützt, wenn sie zusammen mit einem Unicode-Client eingesetzt werden, der mit einer Datenbank verbunden ist, die nicht das Unicode-Format verwendet.

Die DEL-, ASC- und PC/IXF-Dateiformate werden - wie in diesem Abschnitt noch erläutert wird - bei einer Unicode-Datenbank unterstützt.

Beim Exportieren von Daten aus einer Unicode-Datenbank in eine ASCII-Datei ohne universelle Zeilenbegrenzer werden alle Zeichendaten in die Codepage der Anwendung konvertiert. Sowohl Zeichenfolgedaten als auch Grafikzeichenfolgedaten werden in dieselbe SBCS- oder MBCS-Codepage des Clients konvertiert. Dies ist das erwartete Verhalten beim Export von Datenbanken. Es kann nicht geändert werden, weil die gesamte ASCII-Datei ohne universelle Zeilenbegrenzer nur eine Codepage haben kann. Daher werden beim Export in eine solche ASCII-Datei nur diejenigen UCS-2-Zeichen gesichert, die in der Codepage der Anwendung enthalten sind. Andere Zeichen werden durch das Standardsubstitutionszeichen für die Codepage der Anwendung ersetzt. Bei UTF-8-Clients (Codepage 1208) gehen keine Daten verloren, weil alle UCS-2-Zeichen durch UTF-8-Clients unterstützt werden.

Beim Importieren von Daten aus einer ASCII-Datei (mit dem DEL- oder ASC-Format) in eine Unicode-Datenbank werden Zeichenfolgedaten aus der Codepage der Anwendung in UTF-8 und Grafikzeichenfolgedaten aus der Codepage der Anwendung in UCS-2 konvertiert. Es gehen keine Daten verloren. Wenn Sie ASCII-Daten importieren wollen, die unter einer anderen Codepage gesichert wurden, sollten Sie die Codepage der Datendatei ändern, bevor Sie den Befehl IMPORT absetzen. Sie können die Codepage der Datendatei angeben, indem Sie die Registrierdatenbankvariable **DB2CODEPAGE** auf die Codepage der ASCII-Datendatei setzen oder indem Sie den Änderungswert `codepage` für den Dateityp verwenden.

Der Bereich der gültigen ASCII-Begrenzer für Clients mit Einzelbytezeichensätzen (SBCS) und Mehrbytezeichensätzen (MBCS) entspricht dem Bereich, der gegenwärtig von DB2 for Linux, UNIX and Windows für diese Clients unterstützt wird. Der Bereich der gültigen Begrenzer für UTF-8-Clients erstreckt sich von X'01' bis X'7F', wobei die üblichen Einschränkungen gelten.

Beim Exportieren von Daten aus einer Unicode-Datenbank in eine PC/IXF-Datei werden Zeichenfolgedaten in die SBCS/MBCS-Codepage des Clients konvertiert. Grafikzeichenfolgedaten werden nicht konvertiert, sondern in UCS-2 (Codepage 1200) gespeichert. Es gehen keine Daten verloren.

Beim Importieren von Daten aus einer PC/IXF-Datei in eine Unicode-Datenbank müssen Zeichenfolgedaten in der in den PC/IXF-Kopfdaten gespeicherten SBCS/MBCS-Codepage vorliegen und Grafikzeichenfolgedaten in der DBCS-Codepage, die in den PC/IXF-Kopfdaten gespeichert ist. Zeichenfolgedaten werden durch das Dienstprogramm IMPORT aus der in den PC/IXF-Kopfdaten angegebenen Codepage in die Codepage des Clients konvertiert. Anschließend werden sie durch die Anweisung INSERT von der Client-Codepage in UTF-8 konvertiert. Grafikzeichenfolgedaten werden durch das Dienstprogramm IMPORT aus der DBCS-Codepage, die in den PC/IXF-Kopfdaten angegeben ist, direkt in UCS-2 (Codepage 1200) konvertiert.

Das Dienstprogramm LOAD stellt die Daten direkt in die Datenbank und geht standardmäßig davon aus, dass Daten in ASC- oder DEL-Dateien in der Codepage der Datenbank vorliegen. Daher findet für ASCII-Dateien standardmäßig keine Codepagekonvertierung statt. Wenn die Codepage für die Datendatei explizit angegeben wurde (mit dem Änderungswert `codepage` für den Dateityp), verwendet das Dienstprogramm LOAD diese Informationen, um vor dem Laden der Daten eine Konvertierung aus der angegebenen Codepage in die Datenbankcodepage vorzunehmen. Bei PC/IXF-Dateien konvertiert das Dienstprogramm LOAD die in den IXF-Kopfdaten angegebenen Codepages immer in die Datenbankcodepage (Codepage 1208 bei CHAR und Codepage 1200 bei GRAPHIC).

Die Codepage für DBCLOB-Dateien ist bei UCS-2 immer 1200. Für CLOB-Dateien wird die gleiche Codepage wie für die importierten, geladenen oder exportierten Daten verwendet. Wenn beispielsweise Daten mit dem PC/IXF-Format geladen oder importiert werden, wird davon ausgegangen, dass die CLOB-Datei die in den PC/IXF-Kopfdaten angegebene Codepage verwendet. Falls die DBCLOB-Datei im ASC- oder DEL-Format vorliegt, geht das Dienstprogramm LOAD davon aus, dass die CLOB-Daten die Codepage der Datenbank verwenden. Das Dienstprogramm IMPORT geht hingegen davon aus, dass die Codepage der Clientanwendung verwendet wird.

Der Änderungswert `nochecklengths` wird aus den folgenden Gründen immer bei einer Unicode-Datenbank angegeben:

- SBCS kann mit einer Datenbank verbunden sein, für die keine DBCS-Codepage vorhanden ist.
- Zeichenfolgen im Format UTF-8 haben in der Regel eine andere Länge als Zeichenfolgen in Client-Codepages.

Besonderheiten für die Codepages 1394, 1392 und 5488

Mit den Dienstprogrammen IMPORT, EXPORT und LOAD können Daten aus der Codepage GB18030 für Chinesisch (Codepage-IDs 1392 und 5488) und der Codepage ShiftJISX 0213 für Japanisch (Codepage-ID 1394) an Unicode-Datenbanken von DB2 übertragen werden. Zusätzlich können mit dem Dienstprogramm EXPORT Daten aus Unicode-Datenbanken von DB2 an Daten übertragen werden, die die Codepage GB18030 oder ShiftJIS X0213 verwenden.

Der folgende Befehl lädt beispielsweise die Datendatei `u/jp/user/x0213/data.del` mit dem Format Shift JIS X0213, die sich auf einem fernen verbundenen Client befindet, in die Tabelle MYTABLE:

```
db2 load client from /u/jp/user/x0213/data.del
of del modified by codepage=1394 insert into mytable
```

Hierbei befindet sich die Tabelle MYTABLE in einer Unicode-Datenbank von DB2.

Da nur Verbindungen zwischen einen Unicode-Client und einem Unicode-Server unterstützt werden, müssen Sie entweder einen Unicode-Client verwenden oder die DB2-Registrierdatenbankvariable **DB2CODEPAGE** auf den Wert 1208 setzen, bevor Sie das Dienstprogramm LOAD, IMPORT oder EXPORT verwenden.

Eine Konvertierung aus der Codepage 1394 in Unicode kann zu einer Vergrößerung des Datenumfangs führen. Beispiel: Ein 2-Byte-Zeichen kann in Spalten mit dem Datentyp GRAPHIC in Form von zwei 16-Bit-Unicode-Zeichen gespeichert werden. Sie müssen gewährleisten, dass die Zielspalten in der Unicode-Datenbank groß genug sind, um alle vergrößerten Unicode-Byte aufnehmen zu können.

Inkompatibilität

Bei Anwendungen, die mit einer Unicode-Datenbank verbunden sind, liegen Grafikzeichenfolgedaten immer im Format UCS-2 (Codepage 1200) vor. Bei Anwendungen, die mit Datenbanken verbunden sind, welche ein anderes Format als Unicode verwenden, liegen die Grafikzeichenfolgedaten entweder in der DBCS-Codepage der Anwendung vor oder sind nicht zulässig, falls die Anwendungscodepage SBCS ist. Beispiel: Wenn ein 932-Client mit einer Datenbank verbunden ist, die ein anders Format als Unicode für Japanisch verwendet, liegen die Grafikzeichenfolgedaten in der Codepage 301 vor. Bei Anwendungen des 932-Clients, die mit einer Unicode-Datenbank verbunden sind, liegen die Grafikzeichenfolgedaten in UCS-2-Codierung vor.

Zeichensatz und NLS

Die DB2-Dienstprogramme zum Versetzen von Daten bieten die folgende Unterstützung in der Landessprache:

- Die Dienstprogramme IMPORT und EXPORT stellen eine automatische Konvertierung aus einer Client-Codepage in die Server-Codepage bereit.
- Für das Dienstprogramme LOAD können Daten aus einer beliebigen Codepage in die Server-Codepage konvertiert werden, indem bei DEL- und ASC-Dateien der Änderungswert `codepage` verwendet wird.
- Bei allen Dienstprogrammen werden IXF-Daten automatisch aus ihrer ursprünglichen Codepage (wie in der IXF-Datei gespeichert) in die Server-Codepage konvertiert.

Manchmal kann es aufgrund ungleicher Codepages zu Situationen kommen, in denen die Zeichendaten möglicherweise verlängert oder gekürzt werden. Solche Situationen können bei Zeichensätzen mit erweitertem UNIX-Code (Extended UNIX Code, EUC) und Doppelbytezeichensätzen (DBCS) für Japanisch oder traditionelles Chinesisch auftreten, die möglicherweise verschiedene Längencodierungen für dasselbe Zeichen verwenden. Normalerweise wird die Länge der Eingabedaten vor dem Einlesen von Daten mit der Länge der Zielspalte verglichen. Wenn die Länge der Eingabedaten die Länge der Zielspalte überschreitet, werden Nullwerte in die Spalte eingefügt, wenn für die Spalte Nullwerte zulässig sind. Andernfalls wird die Anforderung zurückgewiesen. Wenn der Änderungswert `nochecklengths` für den Dateityp angegeben ist, wird vorab kein Vergleich durchgeführt, und die Daten werden nach Möglichkeit importiert bzw. geladen. Wenn die Daten nach der Umsetzung zu lang sind, wird die entsprechende Zeile zurückgewiesen. Ansonsten werden die Daten importiert bzw. geladen.

Versetzen von XML-Daten

Unterstützung für das Versetzen von XML-Daten wird durch die Dienstprograme LOAD, IMPORT und EXPORT bereitgestellt. Unterstützung für das Versetzen von

Tabellen, die XML-Spalten enthalten, ohne in den Offlinestatus zu wechseln, wird durch die gespeicherte Prozedur ADMIN_MOVE_TABLE bereitgestellt.

Importieren von XML-Daten

Das Dienstprogramm IMPORT kann zum Einfügen von XML-Dokumenten in eine reguläre, relationale Tabelle verwendet werden. Nur korrekt formatierte XML-Dokumente können importiert werden.

Verwenden Sie die Option XML FROM des Befehls IMPORT zum Angeben der Speicherposition der XML-Dokumente, die importiert werden sollen. Die Option XMLVALIDATE gibt an, wie die importierten Dokumente auf ihre Gültigkeit überprüft werden sollen. Sie können auswählen, ob die importierten XML-Daten auf der Basis des im Befehl IMPORT angegebenen Schemas, auf der Basis eines Schemapositionshinweises im XML-Quellendokument oder auf der Basis eines Schemas überprüft werden sollen, das von der XML-Datenkennung in der Hauptdatendatei angegeben wird. Sie können auch die Option XMLPARSE verwenden, um anzugeben, wie Leerzeichen beim Importieren des XML-Dokuments verarbeitet werden sollen. Die Dateitypänderungswerte `xmlchar` und `xmlgraphic` ermöglichen Ihnen die Angabe der Codierungsmerkmale für die importierten XML-Daten.

Laden von XML-Daten

Das Dienstprogramm LOAD bietet eine effiziente Methode zum Einfügen großer XML-Datenvolumina in eine Tabelle. Darüber hinaus können mit diesem Dienstprogramm auch bestimmte Optionen verwendet werden, die beim Dienstprogramm IMPORT nicht verfügbar sind, wie beispielsweise die Möglichkeit zum Laden von Daten aus einem benutzerdefinierten Cursor.

Ebenso wie beim Befehl IMPORT können Sie mit dem Befehl LOAD die Speicherposition der zu ladenden XML-Daten, die Prüfungsoptionen für die XML-Daten sowie die Art der Leerzeichenverarbeitung angeben. Und ebenso wie bei IMPORT können Sie die Dateitypänderungswerte `xmlchar` und `xmlgraphic` verwenden, um die Codierungsmerkmale für die geladenen XML-Daten anzugeben.

Exportieren von XML-Daten

Daten können aus Tabellen exportiert werden, die eine oder auch mehrere Spalten mit dem XML-Datentyp umfassen. Exportierte XML-Daten werden separat von der Hauptdatendatei gespeichert, die die exportierten relationalen Daten enthält. Informationen zu den einzelnen exportierten XML-Dokumenten werden in der Hauptdatei der exportierten Daten durch eine XML-Datenkennung (XDS) dargestellt. Die XML-Datenkennung besteht aus einer Zeichenfolge, in der der Name der Systemdatei angegeben ist, in der das XML-Dokument gespeichert ist. Darüber hinaus enthält diese Zeichenfolge auch die exakte Position und Länge des XML-Dokuments innerhalb dieser Datei sowie das XML-Schema, das zur Gültigkeitsprüfung des XML-Dokuments verwendet wird.

Sie können die Parameter XMLFILE, XML TO und XMLSAVESCHEMA des Befehls EXPORT verwenden, um detaillierte Informationen zur Speicherung exportierter XML-Dokumente anzugeben. Die Dateitypänderungswerte `xmlinsepfiles`, `xmlnodeclaration`, `xmlchar` und `xmlgraphic` ermöglichen Ihnen die Angabe weiterführender Details zur Speicherposition und zur Codierung der exportierten XML-Daten.

Versetzen von Tabellen im Onlinestatus

Mithilfe der gespeicherten Prozedur ADMIN_MOVE_TABLE werden die Daten in einer aktiven Tabelle in ein neues Tabellenobjekt mit demselben Namen versetzt, wobei die Daten online und im Zugriff bleiben. Die Tabelle kann eine oder auch mehrere Spalten mit dem XML-Datentyp enthalten. Versetzen Sie Tabellen online statt offline, wenn die Verfügbarkeit für Sie eine höhere Priorität hat als Kosten, Speicherplatz, Versetzungsleistung und Transaktionsaufwand.

Sie können die Prozedur einmal oder mehrmals aufrufen, wobei die Prozedur für jede Operation einen eigenen Aufruf ausführt. Wenn Sie mehrere Aufrufe verwenden, stehen Ihnen weitere Optionen zur Verfügung. Sie können beispielsweise das Versetzen abbrechen oder bestimmen, wann die Zieltabelle zur Aktualisierung in den Offlinestatus wechseln soll.

Versetzen von XML-Daten - zentrale Aspekte

Beim Importieren oder Exportieren von XML-Daten ist eine Reihe von Einschränkungen, Voraussetzungen und Erinnerungen zu berücksichtigen. Prüfen Sie diese Aspekte, bevor Sie XML-Daten importieren oder exportieren.

Berücksichtigen Sie die folgenden Aspekte, wenn Sie XML-Daten exportieren oder importieren:

- Exportierte XML-Daten werden immer separat von der Hauptdatendatei gespeichert, die die exportierten relationalen Daten enthält.
- Standardmäßig schreibt das Dienstprogramm EXPORT XML-Daten im Unicode-Format. Verwenden Sie den Dateitypänderungswert `xmlchar`, um XML-Daten in der Zeichencodepage zu schreiben, oder den Dateitypänderungswert `xmlgraphic`, um XML-Daten in UTF-16 (der grafischen Codepage) zu schreiben, unabhängig von der Anwendungscodepage.
- XML-Daten können in Nicht-Unicode-Datenbanken gespeichert werden und die in eine XML-Spalte einzufügenden Daten werden vor dem Einfügen aus der Datenbankcodepage in UTF-8 konvertiert. Um während des XML-Parsings die Einführung von Substitutionszeichen zu vermeiden, sollten einzufügende Zeichendaten ausschließlich aus Codepunkten bestehen, die Teil der Datenbankcodepage sind. Bei Einstellung des Konfigurationsparameters `enable_xmlchar` auf `no` wird das Einfügen von Zeichendatentypen während des XML-Parsings blockiert und so das Einfügen auf Datentypen beschränkt, die keiner Codepagekonvertierung unterzogen werden, beispielsweise BIT DATA, BLOB und XML.
- Beim Importieren oder Laden von XML-Daten wird davon ausgegangen, dass die XML-Daten im Unicode-Format codiert sind, sofern das zu importierende XML-Dokument keinen Deklarationstag enthält, der ein Codierungsattribut umfasst. Sie können den Dateitypänderungswert `xmlchar` verwenden, um anzugeben, dass die zu importierenden XML-Dokumente in der Zeichencodepage codiert werden, während der Dateitypänderungswert `xmlgraphic` angibt, dass die zu importierenden XML-Dokumente in UTF-16 codiert werden.
- Zeilen, die nicht korrekt formatierte Dokumente enthalten, werden von den Dienstprogrammen IMPORT und LOAD zurückgewiesen.
- Wenn für die Dienstprogramme IMPORT und LOAD die Option `XMLVALIDATE` angegeben wird, werden Dokumente, die erfolgreich anhand ihres übereinstimmenden Schemas überprüft werden konnten, während des Einfügens in eine Tabelle mit Annotationen versehen, die die Schemadaten enthalten. Zeilen, die Dokumente enthalten, deren Gültigkeitsprüfung auf der Basis des zugeordneten Schemas fehlgeschlagen ist, werden zurückgewiesen.

- Wenn für ein Dienstprogramm LOAD oder IMPORT die Option XMLVALIDATE angegeben wird und mehrere XML-Schemata zur Prüfung von XML-Dokumenten verwendet werden, müssen Sie unter Umständen den Konfigurationsparameter **catalogcache_sz** für die Katalogcachegröße erhöhen. Ist die Erhöhung von **catalogcache_sz** nicht möglich, können Sie den integrierten Import- oder Ladebefehl in mehrere Befehle aufteilen, die weniger Schemadokumente verwenden.
- Wenn Sie beim Exportieren von XML-Daten eine XQuery-Anweisung angeben, können Sie XDM-Instanzen (XDM = XQuery and XPath Data Model; XQuery- und XPath-Datenmodell) exportieren, bei denen es sich nicht um korrekt formatierte XML-Dokumente handelt. Exportierte XML-Dokumente, die nicht korrekt formatiert sind, können nicht direkt in eine XML-Spalte importiert werden, weil Spalten, für die der XML-Datentyp definiert wurde, nur vollständige, korrekt formatierte XML-Dokumente enthalten dürfen.
- Die Einstellung **CPU_PARALLELISM** wird während einer Ladeoperation auf den Wert 1 reduziert, wenn Statistikdaten erfasst werden.
- Für eine XML-Ladeoperation muss gemeinsam genutzter Sortierspeicher verwendet werden, um weiterarbeiten zu können. Aktivieren Sie den Parameter **SHEAPTHRES_SHR** oder **INTRA_PARALLEL** oder den Verbindungskonzentrator. Der Parameter **SHEAPTHRES_SHR** wird standardmäßig eingestellt, damit gemeinsam genutzter Sortierspeicher für die Standardkonfiguration verfügbar ist.
- Sie können die Option **SOURCEUSEREXIT** oder den Parameter **SAVECOUNT** des Befehls LOAD nicht angeben, wenn Sie eine Tabelle laden, die eine XML-Spalte enthält.
- Ebenso wie LOB-Dateien müssen auch XML-Dateien bei der Verwendung des Befehls LOAD auf der Serverseite gespeichert werden.
- Wenn Sie XML-Daten in einer Umgebung mit partitionierten Datenbanken auf mehrere Datenbankpartitionen laden, müssen alle Datenbankpartitionen Zugriff auf die Dateien haben, die die XML-Daten enthalten. Sie können den Zugriff auf die Dateien beispielsweise ermöglichen, indem Sie die Dateien kopieren oder einen NFS-Mount erstellen.

Verhalten von LOB- und XML-Dateien hinsichtlich IMPORT und EXPORT

LOB- und XML-Dateien weisen im Hinblick auf ihr Verhalten und die Kompatibilität bestimmte Gemeinsamkeiten auf, die beim Importieren (IMPORT) und Exportieren (EXPORT) von Daten genutzt werden können.

Export Wenn beim Exportieren von Daten mit der Option LOBS TO mindestens ein LOB-Pfad angegeben wird, dann schreibt das Dienstprogramm EXPORT reihum die aufeinander folgenden LOB-Werte in die entsprechenden LOB-Dateien. Wird mit der Option XML TO mindestens ein XML-Pfad angegeben, dann schreibt das Dienstprogramm EXPORT ebenfalls reihum alle aufeinander folgenden XDM-Instanzen (XDM = XQuery- und XPath-Datenmodell) in die entsprechenden XML-Dateien. Standardmäßig werden LOB-Werte und XDM-Instanzen in den Pfad geschrieben, in dem auch die exportierten relationalen Daten abgelegt werden. Sofern nicht der Dateitypmodifikator OBSINSEPFILES oder XMLINSEPFILES definiert wurde, sind sowohl für LOB- als auch für XML-Dateien mehrere verknüpfte Werte für dieselbe Datei zulässig.

Die Option LOBFILE bietet eine Möglichkeit zur Angabe des Basisdateinamens der LOB-Dateien, die vom Dienstprogramm EXPORT erstellt wurden. Ihn ähnlicher Weise bietet die Option XMLFILE die Möglichkeit, den Basisdateinamen der XML-Dateien anzugeben, die vom Dienstprogramm EXPORT generiert wurden. Der Standardbasisdateiname der LOB-Datei stimmt mit dem Namen der exportierten Datendatei überein, verfügt je-

doch über die Erweiterung .lob. Der Standardbasisdateiname der XML-Datei stimmt mit dem Namen der exportierten Datendatei überein, verfügt jedoch über die Erweiterung .xml. Der vollständige Name der exportierten LOB- oder XML-Datei besteht deshalb aus dem Basisdateinamen und einer numerischen Erweiterung, die auf drei Ziffern aufgefüllt wird, sowie der Erweiterung .lob oder .xml.

Import

Beim Importieren von Daten ist ein LLS (LOB Location Specifier) mit einer XML-Zielspalte und eine XML-Datenkennung (XDS) mit einer LOB-Zielspalte kompatibel. Wird die Option LOBS FROM nicht angegeben, dann geht das System davon aus, dass die zu importierenden LOB-Dateien sich im selben Pfad wie die relationale Eingabedatendatei befinden. Wenn die Option XML FROM nicht angegeben wurde, geht das System davon aus, dass die zu importierenden XML-Dateien sich im selben Pfad wie die relationalen Eingabedatendatei befinden.

Beispiele zum Exportieren

In dem folgenden Beispiel werden alle LOB-Werte in die Datei /mypath/t1export.del.001.lob geschrieben, und alle XDM-Instanzen werden in die Datei /mypath/t1export.del.001.xml geschrieben:

```
EXPORT TO /mypath/t1export.del OF DEL MODIFIED BY LOBSINFILE
SELECT * FROM USER.T1
```

In dem folgenden Beispiel wird der erste LOB-Wert in die Datei /lob1/t1export.del.001.lob und der zweite in die Datei /lob2/t1export.del.002.lob geschrieben, während der dritte an die Datei /lob1/t1export.del.001.lob und der vierte an die Datei /lob2/t1export.del.002.lob angehängt wird usw.:

```
EXPORT TO /mypath/t1export.del OF DEL LOBS TO /lob1,/lob2
MODIFIED BY LOBSINFILE SELECT * FROM USER.T1
```

In dem folgenden Beispiel wird die erste XDM-Instanz in die Datei /xml1/xmlbase.001.xml, die zweite in die Datei /xml2/xmlbase.002.xml, die dritte in die Datei /xml1/xmlbase.003.xml und die vierte in die Datei /xml2/xmlbase.004.xml geschrieben usw.:

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /xml1,/xml2 XMLFILE xmlbase
      MODIFIED BY XMLINSEPFILES SELECT * FROM USER.T1
```

Beispiele zum Importieren

Für eine Tabelle mit dem Namen "mytable", die eine einzige XML-Spalte enthält, und den unten aufgeführten Befehl IMPORT gilt Folgendes:

```
IMPORT FROM myfile.del of del LOBS FROM /lobpath XML FROM /xmlpath
MODIFIED BY LOBSINFILE XMLCHAR replace into mytable
```

Wenn "myfile.del" die folgenden Daten enthält:

```
mylobfile.001.lob.123.456/
```

Das Dienstprogramm IMPORT versucht in diesem Fall, ein XML-Dokument aus der Datei /lobpath/mylobfile.001.lob zu importieren. Hierbei wird an der relativen Dateiposition (Offset) 123 begonnen und mit einer Länge von 456 Byte gearbeitet.

Es wird davon ausgegangen, dass sich die Datei "mylobfile.001.lob" in LOB-Pfad und nicht im XML-Pfad befindet, da auf den Wert mithilfe eines LLS (LOB Location Specifier) und nicht mit einer XML-Datenkennung verwiesen wird.

Außerdem geht das System davon aus, dass das Dokument in der Zeichencodepage codiert ist, da der Dateitypmodifikator XMLCHAR angegeben wurde.

XML-Datenkennung

XML-Daten, die mit den Dienstprogrammen EXPORT, IMPORT und LOAD versetzt werden, müssen in Dateien gespeichert werden, die separat von der Hauptdatendatei abgelegt sind. In der Hauptdatendatei werden die XML-Daten durch eine XML-Datenkennung (XDS = XML Data Specifier) dargestellt.

Die XML-Datenkennung besteht aus einer Zeichenfolge, die durch einen XML-Tag mit dem Namen "XDS" dargestellt wird, und verfügt über Attribute, die Informationen zu den eigentlichen XML-Daten in der Spalte enthalten. Hierzu gehören z. B. Angaben zum Namen der Datei, in der die eigentlichen XML-Daten enthalten sind, sowie Angaben zur relativen Position und Länge der XML-Daten in dieser Datei. Die Attribute der XDS werden in der folgenden Liste beschrieben.

FIL Der Name der Datei (File), die die XML-Daten enthält. Sie können keine benannte Pipe angeben. Das Importieren oder Laden von XML-Dokumenten aus einer benannten Pipe wird nicht unterstützt.

OFF Die relative Byteadresse (Offset) der XML-Daten in der Datei, die im Attribut FIL angegeben wurde. Hierbei beginnt die relative Adresse bei 0.

LEN Die Länge (Length) der XML-Daten, die sich in der im Attribut FIL angegebenen Datei befinden, in Byte.

SCH Die vollständig qualifizierte SQL-Kennung des XML-Schemas, das zur Gültigkeitsprüfung dieses XML-Dokuments verwendet wird. Die Schema- und Namenskomponenten der SQL-Kennung werden als Werte für "OBJECTSCHEMA" und "OBJECTNAME" der Zeile in der Katalogtabelle SYSCAT.XSROBJECTS gespeichert, die diesem XML-Schema zugeordnet ist.

Die XML-Datenkennung (XDS) wird als Zeichenfeld in der Datendatei interpretiert und unterliegt dem Parsingverhalten für Zeichenspalten des Dateiformats. Beim ASCII-Dateiformat mit Begrenzern (DEL) muss der Zeichenbegrenzer z. B. verdoppelt werden, sofern dieser in der XML-Datenkennung vorhanden ist. Die Sonderzeichen (<, >, &, ', ") in den Attributwerten müssen immer mit einem Escapezeichen versehen werden. Objektnamen, bei denen die Groß-/Kleinschreibung zu beachten ist, müssen zwischen Zeicheneinheiten vom Typ " eingeschlossen werden.

Beispiele

Gehen Sie von einem Attribut vom Typ FIL mit dem Wert `abc&"def".del` aus. Um diese XMS-Datenkennung in eine ASCII-Datei mit Begrenzern einzufügen, wobei als Zeichenbegrenzer das doppelte Anführungszeichen (") verwendet wird, müssen die doppelten Anführungszeichen (") verdoppelt und Sonderzeichen mit einem Escapezeichen versehen werden:

```
<XDS FIL=""abc&amp;&quot;def&quot;.del"" />
```

Das nachstehende Beispiel zeigt eine XML-Datenkennung (XDS), die in einer ASCII-Datendatei mit Begrenzern angegeben sein kann. XML-Daten werden in der Datei `xmldocs.xml.001` gespeichert, wobei bei der relativen Byteadresse 100 begonnen und eine Länge von 300 Byte verwendet wird. Da sich diese XML-Datenken-

nung in einer ASCII-Datei befindet, bei der als Begrenzer doppelte Anführungszeichen verwendet werden, müssen die doppelten Anführungszeichen innerhalb des XDS-Tags verdoppelt werden.

```
"<XDS FIL = ""xmldocs.xml.001"" OFF=""100"" LEN=""300"" />"
```

Das nachstehende Beispiel zeigt die vollständig qualifizierte SQL-Kennung `ANTHONY.purchaseOrderTest`. Der Teil der Kennung, bei dem die Groß-/Kleinschreibung zu beachten ist, muss in der XML-Datenkennung zwischen Zeicheneinheiten vom Typ " eingeschlossen werden:

```
"<XDS FIL='/home/db2inst1/xmlload/a.xml' OFF='0' LEN='6758'
   SCH='ANTHONY.&quot;purchaseOrderTest&quot;' />"
```

Abfrage- und XPath-Datenmodell

Auf XML-Daten in Tabellen können Sie entweder mithilfe der XQuery-Funktionen, die in SQL zur Verfügung stehen, oder durch direktes Aufrufen von XQuery zugreifen. Bei einer Instanz des XQuery- und XPath-Datenmodells (XDM) kann es sich um korrekt formatierte XML-Dokumente, um Folgen von Knoten oder atomaren Werten oder um beliebige Kombination aus Knoten und atomaren Werten handeln.

Einzelne XDM-Instanzen können mithilfe des Befehls EXPORT in eine XML-Datei oder auch in mehrere XML-Dateien geschrieben werden.

Anhang A. Unterschiede zwischen den Dienstprogrammen IMPORT und LOAD

Die folgende Tabelle fasst die wichtigen Unterschiede zwischen den DB2-Dienstprogrammen LOAD und IMPORT zusammen.

Dienstprogramm IMPORT	Dienstprogramm LOAD
Langsam beim Versetzen von großen Datenmengen.	Schneller als das Dienstprogramm IMPORT beim Versetzen großer Datenmengen, da das Dienstprogramm LOAD formatierte Seiten direkt in die Datenbank schreibt.
Partitionsinterne Parallelität wird eingeschränkt genutzt. Partitionsinterne Parallelität kann nur durch gleichzeitige Aufrufe des Importdienstprogramms im Modus ALLOW WRITE ACCESS erreicht werden.	Partitionsinterne Parallelität wird genutzt. Dazu sind in der Regel symmetrische Mehrprozessormaschinen (SMP-Maschinen) erforderlich.
FASTPARSE wird nicht unterstützt.	FASTPARSE wird unterstützt und bietet eine reduzierte Datenprüfung der vom Benutzer bereitgestellten Daten.
Hierarchische Daten werden unterstützt.	Hierarchische Daten werden nicht unterstützt.
Das Erstellen von Tabellen, Hierarchien und Indizes wird beim PC/IXF-Format unterstützt.	Tabellen und Indizes müssen vorhanden sein.
Das Importieren von Daten in MQTs (Materialized Query Tables, gespeicherte Abfragetabellen) wird nicht unterstützt.	Das Laden von Daten in MQTs wird unterstützt.
BINARYNUMERICS wird nicht unterstützt.	BINARYNUMERICS wird unterstützt.
PACKEDDECIMAL wird nicht unterstützt.	PACKEDDECIMAL wird unterstützt.
ZONEDDECIMAL wird nicht unterstützt.	ZONEDDECIMAL wird unterstützt.
Das Überschreiben von Spalten, die als GENERATED ALWAYS definiert sind, ist nicht möglich.	Das Überschreiben von Spalten, die als GENERATED ALWAYS definiert sind, ist durch Verwendung der Änderungswerte für den Dateityp `generatedoverride` und `identityoverride` möglich.
Das Importieren von Daten in Tabellen, Sichten und Kurznamen wird unterstützt.	Daten können nur in Tabellen geladen werden.
Alle Zeilen werden protokolliert.	Es erfolgt nur eine minimale Protokollierung.
Trigger werden unterstützt.	Trigger werden nicht unterstützt.
Wenn die Importoperation unterbrochen wird und ein Wert für *commitcount* angegeben wurde, ist die Tabelle verwendbar und enthält die Zeilen, die bis zum letzten Commit geladen wurden. Der Benutzer kann dann die Importoperation erneut starten oder die Tabelle so akzeptieren, wie sie vorliegt.	Wenn die Ladeoperation unterbrochen wird und ein Wert für *savecount* angegeben wurde, verbleibt die Tabelle im Status "Laden anstehend" und kann nicht verwendet werden, bis die Ladeoperation erneut gestartet und fortgesetzt wird oder der Tabellenbereich anhand eines vor der Ladeoperation erstellten Backup-Images wiederhergestellt wird.

Dienstprogramm IMPORT	Dienstprogramm LOAD
Der erforderliche Speicherbereich entspricht in etwa dem größten Index plus 10%. Dieser Speicherbereich wird den Tabellenbereichen für temporäre Tabellen in der Datenbank entnommen.	Der erforderliche Speicherbereich entspricht in etwa der Summe der Größen aller Indizes, die für die Tabelle definiert sind, und kann u. U. bis doppelt so groß werden. Dieser Speicherbereich wird dem temporären Speicherplatz in der Datenbank entnommen.
Alle Integritätsbedingungen werden während der Importoperation ausgewertet.	Das Dienstprogramm LOAD überprüft die Eindeutigkeit und berechnet Werte für generierte Spalten. Alle anderen Integritätsbedingungen müssen jedoch mit der Anweisung SET INTEGRITY überprüft werden.
Die Schlüsselwerte werden während einer Importoperation einzeln in den Index eingefügt.	Nach dem Laden der Daten werden die Schlüsselwerte sortiert, und der Index wird erstellt.
Wenn aktualisierte Statistikdaten erforderlich sind, muss das Dienstprogramm RUNSTATS nach einer Importoperation ausgeführt werden.	Statistikdaten können während der Ladeoperation zusammengestellt werden, wenn die Daten in der Tabelle ersetzt (REPLACE) werden.
Sie können Daten über DB2 Connect in eine Hostdatenbank importieren.	Es ist nicht möglich, Daten in eine Host-Datenbank zu laden.
Importdateien müssen auf dem Client vorhanden sein, auf dem das Dienstprogramm IMPORT aufgerufen wird.	In Abhängigkeit von den angegebenen Optionen können sich die zu ladenden Dateien oder Pipes entweder auf der/den Datenbankpartition(en) befinden, die die Datenbank enthält/enthalten, oder auf dem Client mit Fernverbindung, von dem aus das Dienstprogramm LOAD aufgerufen wird. **Anmerkung:** LOBs und XML-Daten können nur serverseitig gelesen werden.
Ein Backup-Image ist nicht erforderlich. Da das Dienstprogramm IMPORT SQL-INSERT-Anweisungen verwendet, wird die Aktivität protokolliert, und es sind keine Backups erforderlich, um diese Operationen bei Fehlern wiederherzustellen.	Ein Backup-Image kann während der Ladeoperation erstellt werden.

Anhang B. Von den Dienstprogrammen EXPORT, IMPORT und LOAD verwendete Bindedateien

Die folgende Tabelle listet Bindedateien mit ihren jeweiligen Standardisolationsstufen auf. Außerdem ist angegeben, welche Dienstprogramme diese Dateien einsetzen und zu welchem Zweck sie verwendet werden.

Bindedatei (Standardisolationsstufe)	Dienstprogramm/Zweck
db2ueiwi.bnd (CS)	IMPORT/EXPORT. Abfragen von Informationen zu Tabellenspalten und Indizes.
db2uexpm.bnd (CS)	EXPORT. Abrufen von Daten aus der für die Exportoperation angegebenen Abfrage.
db2uimpm.bnd (RS)	IMPORT. Einfügen von Daten aus der Quellendatendatei in die Zieltabelle, wenn die Option INSERT, REPLACE oder REPLACE_CREATE verwendet wird. **Anmerkung:** Die Optionen CREATE und REPLACE_CREATE des Befehls IMPORT werden nicht weiter unterstützt und in zukünftigen Releases möglicherweise entfernt.
db2uipkg.bnd (CS)	IMPORT. Prüfen von Bindeoptionen.
db2ucktb.bnd (CS)	LOAD. Ausführen allgemeiner Initialisierungsprozesse für eine Ladeoperation.
db2ulxld.bnd (CS)	LOAD. Verarbeiten der Abfrage, die während einer LOAD FROM CURSOR-Operation bereitgestellt wird.
db2uigsi.bnd (RS auf UNIX-basierten Systemen, RR auf allen anderen Plattformen)	IMPORT/EXPORT. Löschen von Indizes und Prüfen von referenziellen Integritätsbedingungen für eine IMPORT REPLACE-Operation. Außerdem Abrufen von Informationen zu Identitätsspalten beim Export von IXF-Dateien.
db2uqtpd.bnd (RR)	IMPORT/EXPORT. Ausführen der Verarbeitung für hierarchische Tabellen.
db2uimtb.bnd (RS)	IMPORT. Ausführen allgemeiner Initialisierungsprozesse für eine Importoperation.
db2uImpInsUpdate.bnd (RS)	IMPORT. Einfügen von Daten aus der Quellendatendatei in die Zieltabelle, wenn die Option INSERT_UPDATE verwendet wird. Kann nicht mit der Option INSERT BUF gebunden werden.
db2uiDescribe.bnd (RS)	IMPORT/EXPORT. Abfragen von Informationen zu der Definition und den Eigenschaften einer Tabelle, die verarbeitet wird. Beispielsweise ein Wrappermodul zum Ausführen einer SQL-Operation DESCRIBE für eine Tabelle, die verarbeitet wird.

Anhang C. Lesen von Syntaxdiagrammen

In diesem Abschnitt wird die Struktur von SQL-Syntaxdiagrammen beschrieben.

Lesen Sie die Syntaxdiagramme von links nach rechts sowie von oben nach unten, indem Sie dem Pfad des Zeilenverlaufs folgen.

Das Symbol ►►— kennzeichnet den Beginn eines Syntaxdiagramms.

Das Symbol —► gibt an, dass die Syntax in der nächsten Zeile fortgesetzt wird.

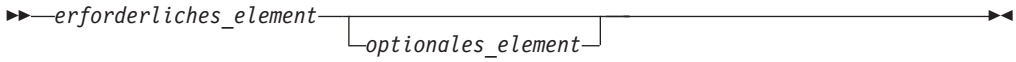
Das Symbol ►— gibt an, dass die Syntax von der vorherigen Zeile hier fortgesetzt wird.

Das Symbol —►◄ kennzeichnet das Ende eines Syntaxdiagramms.

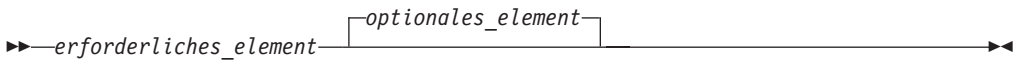
Syntaxfragmente beginnen mit dem Symbol ├— und Enden mit dem Symbol —┤.

Erforderliche Elemente werden auf der horizontalen Linie (dem Hauptpfad) angegeben.

Optionale Elemente werden unterhalb des Hauptpfads angegeben.

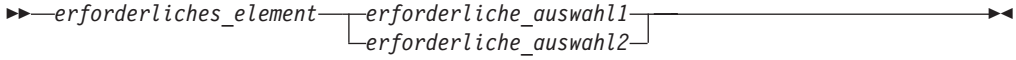


Wird ein optionales Element oberhalb des Hauptpfads angezeigt, hat dieses Element keinen Einfluss auf die Ausführung, sondern wird nur der einfacheren Lesbarkeit halber angegeben.

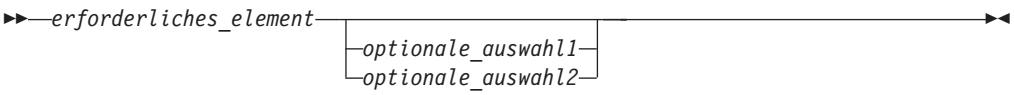


Ist eine Auswahl aus zwei oder mehr Elementen möglich, werden die Optionen in einer Stapelgruppe angezeigt.

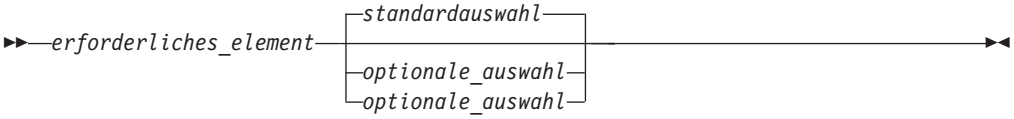
Wenn eines der Elemente ausgewählt werden *muss*, wird ein Element des Stapels im Hauptpfad angegeben.



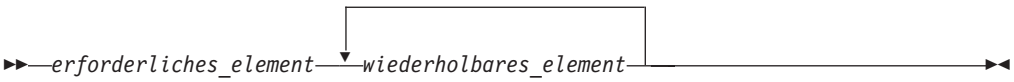
Ist die Auswahl eines der Elemente optional, wird der gesamte Stapel unterhalb des Hauptpfads angegeben.



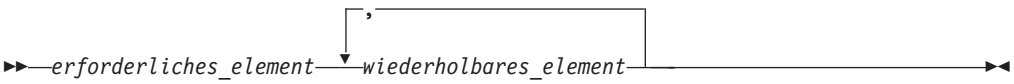
Ist eines der Elemente die Standardauswahl, wird das betreffende Element oberhalb des Hauptpfads angezeigt, und die verbleibenden Auswahlmöglichkeiten werden unterhalb des Hauptpfads angegeben.



Ein nach links zurückweisender Pfeil oberhalb des Hauptpfads gibt ein Element an, das wiederholt werden kann. In diesem Fall müssen wiederholte Elemente durch mindestens ein Leerzeichen voneinander getrennt werden.



Enthält der Wiederholungspfeil ein Komma, müssen die wiederholten Elemente durch ein Komma voneinander getrennt werden.

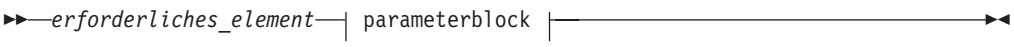


Ein Wiederholungspfeil oberhalb einer Stapelgruppe gibt an, dass aus den gestapelten Elementen mehr als eine Option ausgewählt werden kann oder dass eine ausgewählte Einzeloption wiederholt werden kann.

Schlüsselwörter werden in Großbuchstaben angegeben (beispielsweise `FROM`). Ihre Schreibweise muss genau der angegebenen Schreibweise entsprechen. Variablen werden in Kleinbuchstaben angegeben (beispielsweise `spaltenname`). Sie stehen für Namen oder Werte in der Syntax, die vom Benutzer angegeben werden.

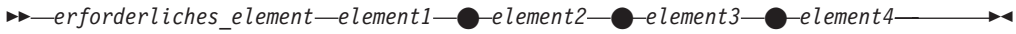
Werden Interpunktionszeichen, runde Klammern, arithmetische Operatoren oder ähnliche Symbole angezeigt, müssen diese als Teil der Syntax eingegeben werden.

In einigen Fällen steht eine einzelne Variable für ein größeres Fragment der Syntax. Im folgenden Diagramm beispielsweise steht die Variable `parameterblock` für das gesamte Syntaxfragment, das mit **parameterblock** gekennzeichnet ist:



parameterblock:

Benachbarte Segmente zwischen „großen Listenpunkten" (●) können in beliebiger Reihenfolge angegeben werden.



Das vorstehende Diagramm zeigt, dass 'element2' und 'element3' in beiden möglichen Reihenfolgen angegeben werden können. Die folgenden beiden Reihenfolgen sind gültig:

```
erforderliches_element element1 element2 element3 element4
erforderliches_element element1 element3 element2 element4
```

Anhang D. Erfassen von Daten für Probleme beim Versetzen von Daten

Wenn bei der Ausführung von Befehlen für das Versetzen von Daten Fehler auftreten und die Ursache des Problems nicht festgestellt werden kann, können Sie Diagnosedaten erfassen, die Sie selbst oder die Mitarbeiter der IBM Unterstützungsfunktion dazu verwenden können, das Problem zu diagnostizieren und zu beheben.

Führen Sie für Ihre Situation geeignete Anweisungen der folgenden Liste zum Erfassen von Daten aus:

- Zum Erfassen von Daten für Probleme beim Befehl **db2move** wechseln Sie in das Verzeichnis, in dem Sie den Befehl abgesetzt haben. Suchen Sie die folgende(n) Datei(en), abhängig von der im Befehl angegebenen Aktion:
  - Suchen Sie für die Aktion COPY nach Dateien mit dem Namen COPY.*zeitmarke*.ERR und COPYSCHEMA.*zeitmarke*.MSG. Wenn Sie darüber hinaus den Modus LOAD_ONLY oder DDL_AND_LOAD angegeben haben, suchen Sie außerdem nach einer Datei mit dem Namen LOADTABLE.*zeitmarke*.MSG.
  - Suchen Sie für die Aktion EXPORT nach der Datei EXPORT.out.
  - Suchen Sie für die Aktion IMPORT nach der Datei IMPORT.out.
  - Suchen Sie für die Aktion LOAD nach der Datei LOAD.out.
- Zum Erfassen von Daten für Probleme bei den Befehlen **EXPORT**, **IMPORT** oder **LOAD** stellen Sie fest, ob der Befehl den Parameter MESSAGES enthielt. War dies der Fall, erfassen Sie die Daten der entsprechenden Ausgabedatei. Diese Dienstprogramme verwenden das aktuelle Verzeichnis und das Standardlaufwerk als Ziel, falls keine anderen Angaben gemacht werden.
- Suchen Sie zum Erfassen von Daten für Probleme beim Befehl **REDISTRIBUTE** nach der Datei "*datenbankname.name_der_datenbankpartitionsgruppe.zeitmarke*" unter Linux- und UNIX-Betriebssystemen bzw. nach "*datenbankname.name_der_datenbankpartitionsgruppe.datum.uhrzeit*" unter Windows-Betriebssystemen. Sie befindet sich im Verzeichnis *$HOME*/sqllib/db2dump unter Linux- und UNIX-Betriebssystemen bzw. im Verzeichnis $DB2PATH\sqllib\redist unter Windows-Betriebssystemen, wobei *$HOME* das Ausgangsverzeichnis des Instanzeigners ist.

Anhang E. 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 33. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Administrative API Reference	SC27-3864-00	Ja	April 2012
Administrative Routines and Views	SC27-3865-01	Nein	Januar 2013
Call Level Interface Guide and Reference Volume 1	SC27-3866-01	Ja	Januar 2013
Call Level Interface Guide and Reference Volume 2	SC27-3867-01	Ja	Januar 2013
Command Reference	SC27-3868-01	Ja	Januar 2013
Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen	SC12-4673-01	Ja	Januar 2013
Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz	SC12-4691-01	Ja	Januar 2013
Datenbanküberwachung - Handbuch und Referenz	SC12-4674-01	Ja	Januar 2013
Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz	SC12-4692-01	Ja	Januar 2013
Datenbanksicherheit	SC12-4693-01	Ja	Januar 2013
DB2 Workload Management - Handbuch und Referenz	SC12-4683-01	Ja	Januar 2013
Developing ADO.NET and OLE DB Applications	SC27-3873-01	Ja	Januar 2013

Tabelle 33. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Developing Embedded SQL Applications	SC27-3874-01	Ja	Januar 2013
Developing Java Applications	SC27-3875-01	Ja	Januar 2013
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-3876-00	Nein	April 2012
Developing RDF Applications for IBM Data Servers	SC27-4462-00	Ja	Januar 2013
Developing User-defined Routines (SQL and External)	SC27-3877-01	Ja	Januar 2013
Getting Started with Database Application Development	GI13-2046-01	Ja	Januar 2013
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GI11-3285-00	Ja	April 2012
Globalisierung	SC12-4694-00	Ja	April 2012
DB2-Server - Installation	SC12-4677-01	Ja	Januar 2013
IBM Data Server-Clients - Installation	SC12-4678-00	Nein	April 2012
Fehlernachrichten, Band 1	SC12-4686-01	Nein	Januar 2013
Fehlernachrichten, Band 2	SC12-4687-01	Nein	Januar 2013
Net Search Extender - Verwaltung und Benutzerhandbuch	SC12-4689-01	Nein	Januar 2013
Partitionierung und Clustering	SC12-4695-01	Ja	Januar 2013
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	Nein	Januar 2013
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	Nein	Januar 2013
pureXML - Handbuch	SC12-4684-01	Ja	Januar 2013
Spatial Extender - Benutzer- und Referenzhandbuch	SC12-4688-00	Nein	April 2012
SQL Procedural Languages: Application Enablement and Support	SC27-3896-01	Ja	Januar 2013
SQL Reference Volume 1	SC27-3885-01	Ja	Januar 2013

Tabelle 33. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
SQL Reference Volume 2	SC27-3886-01	Ja	Januar 2013
Text Search	SC12-4690-01	Ja	Januar 2013
Fehlerbehebung und Optimieren der Datenbankleistung	SC12-4675-01	Ja	Januar 2013
Upgrade auf DB2 Version 10.1	SC12-4676-01	Ja	Januar 2013
Neuerungen in DB2 Version 10.1	SC12-4682-01	Ja	Januar 2013
XQuery - Referenz	SC12-4685-01	Nein	Januar 2013

Tabelle 34. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
DB2 Connect - Installation und Konfiguration von DB2 Connect Personal Edition	SC12-4679-00	Ja	April 2012
DB2 Connect - Installation und Konfiguration von DB2 Connect-Servern	SC12-4680-01	Ja	Januar 2013
DB2 Connect - Benutzerhandbuch	SC12-4681-01	Ja	Januar 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://publib.boulder.ibm.com/infocenter/db2luw/v10r1.

Für Themen aus DB2 Version 9.8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Für Themen aus DB2 Version 9.7 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Für Themen aus DB2 Version 9.5 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Für Themen aus DB2 Version 9.1 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Für Themen aus DB2 Version 8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Ein lokal installiertes DB2 Information Center muss regelmäßig aktualisiert werden.

Vorbereitende Schritte

Ein DB2 Version 10.1 Information Center muss bereits installiert sein. Einzelheiten hierzu finden Sie unter „Installation des DB2 Information Center mit dem DB2-Installationsassistenten" in *DB2-Server - Installation*. Alle für die Installation des Information Center geltenden Voraussetzungen und Einschränkungen gelten auch für die Aktualisierung des Information Center.

Informationen zu diesem Vorgang

Ein vorhandenes DB2 Information Center kann automatisch oder manuell aktualisiert werden:

- Mit automatischen Aktualisierungen werden vorhandene Komponenten und Sprachen des Information Center aktualisiert. Ein Vorteil von automatischen Aktualisierungen ist, dass das Information Center im Vergleich zu einer manuellen Aktualisierung nur für einen kurzenZeitraum nicht verfügbar ist. Darüber hinaus können automatische Aktualisierungen so konfiguriert werden, dass sie als Teil anderer, regelmäßig ausgeführter Stapeljobs ausgeführt werden.
- Mit manuellen Aktualisierungen können Sie vorhandene Komponenten und Sprachen des Information Center aktualisieren. Automatische Aktualisierungen reduzieren die Ausfallzeiten während des Aktualisierungsprozesses, Sie müssen jedoch den manuellen Prozess verwenden, wenn Sie Komponenten oder Sprachen hinzufügen möchten. Beispiel: Ein lokales Information Center wurde ursprünglich sowohl mit englischer als auch mit französischer Sprachunterstützung installiert; nun soll auch die deutsche Sprachunterstützung installiert werden. Bei einer manuellen Aktualisierung werden sowohl eine Installation der deutschen Sprachunterstützung als auch eine Aktualisierung der vorhandenen Komponenten und Sprachen des Information Center durchgeführt. Sie müssen jedoch bei einer manuellen Aktualisierung das Information Center manuell stop-

pen, aktualisieren und erneut starten. Das Information Center ist während des gesamten Aktualisierungsprozesses nicht verfügbar. Während des automatischen Aktualisierungsprozesses kommt es zu einem Ausfall des Information Center, und es wird erst wieder nach der Aktualisierung erneut gestartet.

Dieser Abschnitt enthält Details zum Prozess der automatischen Aktualisierung. Anweisungen zur manuellen Aktualisierung finden Sie im Abschnitt „Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center".

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte DB2 Information Center automatisch zu aktualisieren:

1. Unter Linux:
   a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `/opt/ibm/db2ic/V10.1` installiert.
   b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc/bin`.
   c. Führen Sie das Script `update-ic` aus:
      ```
      update-ic
      ```
2. Unter Windows:
   a. Öffnen Sie ein Befehlsfenster.
   b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `<Programme>\IBM\DB2 Information Center\Version 10.1` installiert, wobei `<Programme>` das Verzeichnis der Programmdateien angibt.
   c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc\bin`.
   d. Führen Sie die Datei `update-ic.bat` aus:
      ```
      update-ic.bat
      ```

Ergebnisse

Das DB2 Information Center wird automatisch erneut gestartet. Standen Aktualisierungen zur Verfügung, zeigt das Information Center die neuen und aktualisierten Abschnitte an. Waren keine Aktualisierungen für das Information Center verfügbar, wird eine entsprechende Nachricht zum Protokoll hinzugefügt. Die Protokolldatei befindet sich im Verzeichnis `doc\eclipse\configuration`. Der Name der Protokolldatei ist eine Zufallszahl. Beispiel: `1239053440785.log`.

Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Wenn Sie das DB2 Information Center lokal installiert haben, können Sie Dokumentationsaktualisierungen von IBM abrufen und installieren.

Informationen zu diesem Vorgang

Zur manuellen Aktualisierung des lokal installierten *DB2 Information Center* sind die folgenden Schritte erforderlich:

1. Stoppen Sie das *DB2 Information Center* auf Ihrem Computer und starten Sie das Information Center im Standalone-Modus erneut. Die Ausführung des Information Center im Standalone-Modus verhindert, dass andere Benutzer in Ihrem

Netz auf das Information Center zugreifen, und ermöglicht das Anwenden von Aktualisierungen. Die Workstationversion des DB2 Information Center wird stets im Standalone-Modus ausgeführt.

2. Verwenden Sie die Aktualisierungsfunktion, um zu prüfen, welche Aktualisierungen verfügbar sind. Falls Aktualisierungen verfügbar sind, die Sie installieren müssen, können Sie die Aktualisierungsfunktion verwenden, um diese abzurufen und zu installieren.

   Anmerkung: Wenn es in der verwendeten Umgebung erforderlich ist, die Aktualisierungen für das *DB2 Information Center* auf einer Maschine zu installieren, die nicht über ein Verbindung zum Internet verfügt, spiegeln Sie die Aktualisierungssite auf ein lokales Dateisystem und verwenden Sie dabei eine Maschine, die mit dem Internet verbunden ist und auf der das *DB2 Information Center* installiert ist. Wenn viele Benutzer Ihres Netzes die Dokumentationsaktualisierungen installieren sollen, können Sie die Zeit, die jeder einzelne Benutzer für die Aktualisierungen benötigt, reduzieren, indem Sie die Aktualisierungssite lokal spiegeln und ein Proxy dafür erstellen.
   Ist dies der Fall, verwenden Sie die Aktualisierungsfunktion, um die Pakete abzurufen. Die Aktualisierungsfunktion ist jedoch nur im Standalone-Modus verfügbar.

3. Stoppen Sie das im Standalone-Modus gestartete Information Center und starten Sie das *DB2 Information Center* auf Ihrem Computer erneut.

Anmerkung: Unter Windows 2008 und Windows Vista (und neueren Versionen) müssen die in diesem Abschnitt aufgeführten Befehle mit Administratorberechtigung ausgeführt werden. Zum Öffnen einer Eingabeaufforderung oder eines Grafiktools mit vollen Administratorberechtigungen klicken Sie mit der rechten Maustaste die Verknüpfung an und wählen Sie **Als Administrator ausführen** aus.

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte *DB2 Information Center* zu aktualisieren:

1. Stoppen Sie das *DB2 Information Center*.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Beenden** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv10 stop
     ```
2. Starten Sie das Information Center im Standalone-Modus.
   - Unter Windows:
     a. Öffnen Sie ein Befehlsfenster.
     b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis *Programme*\IBM\DB2 Information Center\Version 10.1 installiert, wobei *Programme* das Verzeichnis der Programmdateien angibt.
     c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc\bin.
     d. Führen Sie die Datei help_start.bat aus:
        ```
        help_start.bat
        ```
   - Unter Linux:

a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis `/opt/ibm/db2ic/V10.1` installiert.
b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc/bin`.
c. Führen Sie das Script `help_start` aus:

```
help_start
```

Der standardmäßig auf dem System verwendete Web-Browser wird geöffnet und zeigt die Standalone-Version des Information Center an.

3. Klicken Sie die Aktualisierungsschaltfläche () an. (JavaScript muss im verwendeten Browser aktiviert sein.) Klicken Sie im rechten Fenster des Information Center die Schaltfläche für die Suche nach Aktualisierungen an. Eine Liste der Aktualisierungen für die vorhandene Dokumentation wird angezeigt.
4. Wählen Sie zum Initiieren des Installationsprozesses die gewünschten Aktualisierungen aus und klicken Sie anschließend die Schaltfläche für die Installation der Aktualisierungen an.
5. Klicken Sie nach Abschluss des Installationsprozesses **Fertigstellen** an.
6. Stoppen Sie das im Standalone-Modus gestartete Information Center:
   - Unter Windows: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc\bin`, und führen Sie die Datei `help_end.bat` aus:

     ```
     help_end.bat
     ```

     Anmerkung: Die Stapeldatei `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit der Stapeldatei `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie nicht die Tastenkombination `Strg+C` oder eine andere Methode, um `help_start.bat` zu stoppen.
   - Unter Linux: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc/bin`, und führen Sie das Script `help_end` aus:

     ```
     help_end
     ```

     Anmerkung: Das Script `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit dem Script `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie keine andere Methode, um das Script `help_start` zu stoppen.
7. Starten Sie das *DB2 Information Center* erneut.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Start** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:

     ```
     /etc/init.d/db2icdv10 start
     ```

Ergebnisse

Im aktualisierten *DB2 Information Center* werden die neuen und aktualisierten Themen angezeigt.

DB2-Lernprogramme

Die DB2-Lernprogramme unterstützen Sie dabei, sich mit den unterschiedlichen Aspekten der DB2-Produkte vertraut zu machen. Die Lerneinheiten bieten eine in einzelne Schritte unterteilte Anleitung.

Vorbereitungen

Die XHTML-Version des Lernprogramms kann über das Information Center unter http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/ angezeigt werden.

In einigen der Lerneinheiten werden Beispieldaten und Codebeispiele verwendet. Informationen zu bestimmten Voraussetzungen für die Ausführung der Tasks finden Sie in der Beschreibung des Lernprogramms.

DB2-Lernprogramme

Klicken Sie zum Anzeigen des Lernprogramms den Titel an.

„pureXML" in *pureXML - Handbuch*
Einrichten einer DB2-Datenbank, um XML-Daten zu speichern und Basisoperationen mit dem nativen XML-Datenspeicher auszuführen.

Informationen zur Fehlerbehebung in DB2

Es steht eine breite Palette verschiedener Informationen zur Fehlerbestimmung und Fehlerbehebung zur Verfügung, um Sie bei der Verwendung von DB2-Datenbankprodukten zu unterstützen.

DB2-Dokumentation
Informationen zur Fehlerbehebung stehen im Handbuch *Fehlerbehebung und Optimieren der Datenbankleistung* oder im Abschnitt mit grundlegenden Informationen zu Datenbanken im *DB2 Information Center* zur Verfügung, darunter:

- Informationen zum Eingrenzen und Aufdecken von Problemen mithilfe der Diagnosetools und -dienstprogramme von DB2.
- Lösungsvorschläge zu den am häufigsten auftretenden Problemen.
- Ratschläge zum Lösen anderer Probleme, die bei Verwendung der DB2-Datenbankprodukte auftreten können.

IBM Support Portal
Im IBM Support Portal finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die Website mit technischer Unterstützung enthält Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports) und Fehlerkorrekturen, Fixpacks sowie weiteren Ressourcen. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Sie können auf das IBM Support Portal über die folgende Website zugreifen: http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows.

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 F. 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

I

K

L

M

N

P

R

S

T

U

V

W

X

Z

SC12-4691-01

Spine information:

IBM DB2 10.1 for Linux, UNIX and Windows

Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz