IBM DB2 Universal Database

XML Extender Verwaltung und Programmierung

Version 7

IBM DB2 Universal Database

XML Extender Verwaltung und Programmierung

Version 7

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Bemerkungen” auf Seite 307 gelesen werden.

- Die IBM Homepage finden Sie im Internet unter: **ibm.com**
- IBM und das IBM Logo sind eingetragene Marken der International Business Machines Corporation.
- Das e-business Symbol ist eine Marke der International Business Machines Corporation
- Infoprint ist eine eingetragene Marke der IBM.
- ActionMedia, LANDesk, MMX, Pentium und ProShare sind Marken der Intel Corporation in den USA und/oder anderen Ländern.
- C-bus ist eine Marke der Corollary, Inc. in den USA und/oder anderen Ländern.
- Java und alle Java-basierenden Marken und Logos sind Marken der Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- Microsoft Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.
- PC Direct ist eine Marke der Ziff Communications Company in den USA und/oder anderen Ländern.
- SET und das SET-Logo sind Marken der SET Secure Electronic Transaction LLC.
- UNIX ist eine eingetragene Marke der Open Group in den USA und/oder anderen Ländern.
- Marken anderer Unternehmen/Hersteller werden anerkannt.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Universal Database, XML Extender Administration and Programming Guide,,
IBM Form xxxxxxxx

herausgegeben von International Business Machines Corporation, USA

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

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

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
SW NLS Center
Kst. 2877
November 2000

Inhaltsverzeichnis

Tabellen

Zu diesem Handbuch

Dieser Abschnitt beschreibt die folgenden Informationen:

Zielgruppe

Dieses Handbuch richtet sich an folgende Personen:

- Personen, die mit XML-Daten in DB2-Anwendungen arbeiten und die mit den Konzepten von XML vertraut sind. Diese Personen sollten ein allgemeines Verständnis zu den Bereichen XML und DB2 haben. Weitere Informationen zu XML und zugehörigen Themen finden Sie auf der folgenden Web-Site:

  http://www.w3c.org/XML

  Weitere Informationen zu DB2 finden Sie auf der folgenden Web-Site:

  http://www.ibm.com/software/data/db2/library
- DB2-Datenbankadministratoren, die mit den Konzepten, Tools und Techniken zur Verwaltung von DB2 vertraut sind.
- DB2-Anwendungsprogrammierer, die mit SQL sowie einer oder mehreren Programmiersprachen für DB2-Anwendungen vertraut sind.

Eine aktuelle Version dieses Handbuchs abrufen

Sie können die neueste Version dieses Handbuchs über die XML Extender-Web-Site abrufen:

http://www.ibm.com/software/data/db2/extenders/xmlext/library.html

Benutzung des Handbuchs

Dieses Handbuch ist wie folgt strukturiert:

Teil 1. Einführung
Dieser Teil des Handbuchs enthält einen Überblick zum XML Extender und darüber, wie Sie ihn in Ihren Geschäftsanwendungen einsetzen können. Er enthält ein Szenario ″Erste Schritte″, der Ihnen den Einstieg erleichtert.

Teil 2. Verwaltung
Dieser Teil beschreibt die Vorbereitung und Verwaltung einer DB2-Datenbank für XML-Daten. Lesen Sie diesen Teil, wenn Sie eine DB2-Datenbank verwalten müssen, die XML-Daten enthält.

Teil 3. Programmierung
Dieser Teil beschreibt die Verwaltung Ihrer XML-Daten. Lesen Sie diesen Teil, wenn Sie XML-Daten in einem DB2-Anwendungsprogramm aufrufen und bearbeiten müssen.

Teil 4. Referenz
Dieser Teil beschreibt die Verwendung der XML Extender-Verwaltungsbefehle, der benutzerdefinierten Typen und der gespeicherten Prozeduren. Außerdem listet er die Nachrichten und Codes auf, die vom XML Extender ausgegeben werden. Lesen Sie diesen Teil, wenn Sie mit den Konzepten und Tasks des XML Extender vertraut sind, aber Informationen zu einem benutzerdefinierten Typ (UDT), einer benutzerdefinierten Funktion (UDF), einem Befehl, einer Nachricht, Metadatentabellen, Steuertabellen oder Code benötigen.

Teil 5. Anhänge
Die Anhänge beschreiben die DTD für die Dokumentzugriffsdefinition, Muster zu den Beispielen, ein Szenario ″Erste Schritte″ sowie weitere IBM XML-Produkte.

Was ist neu in diesem Buch für DB2 UDB Version Fixpak 2

Dieses Dokument enthält neue oder umgeschriebene Informationen zu folgenden Themen:

- Verwaltungsassistent einstellen und starten
- Wie die Aktualisierungs-UDF XML-Dokumente aktualisiert
- Wie die XMLClob-UDF Ergebnisse zurückgibt
- Vergrößern der CLOB-Begrenzungen für gespeicherte Prozeduren
- Änderungen der DAD-Voraussetzungen:
  - Einbeziehen der Verarbeitungsanweisungen für Formatvorlagen beim Erstellen neuer XML-Dokumente
  - Verwenden einer fehlenden oder leeren Bedingung für den ersten RDB-Knoten, wenn nur eine Tabelle angegeben ist.
- Umsetzen von Parametermarken mit JDBC
- Arbeiten mit Codepages:
  - Gültige Codierungs-Deklarationen
  - Annahmen bei der Konvertierung
  - Szenarien bei der Konvertierung
- Anhang zu XML Extender-Parameterbegrenzungen

Geänderte Informationen sind mit einem vertikalen Änderungsbalken „|" markiert.

Informationen zu Korrekturen von Defekten mit diesem Fixpack finden Sie in der Readme-Datei.

Schlagen Sie Fragen und Antworten, bekannte Probleme sowie weitere Informationen zu Problemen und Lösungen in der Unterstützungsseite der XML Extender-Website nach:
http://www.ibm.com/software/data/db2/extenders/xmlext/support.html

Dieses Buch in das DB2 UDB Version 7 Information Center einbeziehen

Die Dokumentation zum XML Extender kann über die folgenden Schritte in das DB2 Information Center einbezogen werden:

- Kopieren Sie die HTML-Dateien für dieses Buch aus dem Unterverzeichnis `DOC` für ihre Sprache des XML-Produkts in das Unterverzeichnis im DB2 UDB-Dokumentationsverzeichnis für Ihre Sprache:

  Für Windows lautet das Verzeichnis für das Information Center `sqllib\doc\html\db2sx`

  Für UNIX lautet das Verzeichnis für das Information Center *`Installationsverzeichnis`*`/doc/html/db2sx`
- Starten Sie das Information Center erneut, und dieses Buch wird in die Indexzunge **Bücher** einbezogen.

Hervorhebungskonventionen

In diesem Handbuch werden folgende Konventionen verwendet:

Fettschrift	Text in Fettschrift kennzeichnet: • Befehle • Feldnamen • Menünamen • Druckknöpfe
Kursivschrift	Text in Kursivschrift kennzeichnet: • Variable Parameter, die durch einen Wert zu ersetzen sind • Hervorgehobene Begriffe • Erstes Auftreten eines Glossareintrags
GROSSBUCHSTABEN	Text in Großbuchstaben kennzeichnet: • Datentypen • Spaltennamen • Tabellennamen
`Beispiel`	Beispieltext kennzeichnet: • Systemnachrichten • Von Ihnen eingegebene Werte • Codierungsbeispiele • Verzeichnisnamen • Dateinamen • Pfadnamen

Syntaxdiagramme lesen

In diesem Handbuch wird die Syntax von Befehlen und SQL-Anweisungen anhand von Syntaxdiagrammen beschrieben.

Lesen Sie die Syntaxdiagramme wie folgt:

- Lesen Sie die Syntaxdiagramme von links nach rechts und von oben nach unten. Folgen Sie dabei dem durch die Linie angegebenen Pfad.

  Das Symbol ▸▸— kennzeichnet den Beginn einer Anweisung.

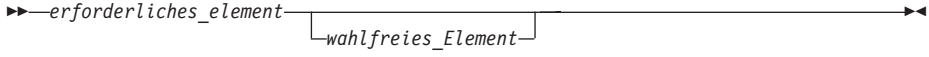
  Das Symbol —▸ gibt an, daß die Syntax der Anweisung auf der nächsten Zeile fortgesetzt wird.

  Das Symbol ▸— gibt an, daß eine Anweisung von der vorherigen Zeile fortgesetzt wird.

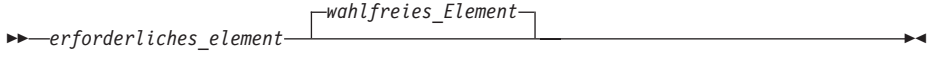
  Das Symbol —▸◂ kennzeichnet das Ende einer Anweisung.

  Diagramme zu anderen syntaktischen Einheiten als vollständigen Anweisungen beginnen mit dem Symbol ▸— und enden mit dem Symbol —▸.
- Erforderliche Elemente erscheinen auf der horizontalen Linien (dem Hauptpfad).

- Wahlfreie Elemente erscheinen unterhalb des Hauptpfads.

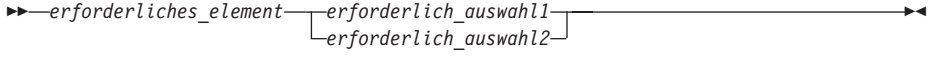


  Wenn ein wahlfreies Element oberhalb des Hauptpfads erscheint, hat dieses Element keine Auswirkung auf die Ausführung der Anweisung und dient lediglich der besseren Lesbarkeit.

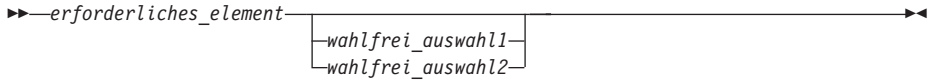


- Wenn Sie aus zwei oder mehr Elementen auswählen können, werden diese übereinander als Stapel angezeigt.

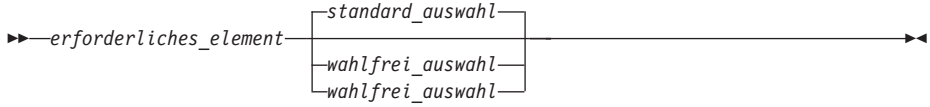
  Wenn Sie eines dieser Elemente auswählen *müssen*, erscheint eines der Elemente des Stapels im Hauptpfad.



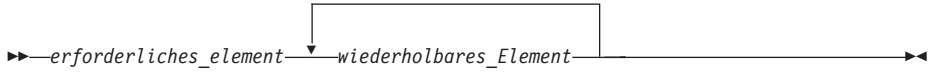
  Wenn die Auswahl eines dieser Elemente wahlfrei ist, erscheint der gesamte Stapel unterhalb des Hauptpfads.



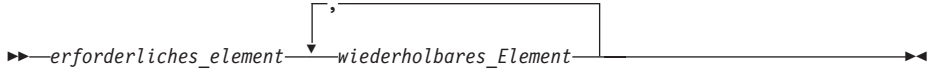
Wenn eines der Elemente den Standardwert darstellt, erscheint es oberhalb des Hauptpfads, und die weiteren Auswahloptionen werden darunter angezeigt.



- Ein Pfeil, der oberhalb der Hauptlinie zurückkehrt, weist auf ein Element hin, das wiederholt werden kann.



Wenn der Wiederholungspfeil Interpunktionszeichen enthält, müssen Sie die wiederholten Elemente mit den angegebenen Interpunktionszeichen trennen.



Ein Wiederholungspfeil oberhalb eines Stapels gibt an, daß Sie die Elemente in dem Stapel wiederholen können.

- Schlüsselwörter erscheinen in Großbuchstaben (z. B. `FROM`). Im XML Extender können Schlüsselwörter in Groß- oder Kleinbuchstaben eingegeben werden. Begriffe, bei denen es sich nicht um Schlüsselwörter handelt (z. B. *spaltenname*), werden in Kleinbuchstaben dargestellt. Sie stehen für benutzerdefinierte Namen oder Werte.
- Wenn Interpunktionszeichen, Klammern, arithmetische Operatoren oder andere Symbole dieser Art darstellt sind, müssen Sie diese als Teil der Syntax angeben.

Zugehörige Informationen

Die folgenden Dokumente können bei der Verwendung des XML Extender und der zugehörigen Produkte hilfreich sein:

Dokument	Bestellnummer	Beschreibung
Call Level Interface Guide and Reference	IBM Form SC09–2950	Dieses Buch beschreibt, wie mit CLI Anwendungen für den Zugriff auf DB2-Server geschrieben werden.
DB2 Application Development Guide	IBM Form SC09–2949	Dieses Buch beschreibt den Prozeß der Anwendungsentwicklung sowie das Codieren, Kompilieren und Ausführen von Anwendungsprogrammen, die über eingebettetes SQL und APIs auf die Datenbank zugreifen.
DB2 Extender page	N/A	Diese Seite enthält Informationen zu den DB2-Extendern sowie zu den für die Extender relevanten Technologien. Die Web-Adresse zu den DB2-Extendern lautet: http://www.software.ibm.com/data/db2/extenders
DB2 SQL Reference for Universal Database Parts 1 and 2	• Part 1: IBM Form SC09–2974 • Part 2: IBM Form SC09–2975	Dieses Buch beschreibt die SQL-Syntax und -Semantik sowie die Regeln der Sprache. Es enthält außerdem Informationen zu Inkompatibilität zwischen verschiedenen Releases, zu Produkteinschränkungen und zu Katalogsichten.
• *DB2 Universal Database Systemverwaltung: Implementierung* • *DB2 Universal Database Systemverwaltung: Optimierung* • *DB2 Universal Database Systemverwaltung: Konzept*	• Implementierung: IBM Form SC12–2877 • Optimierung: IBM Form SC12–2878 • Konzept: IBM Form SC12–2879	Diese Bücher beschreiben die Konzeption, Implementierung und Wartung einer DB2-Datenbank.

Dokument	Bestellnummer	Beschreibung
DB2 Universal Database Image, Audio und Video Extender Verwaltung und Programmierung	IBM Form SC12–2892	Dieses Buch beschreibt die Verwaltung einer DB2-Datenbank für Bild-, Ton- und Videodaten. Außerdem enthält es Informationen zur Verwendung der mit den Extendern bereitgestellten Anwendungsprogrammierschnittstellen für den Zugriff und die Bearbeitung dieser Datentypen.
DB2 Universal Database Text Extender Verwaltung und Programmierung	IBM Form SC12–2893	Dieses Buch beschreibt die Verwaltung einer DB2-Datenbank für Textdaten. Außerdem enthält es Informationen zur Verwendung der mit den Extendern bereitgestellten Anwendungsprogrammierschnittstellen für den Zugriff und die Bearbeitung dieser Datentypen.

Teil 1. Einführung

Dieser Teil des Handbuchs enthält einen Überblick zum XML Extender und darüber, wie Sie ihn in Ihren Geschäftsanwendungen einsetzen können.

Kapitel 1. Einführung in den XML Extender

Die Produktfamilie der IBM DB2 Extender bietet Managementlösungen für Daten und Metadaten zur Verarbeitung traditioneller und nicht traditioneller Daten. Der XML Extender hilft Ihnen, die Leistung der IBM DB2 Universal Database (DB2 UDB) mit der Flexibilität von XLM zu verbinden.

Der XML Extender von DB2 bietet die Möglichkeit zum Speichern und Aufrufen von XML-Dokumenten und zum Generieren von XML-Dokumenten aus vorhandenen relationalen Daten sowie zum Unterteilen (Zerlegen, Speichern nicht markierter Elemente oder Attributinhalte) von XML-Dokumenten in relationale Daten. XML Extender bietet neue Datentypen, Funktionen und gespeicherte Prozeduren zum Verwalten Ihrer XML-Daten in DB2.

Der XML Extender ist für die folgenden Betriebssysteme verfügbar:

- Windows NT
- AIX
- Sun Solaris
- Linux
- NUMA-Q

XML-Dokumente

In der Computerbranche ist eine Vielzahl von Anwendungen verfügbar, die verschiedene Stärken und Schwächen aufweisen. Die Benutzer können selbst auswählen, welche Anwendung für ihre jeweilige Task am besten geeignet ist. Da Benutzer jedoch dazu tendieren, Daten zwischen verschiedenen Anwendungen gemeinsam zu nutzen, sind sie ständig mit dem Problem konfrontiert, ihre Daten zu vervielfältigen, umzuwandeln, zu exportieren oder in verschiedenen Formaten zu speichern, die wiederum in anderen Anwendungen importiert werden können. Dies kann bei Geschäftsanwendungen ein kritisches Problem darstellen, da bei manchen dieser Umwandlungsprozesse ein Teil der Daten verlorengeht oder zumindest die Benutzer in einem relativ aufwendigen Prozeß sicherstellen müssen, daß die Daten konsistent sind. Dies kostet sowohl Zeit als auch Geld.

Anwendungsentwickler können dieses Problem heute unter anderem dadurch lösen, daß sie ODBC-Anwendungen *(Open Database Connectivity)* schreiben, um die Daten in einem Datenbankverwaltungssystem zu speichern. Von hier aus können die Daten verarbeitet und in der Form dargestellt werden, in der sie in einer anderen Anwendung benötigt werden. Die Datenbankan-

wendungen müssen so geschrieben werden, daß sie die Daten in die für eine bestimmte Anwendung erforderliche Form bringen. Die Anwendungen ändern sich jedoch häufig und veralten schnell. Anwendungen, die Daten in HTML umwandeln, bieten eine Lösung für die Präsentation; die Daten können jedoch nicht für andere Zwecke verwendet werden. Wenn es eine andere Möglichkeit gäbe, die Daten von der Präsentation zu trennen, könnte diese Methode als geeignete Form für den Austausch zwischen Anwendungen eingesetzt werden.

XML wurde als Ansatz zur Lösung dieses Problems entwickelt. XML ist ein Akronym für *eXtensible Markup Language* (erweiterbare Formatierungssprache). Sie ist erweiterbar insofern, als die Sprache selbst eine Metasprache darstellt, mit der Sie eine eigene Sprache entsprechend den Anforderungen Ihres Unternehmens definieren können. Sie verwenden XML zum Erfassen der Daten für Ihre spezifische Anwendung wie auch der Datenstruktur. XML ist nicht das einzige Austauschformat; es wird jedoch mittlerweile als Standard für den Datenaustausch akzeptiert. Durch die Einhaltung dieses Standards können Anwendungen endlich Daten gemeinsam nutzen, ohne sie in ein spezielles Format umwandeln zu müssen.

XML-Anwendungen

Da XML heute ein akzeptierter Standard für den Datenaustausch ist, werden immer mehr Anwendungen entwickelt, die mit diesem Standard arbeiten können.

Angenommen, Sie verwenden eine spezifische Projektverwaltungsanwendung und wollen einige der Daten auch in Ihrer Kalenderanwendung nutzen. Mit XML ist die Lösung dieser Aufgabe sehr einfach. In der heutigen vernetzten Welt kann ein Anbieter von Anwendungen nur mithalten, wenn er in seinen Anwendungen integrierte Dienstprogramme zum integrierten XML-Datenaustausch anbieten kann. In diesem Beispiel könnte also die Projektverwaltungsanwendung Tasks in XML exportieren, die wiederum in unveränderter Form in Ihre Kalenderanwendung übernommen werden können (sofern die Informationen einer akzeptierten DTD entspricht).

Warum XML und DB2?

Auch wenn XML viele Probleme durch die Bereitstellung eines Standardformats für den Datenaustausch löst, sind noch weitere Hindernisse zu überwinden. Beim Aufbau einer Anwendung für Unternehmensdaten müssen Sie u. a. folgende Fragen berücksichtigen:

- Wie häufig sollen die Daten repliziert werden?
- Welche Arten von Informationen sollen zwischen den Anwendungen gemeinsam genutzt werden?

- Wie kann schnell nach den gewünschten Informationen gesucht werden?
- Wie kann eine bestimmte Aktion, beispielsweise das Hinzufügen eines neuen Eintrags, einen automatischen Datenaustausch zwischen allen Anwendungen initiieren?

Diese Art von Problemen kann nur über ein Datenbankverwaltungssystem gelöst werden. Durch die Implementierung der XML-Informationen und Meta-Informationen in der Datenbank können Sie direkter (und schneller) die XML-Ergebnisse abrufen, die Ihre anderen Anwendungen für den jeweiligen Zweck benötigen. Hier unterstützt Sie XML Extender besonders effizient. Mit dem XML Extender nutzen Sie die Leistung von DB2 in vielen XML-Anwendungen.

Mit dem Inhalt Ihrer strukturierten XML-Dokumente in einer DB2-Datenbank können Sie strukturierte XML-Informationen mit Ihren traditionellen relationalen Daten kombinieren. Entsprechend der Anwendung können Sie auswählen, ob ganze XML-Dokumente in DB2 als nicht traditionelle benutzerdefinierte Datentypen gespeichert werden sollen, oder Sie können den XML-Inhalt als traditionelle Daten in relationalen Tabellen zuordnen. Für nicht traditionelle XML-Daten bietet der XML Extender zusätzlich zu der strukturellen Textsuche des DB2 UDB Text Extender die Möglichkeit, umfangreiche Datentypen von XML-Elementen oder Attributwerten zu suchen.

Mit der Anwendung XML Extender haben Sie folgende Möglichkeiten:

- Speichern vollständiger XML-Dokumente als *Spaltendaten* in einer Anwendungstabelle oder extern als lokale Tabelle, während die gewünschten XML-Elemente oder Attributwerte in *Seitentabellen* für die Suche extrahiert werden. Mit der XML-Spaltenmethode haben Sie folgende Möglichkeiten:
  - Ausführen einer Schnellsuche nach XML-Elementen oder Attributen von allgemeinen SQL-*Datentypen*, die in Seitentabellen extrahiert und indexiert wurden
  - Aktualisieren des Inhalts eines *XML-Elements* oder des Werts eines *XML-Attributs*
  - Dynamisches Extrahieren von XML-Elementen oder Attributen über SQL-Abfragen
  - Überprüfen von XML-Dokumenten beim Einfügen bzw. Aktualisieren
  - Ausführen einer strukturellen Textsuche mit dem Text Extender
- Zusammensetzen oder Zerlegen des Inhalts von XML-Dokumenten mit einer oder mehreren relationalen Tabellen über die XML-Objektgruppenspeicherungs- und Zugriffsmethode

XML in DB2 integrieren

XML Extender bietet die folgenden Funktionen zur Verwaltung und Nutzung von XML-Daten mit DB2:

- Verwaltungs-Tools zur Verwaltung der Integration von XML-Daten in relationalen Tabellen.
- Speicher- und Nutzungsmethoden für Ihre XML-Daten.
- Ein DTD-Repository, in dem Sie DTDs zur Prüfung von XML-Daten speichern können: DTD_REF
- Ein als DAD (Dateizugriffsdefinition) bezeichnetes Zuordnungsschema für die Zuordnung von XML-Dokumenten zu relationalen Daten

Verwaltungs-Tools

Die Verwaltungs-Tools zum XML Extender erleichtern das Aktivieren der Datenbank und der Tabellenspalten für XML und das Zuordnen der XML-Daten zu den relationalen DB2-Strukturen. Der XML Extender bietet verschiedene Verwaltungs-Tools, je nachdem, ob Sie eine Anwendung entwickeln oder einfach mit einem Assistentenprogramm (Wizard) arbeiten wollen. Sie können die Verwaltungs-Tasks für den XML Extender mit den folgenden Tools ausführen:

- Die XML Extender Verwaltungsassistenten bieten eine grafische Benutzerschnittstelle für die Verwaltungs-Tasks.
- Der Befehl **dxxadm** bietet eine Befehlszeilenoption für Verwaltungs-Tasks.
- Die gespeicherten Prozeduren des XML Extender für die Verwaltung enthalten Optionen für die Anwendungsentwicklung zu Verwaltungs-Tasks.

Speicher- und Zugriffsmethoden

XML Extender bietet zwei Speicher- und Zugriffsmethoden für die Integration von XML-Dokumenten in DB2: XML-Spalte und XML-Objektgruppe. Diese Methoden werden sehr unterschiedlich verwendet; sie können jedoch in derselben Anwendung eingesetzt werden.

XML-Spalte

Diese Methode ermöglicht das Speichern intakter XML-Dokumente in DB2. XML-Spalte eignet sich gut zum Aktivieren von Dokumenten. Die Dokumente werden in Spalten eingefügt, die für XML aktiviert sind und aktualisiert, abgerufen und durchsucht werden können. Element- und attributive Daten können DB2-Tabellen (Seitentabellen) zugeordnet werden; diese wiederum können für eine schnelle strukturelle Suche indexiert werden.

XML-Objektgruppe

Diese Methode ermöglicht das Zuordnen von XML-Dokumentstrukturen zu DB2-Tabellen, so daß Sie entweder XML-Dokumente aus vorhandenen DB2-Daten zusammensetzen oder XML-Dokumente in DB2-Daten zerlegen (nicht markierte Element- oder Attributinhalte speichern) können. Diese Methode eignet sich gut für Anwendungen zum Datenaustausch, besonders wenn der Inhalt der XML-Dokumente häufig aktualisiert wird.

DTD-Repository

Der XML Extender bietet ein XML *DTD-Repository (Document Type Definition)*, eine Gruppe von Deklarationen für XML-Elemente und -Attribute. Beim *Aktivieren* einer Datenbank für XML wird eine *DTD-Referenztabelle (DTD_REF)* erstellt. Jede Zeile dieser Tabelle stellt eine DTD mit zusätzlichen Metadaten-Informationen dar. Die Benutzer können auf diese Tabelle zugreifen, um ihre eigenen DTDs einzufügen. Die *DTDs* in der Tabelle DTD_REF werden zum Überprüfen von XML-Dokumenten verwendet.

Dokumentzugriffsdefinitionen (Document Access Definitions, DADs)

Sie geben in einer Dokumentzugriffsdefinition *Document Access Definition, DAD)* an, wie strukturierte XML-Dokumente verarbeitet werden sollen. Die DAD ist selbst ein formatiertes XML-Dokument. Sie ordnet bei Verwendung von XML-Spalten oder XML-Objektgruppen eine XML-Dokumentstruktur einer DB2-Datenbank zu. Die Struktur der DAD ist bei der Definition einer XML-Spalte anders als bei einer XML-Objektgruppe.

DAD-Dateien werden über die Tabelle XML_USAGE verwaltet; diese Tabelle wurde beim Aktivieren einer Datenbank für XML erstellt.

XML-Spalte: Strukturiertes Speichern und Abrufen von Dokumenten

Da XML alle erforderlichen Informationen zum Erstellen einer Gruppe von Dokumenten enthält, wollen Sie die Dokumentstruktur häufig genau in der jeweiligen Form speichern oder abrufen.

Wenn Sie beispielsweise in einer Publishing-Firma arbeiten, die Artikel auf dem Web bereitstellt, wollen Sie vielleicht ein Archiv der veröffentlichten Artikel verwalten. In einem solchen Szenario gibt Ihnen der XML Extender die Möglichkeit, XML-Artikel vollständig oder teilweise in einer Spalte einer DB2-Tabelle zu speichern. Diese Art von XML-Dokumentspeicherung wird als *XML-Spalte* bezeichnet, wie in Abb. 1 auf Seite 8 gezeigt.

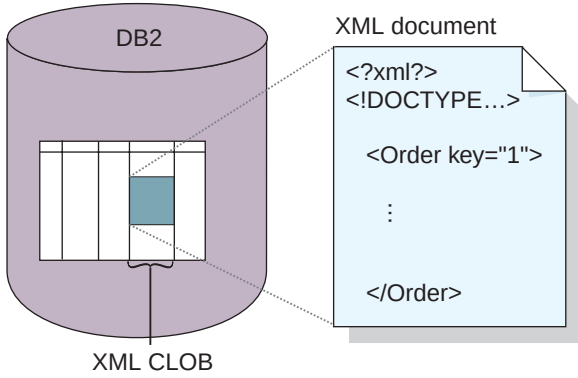


Abbildung 1. Speichern strukturierter XML-Dokumente in einer DB2-Tabellenspalte

Der XML Extender bietet die folgenden benutzerdefinierte Typen (User-Defined Types, UDTs) zur Verwendung mit XML-Spalten:

- XMLVarchar
- XMLCLOB
- XMLFILE

Alle XML Extender-UDTs haben das Präfix db2xml; dies entspricht dem *Schemanamen* der DB2 XML Extender-UDTs. Diese Datentypen werden zum Identifizieren des Speichertyps der XML-Dokumente in der Anwendungstabelle verwendet. Der XML Extender unterstützt herstellerspezifische unstrukturierte Dateien; Sie brauchen keine XML-Dokumente in DB2 zu speichern. Sie können XML-Dokumente auch als Dateien auf dem *lokalen Dateisystem* speichern durch Angeben eines lokalen Dateinamens.

Der DB2 XML Extender bietet leistungsstarke *benutzerdefinierte Funktionen (UDFs)* zum Speichern und Abrufen von XML-Dokumenten in XML-Spalten sowie zum Extrahieren von XML-Elementen oder Attributwerten. Eine UDF ist eine Funktion, die für das Datenbankverwaltungssystem definiert wird und auf die später in SQL-Abfragen verwiesen wird. Der XML Extender bietet die folgenden Typen von UDFs:

- Storage: Speichert intakte XML-Dokumente in für XML aktivierten Spalten als XML-Datentypen
- Extract: Extrahiert XML-Dokumente oder die Werte angegebener Elemente und Attribute als Basisdatentypen
- Update: Aktualisiert vollständige XML-Dokumente oder angegebene Element und Attributwerte

Die Extraktionsfunktionen ermöglichen die Ausführung leistungsstarker Suchfunktionen mit allgemeinen SQL-Datentypen. Darüber hinaus können Sie den DB2 UDB Text Extender mit dem XML Extender verwenden, um eine strukturierte und *Volltextsuche* nach Text in XML-Dokumenten durchzuführen. Mit dieser leistungsstarken Suchfunktion kann beispielsweise auch die Benutzerfreundlichkeit einer Web-Seite verbessert werden, in der große Mengen von lesbarem Text veröffentlicht werden, z. B. Zeitungsartikel oder *EDI-Anwendungen* (Electronic Data Interchange), die häufig Suchelemente oder -attribute enthalten.

Alle XML Extender-UDFs haben das Präfix db2xml; dies entspricht dem Schemanamen der DB2 XML Extender-UDFs. Die UDFs werden auf XML-UDTs angewendet, und sie werden für XML-Spalten verwendet.

Standortpfad

Ein *Standortpfad* ist eine Folge von *XML-Befehlen*, die ein XML-Element oder -attribut kennzeichnen. Der XML Extender verwendet den Standortpfad zum Kennzeichnen der Struktur des XML-Dokuments und zur Angabe des Kontexts für das Element bzw. Attribut. Ein einfacher Schrägstrich (/) als Pfad gibt an, daß der Kontext das gesamte Dokument ist. Der Standortpfad wird in den folgenden Situationen verwendet:

- Zur Identifikation der zu extrahierenden Elemente und Attribute beim Extrahieren von UDFs
- Zum Angeben der Zuordnungsdatei zwischen einem XML-Element bzw. -attribut und einer DB2-Spalte beim Definieren des Indexierungsschemas in der DAD für XML-Spalten
- Zum Angeben eines XML-Elements oder -Attributs bei der Verwendung des Text Extender für eine strukturelle Textsuche

Abb. 2 zeigt ein Beispiel eines Standortpfads und seiner Beziehung zur Struktur des XML-Dokuments.

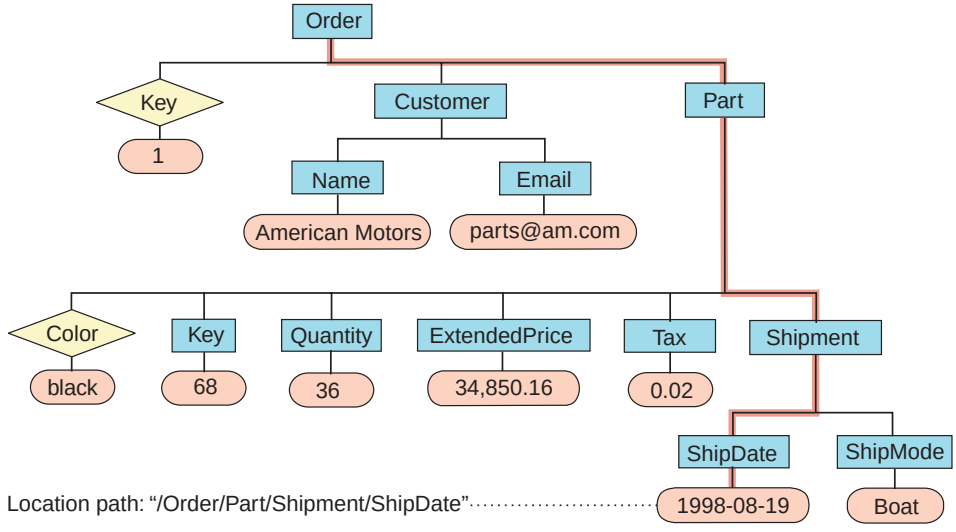


Abbildung 2. Speichern von Dokumenten als strukturierte XML-Dokumente in einer DB2-Tabellenspalte

Zum Angeben des Standortpfads verwendet der XML Extender eine Untermenge der *XML Stylesheet Language Transformation (XSLT)* und der *XML Path Language (XPath)*. In diesem Buch wird der Begriff *Standortpfad* verwendet, der in der Spezifikation für XPath definiert ist. Der Standortpfad ist eine Folge von XML-Befehlen, die ein XML-Element oder -attribut kennzeichnen. Außerdem verwendet dieses Buch die abgekürzte XSLT- oder XPath-Syntax des *absoluten Standortpfads*, der in den Spezifikationen von XPath definiert ist. Der absolute Standortpfad ist der vollständige Pfadname eines Objekts.

XSLT ist eine Sprache zur Umformung von XML-Dokumenten in andere XML-Dokumente. Sie wurde zur Verwendung als Teil der *XLS-Sprache* (XML Stylesheet Language), einer Sprache für Formatvorlagen, konzipiert. Zusätzlich zu XSLT enthält XSL ein XML-Vokabular zur Angabe der Formatierung. XSL definiert das Format eines XML-Dokuments durch Verwendung von XSLT zur Beschreibung, wie das Dokument in ein anderes XML-Dokument umgewandelt wird, das das Formatierungsvokabluar verwendet.

XPath ist eine Sprache zur Adressierung von Teilen eines XML-Dokuments, die zur Verwendung durch XSLT konzipiert wurde. Jeder Standortpfad kann mit Hilfe der für XPath definierten Syntax ausgedrückt werden.

Weitere Informationen zu XSLT und XPath finden Sie auf den folgenden Web-Seiten:

- Informationen zu XSLT siehe: http://www.w3.org/TR/WD-xslt
- Informationen zu XPath siehe: http://www.w3.org/TR/xpath

Die Syntax und Einschränkungen werden im Abschnitt „Standortpfad" auf Seite 56 beschrieben.

XML-Spalte Terminologie

In diesem Abschnitt werden die in diesem Handbuch verwendeten XML-Konzepte und die XML-Terminologie beschrieben.

Dokumentzugriffsdefinition (Document Access Definition, DAD)
Für XML-Spalte eine Zuordnung der XML-Dokumentstruktur zu DB2-Seitentabellen, die für strukturelle Abfragen indexiert wurden.

DXX_INSTALL
Das Installationsverzeichnis des XML Extender.

Seitentabelle
Zusätzliche Tabellen, die der XML Extender erstellt, um die Leistung beim Suchen von Elementen und Attributen in einer XML-Spalte zu verbessern.

XML-Spalte
Eine Methode zum Speichern oder Aufrufen von XML-Dokumenten durch Aktivieren einer DB2-Spalte für XML-Datentypen und zum Speichern eines intakten XML-Dokuments in der aktivierten Spalte. Bezeichnet auch eine Spalte, die mit einem der Verwaltungs-Tools für XML aktiviert wurde.

XML-Tabelle
Eine mit einem der Verwaltungs-Tools für XML aktivierte Anwendungstabelle, die eine oder mehrere Spalten enthält.

XML-UDF
Eine benutzerdefinierte DB2-Funktion, die vom XML Extender bereitgestellt wird.

XML-UDT
Ein benutzerdefinierter DB2-Typ, der vom XML Extender bereitgestellt wird.

XML-Objektgruppe: Integrierte Datenverwaltung

Traditionelle SQL-Daten werden entweder aus ankommenden XML-Dokumenten *zerlegt* oder für ausgehende XML-Dokumente *zusammengesetzt*. Wenn Ihre Daten mit anderen Anwendungen gemeinsam verwendet werden sollen, wollen Sie XML-Dokumente wahrscheinlich zusammensetzen und zerlegen und die Daten wie gewünscht verwalten, um die Vorteile der relationalen Funktionen von DB2 zu nutzen. Diese Art von XML-Dokumentspeicherung wird als *XML-Objektgruppe* bezeichnet.

Ein Beispiel einer XML-Objektgruppe finden Sie in Abb. 3.

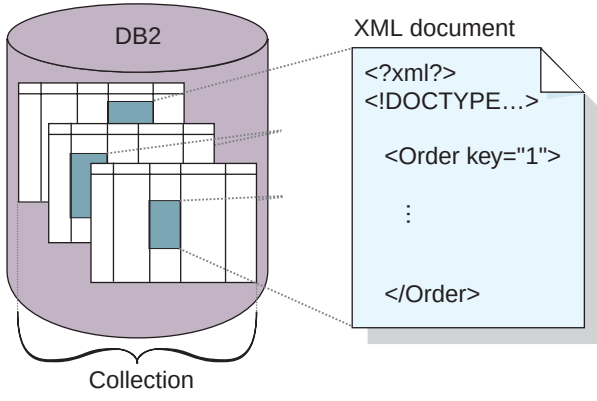


Abbildung 3. Speichern von Dokumenten als nicht markierte Daten in DB2-Tabellen

Die XML-Objektgruppe ist in einer DAD-Datei definiert; diese Datei gibt an, wie Elemente und Attribute einer oder mehreren relationalen Tabellen zugeordnet werden. Sie können einen Objektgruppennamen durch Aktivieren definieren und diesen Namen anschließend mit den gespeicherten Prozeduren zum Zusammensetzen oder Zerlegen von XML-Dokumenten verwenden.

Wenn Sie eine Objektgruppe in der DAD-Datei definieren, verwenden Sie eine von zwei Arten von Zuordnungsschemata, SQL-Zuordnung oder RDB-Knotenzuordnung. Die SQL-Zuordnung verwendet SQL SELECT-Anweisungen zum Definieren der DB2-Tabellen und Bedingungen für die Objektgruppe. Die RDB-Knotenzuordnung verwendet XPath-basierende RDB-Knoten zum Definieren der Tabellen, Spalten und Bedingungen.

Gespeicherte Prozeduren stehen zum Zusammensetzen oder Zerlegen von XML-Dokumenten zur Verfügung. Die gespeicherten Prozeduren verwenden das Präfix db2xml, das dem *Schemanamen* des XML Extender entspricht.

Verwenden Sie die folgenden gespeicherten Prozeduren mit XML-Objektgruppen:

- Zusammensetzung:
  - dxxGenXML(): verwendet eine DAD-Datei für eine XML-Objektgruppe zum Zusammensetzen von XML-Dokumenten
  - dxxRetrieveXML(): verwendet eine aktivierte XML-Objektgruppe zum Zusammensetzen von XML-Dokumenten
- Zerlegen:
  - dxxShredXML(): verwendet eine DAD-Datei für eine XML-Objektgruppe zum Zerlegen von XML-Dokumenten
  - dxxInsertXML(): verwendet eine aktivierte XML-Objektgruppe zum Zerlegen von XML-Dokumenten

XML-Objektgruppe - Terminologie

Die folgenden Begriffe beziehen sich speziell auf den XML Extender und werden in diesem Handbuch häufig verwendet.

Zusammensetzung
: Generieren von XML-Dokumenten aus vorhandenen relationalen Daten, wie in einer DAD-Datei definiert.

Zerlegen
: Speichern von XML-Dokumenten als nicht markierte Daten, wie in einer DAD-Datei definiert.

Dokumentzugriffsdefinition (Document Access Definition, DAD)
: Für XML-Objektgruppe eine Zuordnung von XML-Dokumentstrukturen zu DB2-Datenstrukturen für die Zusammensetzung oder das Zerlegen von XML-Dokumenten.

DXX_INSTALL
: Das Installationsverzeichnis des XML Extender.

XML-Objektgruppe
: Eine Methode zum Speichern und Aufrufen von XML-Daten mit einer Gruppe von relationalen Tabellen. Nicht markierte Daten können zu XML-Dokumenten zusammengesetzt oder aus XML-Dokumenten zerlegt werden. Bezeichnet außerdem die Gruppe von Tabellen, in die bzw. aus denen XML-Dokumente zusammengesetzt bzw. zerlegt werden.

Gespeicherte Prozeduren von XML
: Gespeicherte Prozeduren zum Zusammensetzen oder Zerlegen von XML-Dokumenten.

Kapitel 2. XML Extender - Erste Schritte

Dieses Kapitel beschreibt die ersten Schritte bei der Verwendung des XML Extender zum Aufrufen und Ändern von XML-Daten für Ihre Anwendungen. Anhand der beschriebenen Schritte in den einzelnen Lektionen des Lernprogramms können Sie eine Datenbank mit den bereitgestellten Musterdaten erstellen, SQL-Daten einem XML-Dokument zuordnen, XML-Dokumente in der Datenbank speichern und anschließend Daten in XML-Dokumenten suchen und daraus extrahieren.

In den Lektionen zur Verwaltung verwenden Sie das DB2-Befehlsfenster mit Verwaltungsbefehlen. Sie können diese Tasks mit dem XML Extender-Verwaltungsassistenten ausführen; dieses Programm wird ebenfalls in dem vorliegenden Handbuch beschrieben. In den Lektionen zur XML-Datenverwaltung verwenden Sie die vom XML Extender bereitgestellten UDFs und die gespeicherten Prozeduren. Die meisten Beispiele im weiteren Verlauf dieses Handbuchs verweisen auf die in diesem Kapitel verwendeten Musterdaten.

Erforderlich: Zum Ausführen der Lektionen in diesem Kapitel muß DB2 UDB Version 6.1 oder höher installiert sein. Bei Verwendung von Version 6.1 müssen Sie Fixpak 2 installieren. Außerdem wird bei den Arbeitsschritten in diesen Lektionen davon ausgegangen, daß Sie Windows NT® verwenden.

Die Lektionen haben folgenden Inhalt:

- Speichern eines intakten XML-Dokuments in einer DB2-Tabellenspalte
  - Planen des XML-UDT, in dem das Dokument und die XML-Elemente sowie die häufig zu suchenden Attribute gespeichert werden sollen.
  - Einrichten der Datenbank und der Tabellen
  - Aktivieren der Datenbank für XML
  - Einfügen der DTD in die DTD-Repository-Tabelle
  - Vorbereiten einer DAD für eine XML-Spalte
  - Hinzufügen einer Spalte zu einer vorhandenen Tabelle des XML-Typs
  - Aktivieren der neuen Spalte für XML
  - Erstellen von Indizes zu den Seitentabellen
  - Speichern eines XML-Dokuments in der XML-Spalte
  - Durchsuchen der XML-Spalte mit den XML Extender-UDFs
- Erstellen eines XML-Dokuments aus vorhandenen Daten
  - Planen der Datenstruktur des XML-Dokuments
  - Einrichten der Datenbank und der Tabellen

- Aktivieren der Datenbank für XML
- Vorbereiten einer DAD-Datei (Dokumentzugriffsdefinition) für eine XML-Objektgruppe
- Zusammensetzen des XML-Dokuments aus vorhandenen Daten
- Abrufen des XML-Dokuments aus der Datenbank

• Bereinigen der Datenbank

Szenario für die Lektionen

In diesen Lektionen arbeiten Sie für ACME Auto Direct, ein Unternehmen, das Automobile und LKW an Händlerniederlassungen verteilt. Sie haben zwei Aufgaben zu erledigen. Zuerst richten Sie ein System ein, in dem Bestellungen in der Datenbank SALES_DB zum Abfragen nach Vertriebsabteilung archiviert werden können. Ihre zweite Aufgabe ist es, Informationen in einer vorhandenen Bestellungsdatenbank, SALES_DB, zu erfassen und Schlüsselinformationen aus dieser Datenbank zu extrahieren, um sie in XML-Dokumenten zu speichern.

Lektion: Ein XML-Dokument in einer XML-Spalte speichern

Das Szenario

Sie haben die Aufgabe erhalten, Vertriebsdaten für die Serviceabteilung zu archivieren. Die Daten sind in XML-Dokumenten gespeichert, die dieselbe DTD verwenden. Die Serviceabteilung verwendet diese XML-Dokumente bei der Bearbeitung von Kundenanfragen und Reklamationen.

Die Serviceabteilung hat eine empfohlene Struktur für die XML-Dokumente bereitgestellt und angegeben, welche Elementdaten voraussichtlich am häufigsten abgefragt werden. Die XML-Dokumente sollen in der Tabelle SALES_TAB in der Datenbank SALES_DB gespeichert werden und schnell gesucht werden können. Die Tabelle SALES_DB wird zwei Spalten enthalten mit Daten zu jedem Verkauf sowie eine dritte Spalte, die das XML-Dokument enthalten soll. Diese Spalte hat den Namen ORDER.

Sie legen anschließend die XML-Datentypen fest, in denen das XML-Dokument gespeichert werden soll, und geben an, welche XML-Elemente und -Attribute häufig abgefragt werden. Als nächstes richten Sie die Datenbank SALES_DB für XML ein, erstellen die Tabelle SALES_TAB und aktivieren die Spalte ORDER, so daß Sie das intakte Dokument in DB2 speichern können. Außerdem fügen Sie eine DTD für das XML-Dokument zur Überprüfung ein und zum anschließenden Speichern des Dokuments als Datentyp XMLVARCHAR.

Wenn Sie die Spalte aktivieren, definieren Sie Seitentabellen zum Indexieren für die strukturelle Suche des Dokuments in einer Dokumentzugriffsdefinitionsdatei (DAD), einem XML-Dokument, das die Struktur der Seitentabellen festlegt. Beispiele zu der DAD-Datei, der DTD und dem XML-Dokument finden Sie in „Anhang B. Beispiele" auf Seite 285.

Die Tabelle SALES_TAB ist in Tabelle 1 beschrieben.

Tabelle 1. Tabelle SALES_TAB

Spaltenname	Datentyp
INVOICE_NUM	CHAR(6) NOT NULL PRIMARY KEY
SALES_PERSON	VARCHAR(20)
ORDER	XMLVARCHAR

Planung

Bevor Sie mit dem XML Extender Ihre Dokumente speichern, müssen Sie die Struktur des XML-Dokuments verstehen, um festlegen zu können, wie das Dokument durchsucht werden soll. Beim Planen, wie das Dokument durchsucht werden soll, müssen Sie folgendes festlegen:

- Den benutzerdefinierten XML-Typ, in dem das XML-Dokument gespeichert werden soll.
- Die XML-Elemente und -Attribute, die die Serviceabteilung häufig durchsucht, so daß sie im Interesse einer höheren Leistung indexiert werden kann.

In den folgenden Abschnitten wird beschrieben, wie Sie diese Entscheidungen treffen können.

Die Struktur des XML-Dokuments

Die Struktur des XML-Dokuments für diese Lektion verwendet Informationen zu einer bestimmten Bestellung, die nach Bestellschlüssel auf der Ausgangsebene und anschließend nach Kunde, Teil und Versandinformationen auf der nächsten Stufe strukturiert ist. Das XML-Dokument ist in Abb. 4 auf Seite 18 beschrieben.

Diese Lektion enthält außerdem eine Beispiel-DTD, die Ihnen hilft, die XML-Dokumentstruktur besser zu verstehen und zu überprüfen. Die DTD-Datei ist in „Anhang B. Beispiele" auf Seite 285 dargestellt. Sie entspricht der Struktur in Abb. 4 auf Seite 18.

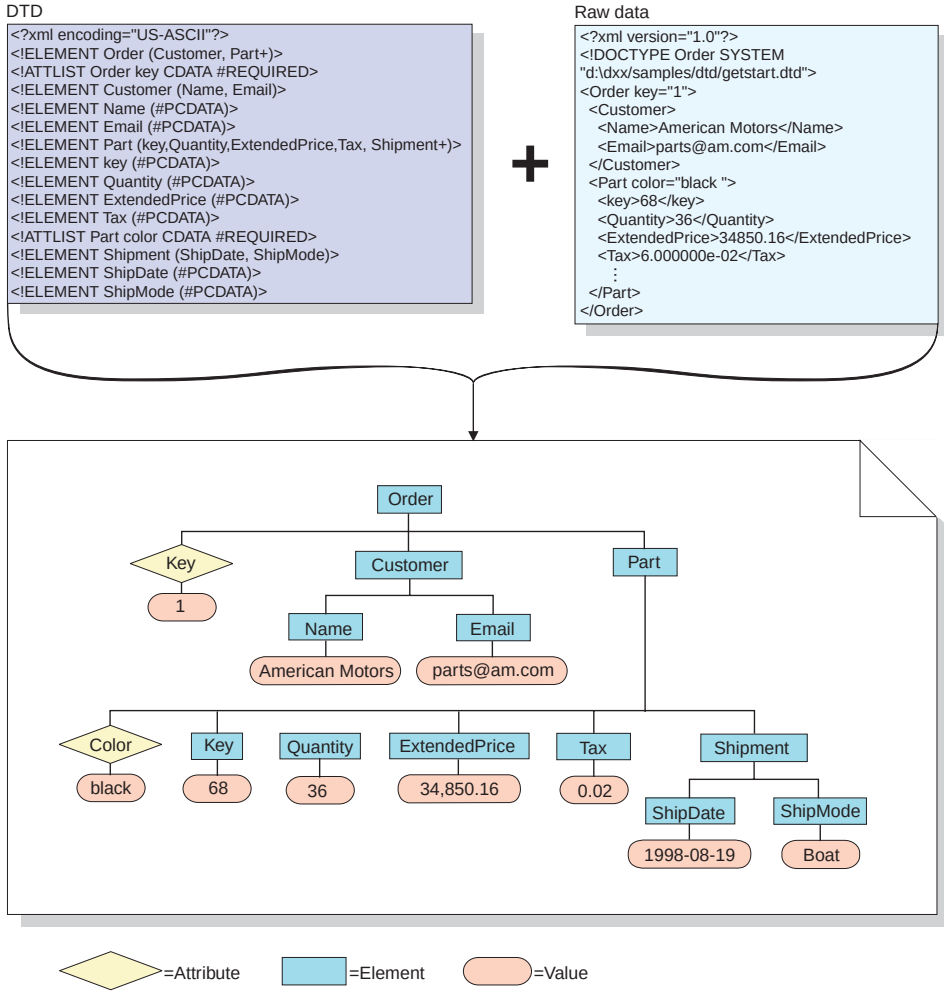


Abbildung 4. Die hierarchische Struktur der DTD und des XML-Dokuments

Den XML-Datentyp für die XML-Spalte festlegen

Der XML Extender bietet benutzerdefinierte XML-Datentypen, in denen Sie eine Spalte definieren, die die XML-Dokumente enthalten soll. Diese Datentypen sind:

- XMLVarchar: für in DB2 gespeicherte kleine Dokumente
- XMLCLOB: für in DB2 gespeicherte große Dokumente
- XMLFILE: für Dokumente, die außerhalb von DB2 gespeichert sind

In dieser Lektion speichern Sie ein kleines Dokument in DB2 und verwenden dazu den Datentyp XMLVarchar.

Zu durchsuchende Elemente und Attribute festlegen

Wenn Sie die XML-Dokumentstruktur und die Anforderungen der Anwendung verstehen, können Sie festlegen, welche Elemente und Attribute durchsucht werden sollen, welche Elemente und Attribute am häufigsten durchsucht und extrahiert werden sollen oder welche am aufwendigsten zu durchsuchen sind. Die Serviceabteilung hat darauf hingewiesen, daß sie häufig den Bestellschlüssel, den Kundennamen, den Preis und das Versanddatum einer Bestellung abfragen wird und bei diesen Abfragen auf einen schnellen Suchlauf angewiesen ist. Diese Informationen sind in den Elementen und Attributen der XML-Dokumentstruktur enthalten. Tabelle 2 beschreibt die Standortpfade aller Elemente und Attribute.

Tabelle 2. Zu durchsuchende Elemente und Attribute

Daten	Standortpfad
Bestellschlüssel	/Order/@key
Kunde	/Order/Customer/Name
Preis	/Order/Part/ExtendedPrice
Versanddatum	/Order/Part/Shipment/ShipDate

Das XML-Dokument den Seitentabellen zuordnen

In diesem Lernprogramm erstellen Sie eine DAD-Datei für die XML-Spalte zum Speichern des XML-Dokuments in DB2. Diese Datei ordnet auch den Inhalt der XML-Elemente und -Attribute den für die Indexierung verwendeten DB2-Seitentabellen zu, was die Suchleistung verbessert. Im letzten Abschnitt haben Sie gesehen, welche Elemente und Attribute durchsucht werden sollen. In diesem Abschnitt erfahren Sie mehr über die Zuordnung dieser Elemente und Attributwerte zu DB2-Tabellen, die indexiert werden können.

Nach dem Kennzeichnen der zu durchsuchenden Elemente und Attribute legen Sie fest, wie diese in den Seitentabellen organisiert werden sollen - wie viele Tabellen verwendet werden und welche Spalten sich in welcher Tabelle befinden sollen. Normalerweise organisieren Sie die Seitentabellen durch Ablegen ähnlicher Informationen in der gleichen Tabelle. Die Struktur wird auch dadurch festgelegt, ob der Standortpfad der Elemente mehr als einmal im Dokument wiederholt werden kann. In unserem Dokument kann das Element ″Teil″ beispielsweise mehrfach wiederholt werden; die Elemente ″Preis″ und ″Datum″ können somit mehrfach vorkommen. Elemente, die mehrfach vorkommen können, müssen in ihren eigenen Tabellen stehen.

Darüber hinaus müssen Sie auch festlegen, welche DB2-Basistypen die Element- und Attributwerte verwenden sollen. Normalerweise kann dies sehr einfach über das Format der Daten festgelegt werden. Wenn es sich bei den Daten um Text handelt, wählen Sie VARCHAR; sind es ganze Zahlen, wählen

Sie INTEGER. Wenn es sich bei den Daten um ein Datum handelt und Sie eine bereichsweise Suche ausführen wollen, wählen Sie DATE.

In diesem Lernprogramm sind die Elemente und Attribute den folgenden Seitentabellen zugeordnet:

ORDER_SIDE_TAB

Spaltenname	Datentyp	Standortpfad	Mehrfaches Vorkommen?
ORDER_KEY	INTEGER	/Order/@key	No
CUSTOMER	VARCHAR(16)	/Order/Customer/Name	No

PART_SIDE_TAB

Spaltenname	Datentyp	Standortpfad	Mehrfaches Vorkommen?
PRICE	DECIMAL(10,2)	/Order/Part/ExtendedPrice	Yes

SHIP_SIDE_TAB

Spaltenname	Datentyp	Standortpfad	Mehrfaches Vorkommen?
DATE	DATE	/Order/Part/Shipment/ShipDate	Yes

Für dieses Lernprogramm wurde eine Reihe von Prozeduren bereitgestellt, mit denen Sie Ihre Umgebung einrichten können. Diese Prozeduren befinden sich im Verzeichnis*DXX_INSTALL*\samples\cmd (*DXX_INSTALL* gibt hierbei das Laufwerk und das Verzeichnis an, in dem Sie den XML Extender installiert haben, z. B. `c:\dxx\samples\cmd`); sie heißen wie folgt:

getstart_db.cmd
Erstellt die Datenbank und füllt die vier Tabellen.

getstart_prep.cmd
Bindet die Datenbank an die gespeicherten Prozeduren des XML Extender und die DB2-CLI.

getstart_insertDTD.cmd
Fügt die DTD ein, die zur Überprüfung des XML-Dokuments in der XML-Spalte verwendet wird.

getstart_createTabCol.cmd
Erstellt eine Anwendungstabelle mit einer für XML aktivierten Spalte.

getstart_alterTabCol.cmd
Ändert die Anwendungstabelle durch Hinzufügen der Spalte, die für XML aktiviert wird.

getstart_enableCol.cmd
Aktiviert die XML-Spalte.

getstart_createIndex.cmd
Erstellt Indizes zu den Seitentabellen für die XML-Spalte

getstart_insertXML.cmd
Fügt das XML-Dokument in die XML-Spalte ein.

getstart_queryCol.cmd
Führt eine ausgewählte Anweisung mit der Anwendungstabelle aus und gibt das XML-Dokument zurück.

getstart_clean.cmd
Bereinigt die Lernprogrammumgebung.

Einstellen

In diesem Abschnitt bereiten Sie die Datenbank für die Verwendung mit dem XML Extender vor. Sie führen hierzu die folgenden Schritte aus:

1. Datenbank erstellen.
2. Datenbank aktivieren.

Datenbank erstellen

In diesem Abschnitt richten Sie mit einem entsprechenden Befehl die Datenbank ein. Dieser Befehl erstellt eine Musterdatenbank, stellt die Verbindung zu dieser Datenbank her, erstellt die Tabellen für die Daten und fügt die Daten anschließend ein.

So erstellen Sie die Datenbank:

1. Wechseln Sie zum Verzeichnis *DXX_INSTALL*`\samples\cmd`, wobei *DXX_INSTALL* das Laufwerk und Verzeichnis angibt, in dem der XML Extender installiert wurde. In diesen Lektionen wird von dem Verzeichnis `c:\dxx` ausgegangen. Passen Sie diese Angaben entsprechend an, wenn Sie ein anderes Laufwerk und Verzeichnis verwenden.
2. Öffnen Sie das DB2-Befehlsfenster über das Windows NT-Startmenü, oder geben Sie den folgenden Befehl über die Windows NT-Eingabeaufforderung ein:
   ```
   DB2CMD
   ```
3. Führen Sie über das DB2-Befehlsfenster aus den folgenden Befehl aus:
   ```
   getstart_db.cmd
   ```

Datenbank aktivieren

Zum Speichern von XML-Informationen in der Datenbank müssen Sie diese für den XML Extender aktivieren. Wenn Sie eine Datenbank für XML aktivieren, führt der XML Extender folgende Schritte aus:

- Erstellt alle benutzerdefinierten Typen (UDTs) und benutzerdefinierten Funktionen (UDFs).
- Erstellt die Steuertabellen und füllt sie mit den erforderlichen Metadaten für den XML Extender.
- Erstellt das db2xml-Schema und ordnet die erforderlichen Berechtigungen zu.

So aktivieren Sie die Datenbank für XML:

Führen Sie vom DB2-Befehlsfenster aus die folgende Prozedur aus, um die Datenbank SALES_DB zu aktivieren:

```
getstart_prep.cmd
```

Diese Prozedur bindet die Datenbank an die gespeicherten Prozeduren des XML Extender und die DB2-CLI. Außerdem führt diese Prozedur die Befehlsoption **dxxadm** aus, mit der die Datenbank aktiviert wird:

```
dxxadm enable_db SALES_DB
```

XML-Spalte erstellen

Der XML Extender bietet eine Methode zum Speichern und Aufrufen ganzer XML-Dokumente in der Datenbank; diese Methode wird als XML-Spalte bezeichnet. Mit der Methode "XML-Spalte" können Sie das Dokument mit dem XMLFILE-Typ speichern, die Spalte in Seitentabellen indexieren und das XML-Dokument anschließend abfragen oder durchsuchen. Diese Speichermethode ist besonders nützlich für Archivierungsanwendungen, in denen Dokumente nicht sehr häufig aktualisiert werden. Für dieses Lernprogramm speichern Sie das bereitgestellte XML-Dokument in der XML-Spalte.

In dieser Lektion speichern Sie das Dokument als XMLVARCHAR. Beim Speichern des Dokuments führen Sie die folgenden Aktionen aus:

1. Einfügen der DTD für das XML-Dokument in die DTD-Referenztabelle DTD_REF.
2. Vorbereiten einer DAD-Datei, die die Position des XML-Dokuments und der Seitentabellen für die strukturelle Suche angibt.
3. Hinzufügen einer Spalte in der Tabelle SALES_TAB mit einem benutzerdefinierten XML-Typ XMLVARCHAR.
4. Aktivieren der Spalte für XML.
5. Indexieren der Seitentabellen für die strukturelle Suche.
6. Speichern des Dokuments mit einer benutzerdefinierten Funktion, die von dem XML Extender bereitgestellt wird.

Die DTD im DTD-Repository speichern

Sie können mit einer DTD die XML-Daten in einer XML-Spalte überprüfen. Der XML Extender erstellt eine Tabelle in der für XML aktivierten Datenbank mit dem Namen DTD_REF. Die Tabelle wird als DTD-Referenz bezeichnet und steht zum Speichern von DTDs zur Verfügung. Wenn Sie XML-Dokumente überprüfen wollen, müssen Sie die DTD in diesem Repository speichern. Die DTD für dieses Lernprogramm lautet `c:\dxx\samples\dtd\getstart.dtd`.

So fügen Sie die DTD ein:

Geben Sie über das DB2-Befehlsfenster den folgenden SQL INSERT-Befehl in einer einzigen Zeile ein:

```
DB2 CONNECT TO SALES_DB
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'benutzer1',
    'user1', 'benutzer1')
```

Sie können auch die folgende Befehlsdatei zum Einfügen der DTD ausführen:

```
getstart_insertDTD.cmd
```

DAD-Datei vorbereiten

Die DAD-Datei für die XML-Spalte hat eine einfache Struktur. Sie geben als Speichermethode ″XML-Spalte″ an und definieren die Tabellen und Spalten für die Indexierung.

In den folgenden Schritten werden die Elemente in der DAD als *Befehle* und die Elemente Ihrer XML-Dokumentstruktur als *Elemente* bezeichnet. Ein Muster einer DAD-Datei ähnlich der von Ihnen erstellten finden Sie in `c:\dxx\samples\dad\getstart_xcolumn.dad`. Dieses Muster weist einige kleine Unterschiede gegenüber der in den folgenden Schritten generierten Datei auf. Wenn Sie das Muster für die Lektion verwenden, beachten Sie, daß die Dateipfade für Ihre Umgebung anders sein können und daß der Wert <validation> auf NO statt auf YES gesetzt ist.

So bereiten Sie die DAD-Datei vor:

1. Öffnen Sie einen Texteditor, und nennen Sie die Datei `getstart_xcolumn.dad`

   Beachten Sie, daß bei allen in der DAD-Datei verwendeten Markierungen zwischen Groß- und Kleinbuchstaben unterschieden wird.
2. Erstellen Sie den Kopfteil der DAD-Datei mit den Deklarationen für XML und den Dokumententyp.

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   Die DAD-Datei ist ein XML-Dokument und erfordert XML-Deklarationen.

3. Fügen Sie Anfangs- und Endbefehle <DAD></DAD> ein. Alle weiteren Befehle befinden sich innerhalb dieser Befehle.
4. Fügen Sie Anfangs- und Endbefehle <DTDID></DTDID> ein, um die DTD-ID anzugeben, die die DAD-Datei der DTD des XML-Dokuments zuordnet und die DTD-Position des Clients angibt.

   ```
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   ```

   Stellen Sie sicher, daß diese Zeichenfolge mit den Werten im ersten Parameter übereinstimmt, wenn Sie die DTD in die DTD-Referenztabelle unter „Die DTD im DTD-Repository speichern" auf Seite 23 einfügen. Der Pfad, den Sie für die DTD-ID verwendet haben, kann beispielsweise ein anderer sein als in der Zeichenfolge oben angegeben, wenn Sie mit einem anderen Laufwerk arbeiten.
5. Geben Sie Anfangs- und Endbefehle <validation></validation> an, um festzulegen, daß der XML Extender die XML-Dokumentstruktur mit der in der DTD-Repository-Tabelle eingefügten DTD überprüfen soll.

   ```
   <validation>YES</validation>
   ```

   Der Wert von <validation> muß in Großbuchstaben eingegeben werden.
6. Fügen Sie Anfangs- und Endbefehle <Xcolumn></Xcolumn> ein, um die Speichermethode "XML-Spalte" festzulegen. Diese Methode legt fest, daß die XML-Daten in einer XML-Spalte gespeichert werden sollen.

   ```
   <Xcolumn>
   </Xcolumn>
   ```
7. Fügen Sie Anfangs- und Endbefehle <table></table> für jede zu generierende Seitentabelle ein.

   ```
   <Xcolumn>
        <table name="order_side_tab">
   </table>
        <table name="part_side_tab">
   </table>
        <table name="ship_side_tab">
   </table>
   </Xcolumn>
   ```
8. Fügen Sie Anfangs- und Endbefehle <column></column> für jede Spalte ein, die in die Seitentabelle eingefügt werden soll. Jeder Befehl <column> hat vier Attribute:
   - **name:** der Name der Spalte
   - **type:** der Datentyp der Spalte
   - **path:** der Standortpfad des entsprechenden Elements in dem XML-Dokument entsprechend der XPath-Syntax. Die Syntax für den Standortpfad ist im Abschnitt „Standortpfad" auf Seite 56 beschrieben.
   - **multi-occurrence:** Angabe, ob der Standortpfad in der XML-Dokumentstruktur mehrfach vorkommen kann

```
<Xcolumn>
     <table name="order_side_tab">
  <column name="order_key"
          type="integer"
          path="/Order/@key"
          multi_occurrence="NO"/>
  <column name="customer"
          type="varchar(50)"
          path="/Order/Customer/Name"
          multi_occurrence="NO"/>
</table>
     <table name="part_side_tab">
  <column name="price"
          type="decimal(10,2)"
          path="/Order/Part/ExtendedPrice"
          multi_occurrence="YES"/>
</table>
     <table name="ship_side_tab">
  <column name="date"
          type="DATE"
          path="/Order/Part/Shipment/ShipDate"
          multi_occurrence="YES"/>
</table>
</Xcolumn>
```

9. Stellen Sie sicher, daß Sie einen Endbefehl </Xcolumn> nach dem letzten Befehl </table> verwenden.
10. Stellen Sie sicher, daß Sie einen Endbefehl </DAD> nach dem letzten Befehl </Xcolumn> verwenden.
11. Speichern Sie die Datei unter dem Namen `getstart_xcolumn.dad`.

Sie können die eben erstellte Datei mit der Beispieldatei `c:\dxx\samples\dad\getstart_xcolumn.dad` vergleichen. Diese Datei ist eine Arbeitskopie der zum Aktivieren der XML-Spalte und zum Erstellen der Seitentabellen erforderlichen DAD-Datei. Die Beispieldatei enthält Pfadanweisungen, die für eine erfolgreiche Ausführung eventuell an Ihre Umgebung angepaßt werden müssen.

Tabelle SALES_TAB erstellen

In diesem Abschnitt erstellen Sie die Tabelle SALES_TAB. Zunächst enthält diese Tabelle zwei Spalten mit den Verkaufsinformationen für die Bestellung.

So erstellen Sie die Tabelle:

Geben Sie über das DB2-Befehlsfenster die folgende Anweisung CREATE TABLE ein:

```
DB2 CONNECT TO SALES_DB

DB2 CREATE TABLE SALES_TAB(INVOICE_NUM CHAR(6) NOT NULL PRIMARY KEY,
        SALES_PERSON VARCHAR(20))
```

Alternativ dazu können Sie auch die folgende Befehlsdatei zum Erstellen der Tabelle ausführen:

```
getstart_createTabCol.cmd
```

Spalte des XML-Typs hinzufügen

Fügen Sie jetzt der Tabelle SALES_TAB eine neue Spalte hinzu. Diese Spalte enthält das intakte XML-Dokument, das Sie zuvor erstellt haben; es muß den UDT XML haben. Der XML Extender bietet mehrere Datentypen, die in „Kapitel 8. Benutzerdefinierte Typen des XML Extender" auf Seite 181 beschrieben werden. In diesem Lernprogramm speichern Sie das Dokument als XMLVARCHAR.

So fügen Sie die Spalte mit dem XML-Typ hinzu:

Geben Sie über das DB2-Befehlsfenster die folgende SQL-Anweisung ein:

```
DB2 ALTER TABLE SALES_TAB ADD ORDER DB2XML.XMLVARCHAR
```

Alternativ dazu können Sie auch die folgende Befehlsdatei zum Ändern der Tabelle ausführen:

```
getstart_alterTabCol.cmd
```

XML-Spalte aktivieren

Nachdem Sie die Spalte mit dem Typ XML erstellt haben, aktivieren Sie sie für den XML Extender. Wenn Sie die Spalte aktivieren, liest der XML Extender die DAD-Datei und erstellt die Seitentabellen. Vor dem Aktivieren der Spalte müssen Sie folgende Punkte ausführen:

- Festlegen, ob eine Standardsicht der XML-Spalte erstellt werden soll, die das XML-Dokument und die Spalten der Seitentabelle enthält. Sie können beim Abfragen des XML-Dokuments die Standardsicht verwenden. In dieser Lektion geben Sie die Sicht mit dem Parameter `-v` an.
- Festlegen, ob ein Primärschlüssel als *ROOT ID*, der Spaltenname des Primärschlüssels in der Anwendungstabelle und eine eindeutige Kennung, die alle Seitentabellen der Anwendungstabelle zuordnet, angegeben werden soll. Wenn Sie keinen Primärschlüssel angegeben, fügt der XML Extender der Anwendungstabelle und den Seitentabellen die Spalte DXXROOT_ID hinzu. Die Spalte ROOT_ID verbindet die Anwendungs- und Seitentabellen und gibt dem XML Extender die Möglichkeit, die Seitentabellen bei der Aktualisierung des XML-Dokuments automatisch zu aktualisieren. In dieser Lektion geben Sie den Namen des Primärschlüssels im Befehl (INVOICE_NUM) mit dem Parameter `-r` an. Der XML Extender verwendet die angegebene Spalte als ROOT_ID und fügt die Spalte den Seitentabellen hinzu.
- Festlegen, ob Sie einen Tabellenbereich angeben oder den Standardtabellenbereich verwenden wollen. In dieser Lektion verwenden Sie den Standardtabellenbereich.

So aktivieren Sie die Spalte für XML:

Geben Sie über das DB2-Befehlsfenster den folgenden Befehl ein:

```
dxxadm enable_column SALES_DB SALES_TAB ORDER GETSTART_XCOLUMN.DAD
        -v SALES_ORDER_VIEW -r INVOICE_NUM
```

Alternativ dazu können Sie auch die folgende Befehlsdatei zum Aktivieren der Spalte für XML ausführen:

```
getstart_enableCol.cmd
```

Der XML Extender erstellt die Seitentabellen mit der Spalte INVOICE_NUM und erstellt die Standardsicht.

Wichtig: Ändern Sie die Seitentabellen nicht. Sie sollten das XML-Dokument nur mit den vom XML Extender bereitgestellten UDFs aktualisieren. Der XML Extender aktualisiert die Seitentabellen automatisch, wenn Sie das XML-Dokument in der XML-Spalte aktualisieren.

Spalten und Seitentabellen anzeigen

Beim Aktivieren der XML-Spalte haben Sie eine Sicht der XML-Spalten und -Seitentabellen erstellt. Sie können diese Sicht beim Arbeiten mit der XML-Spalte verwenden.

So zeigen Sie die XML-Spalte und die Spalten der XML-Seitentabellen an:

Geben Sie über das DB2-Befehlsfenster die folgende SQL SELECT-Anweisung ein:

```
DB2 SELECT * FROM SALES_ORDER_VIEW
```

Die Sicht zeigt die Spalten in den Seitentabellen, wie in der Datei `getstart_xcolumn.dad` angegeben.

Indizes zu den Seitentabellen erstellen

Das Erstellen von Indizes zu Seitentabellen ermöglicht ein schnelles strukturelles Durchsuchen des XML-Dokuments. In diesem Schritt erstellen Sie Indizes zu Schlüsselspalten in den Seitentabellen, die beim Aktivieren der XML-Spalte ORDER erstellt wurden. Die Serviceabteilung hat angegeben, welche Spalten ihre Mitarbeiter voraussichtlich am häufigsten abfragen werden. Tabelle 3 beschreibt diese zu indexierenden Spalten:

Tabelle 3. Zu indexierende Spalten der Seitentabellen

Spalte	Seitentabelle
ORDER_KEY	ORDER_SIDE_TAB
CUSTOMER	ORDER_SIDE_TAB
PRICE	PART_SIDE_TAB

Tabelle 3. Zu indexierende Spalten der Seitentabellen (Forts.)

Spalte	Seitentabelle
DATE	SHIP_SIDE_TAB

So indexieren Sie die Seitentabellen:

Geben Sie über das DB2-Befehlsfenster die folgenden SQL-Befehle ein:

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)

DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)

DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)

DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

Alternativ dazu können Sie auch die folgende Befehlsdatei zum Erstellen der Indizes ausführen:

```
getstart_createIndex.cmd
```

XML-Dokument speichern

Sie haben eine Spalte erstellt, die das XML-Dokument enthalten kann, und die Seitentabellen indexiert; jetzt können Sie das Dokument mit den Funktionen des XML Extender speichern. Beim Speichern von Daten in einer XML-Spalte verwenden Sie entweder die Standardumsetzungsfunktionen oder die XML ExtenderUDFs. Da Sie ein Objekt des Basistyps VARCHAR in einer Spalte des XML-UDT XMLVARCHAR speichern, verwenden Sie die Standardumsetzungsfunktion. Informationen zu den Standardumsetzungsfunktionen zum Speichern sowie zu den vom XML Extender bereitgestellten UDFs finden Sie im Abschnitt „Daten speichern” auf Seite 128.

So speichern Sie das XML-Dokument:

Wichtig: Öffnen Sie das XML-Dokument `c:\dxx\samples\xml\getstart.xml`. Stellen Sie sicher, daß der Dateipfad in DOCTYPE der in der DAD angegebenen DTD-ID und der beim Einfügen in das DTD-Repository angegebenen DTD-ID entspricht. Sie können diese Übereinstimmung überprüfen, indem Sie die Tabelle db2xml.DTD_REF abfragen und das DTD-ID-Element in der DAD-Datei prüfen. Wenn Sie eine vom Standardwert abweichende Laufwerks- und Verzeichnisstruktur verwenden, müssen Sie eventuell den Pfad in der DOCTYPE-Deklaration verwenden.

Geben Sie über das DB2-Befehlsfenster den folgenden SQL INSERT-Befehl ein:

```
DB2 INSERT INTO SALES_TAB (INVOICE_NUM, SALES_PERSON, ORDER) VALUES('123456',
'Sriram Srinivasan', db2xml.XMLVarcharFromFile('c:\dxx\samples\cmd\
getstart.xml'))
```

Beim Speichern des XML-Dokuments aktualisiert der XML Extender die Seitentabellen automatisch.

Alternativ dazu können Sie auch die folgende Befehlsdatei zum Speichern des Dokuments ausführen:

```
getstart_insertXML.cmd
```

Zum Sicherstellen, daß die Tabellen aktualisiert wurden, führen Sie über das DB2-Befehlsfenster die folgenden SELECT-Anweisungen für die Tabellen aus:

```
DB2 SELECT * FROM SALES_TAB

DB2 SELECT * FROM PART_SIDE_TAB

DB2 SELECT * FROM ORDER_SIDE_TAB

DB2 SELECT * FROM SHIP_SIDE_TAB
```

XML-Dokument durchsuchen

Sie können das XML-Dokument mit einer direkten Abfrage in einem Abgleich zu den Seitentabellen durchsuchen. In diesem Schritt suchen Sie nach allen Bestellungen mit einem Preis über 2500,00.

So fragen Sie die Seitentabellen ab:

Geben Sie über das DB2-Befehlsfenster den folgenden SELECT-Befehl ein:

```
DB2 "SELECT DISTINCT SALES_PERSON FROM SALES_TAB S, PART_SIDE_TAB P
   WHERE PRICE > 2500.00 AND
   S.INVOICE_NUM=P.INVOICE_NUM"
```

In den Ergebnisdaten sollten die Namen der Verkäufer angezeigt werden, die einen Artikel mit einem Preis über 2500,00 verkauft haben.

Alternativ dazu können Sie auch die folgende Befehlsdatei zum Durchsuchen des Dokuments ausführen:

```
getstart_queryCol.cmd
```

Sie haben hiermit das Lernprogramm ″Erste Schritte″ zum Speichern von XML-Dokumenten in DB2-Tabellen abgeschlossen. Viele Beispiele in diesem Handbuch basieren auf diesen Lektionen.

Lektion: XML-Dokument zusammensetzen

Das Lernprogramm-Szenario

Sie wurden mit der Aufgabe betraut, Informationen in einer vorhandenen Bestellungsdatenbank, SALES_DB, zu erfassen und Schlüsselinformationen aus dieser Datenbank zu extrahieren, um sie in XML-Dokumenten zu speichern. Die Serviceabteilung verwendet diese XML-Dokumente anschließend bei der Bearbeitung von Kundenanfragen und Reklamationen. Die Serviceabteilung hat angefordert, daß spezifische Daten einbezogen werden, und eine empfohlene Struktur für die XML-Dokumente bereitgestellt.

Sie setzen mit vorhandenen Daten ein XML-Dokument `getstart.xml` aus den Daten in diesen Tabellen zusammen.

Außerdem planen und erstellen Sie eine DAD-Datei, die Spalten aus den zugehörigen Tabellen einer XML-Dokumentstruktur zuordnet und so einen Bestellungsdatensatz zur Verfügung stellt. Da dieses Dokument aus mehreren Tabellen zusammengesetzt wird, erstellen Sie eine XML-Objektgruppe und ordnen diese Tabellen einer XML-Struktur und einer DTD zu. Sie verwenden diese DTD zum Definieren der Struktur des XML-Dokuments. Sie können mit der DTD auch das zusammengesetzte XML-Dokument in Ihren Anwendungen überprüfen.

Die vorhandenden Daten aus der Datenbank für das XML-Dokument sind in den folgenden Tabellen beschrieben. Die Spaltennamen in *Kursivschrift* sind Spalten, die die Serviceabteilung in der XML-Dokumentstruktur angefordert hat.

ORDER_TAB

Spaltenname	Datentyp
ORDER_KEY	INTEGER
CUSTOMER	VARCHAR(16)
CUSTOMER_NAME	VARCHAR(16)
CUSTOMER_EMAIL	VARCHAR(16)

PART_TAB

Spaltenname	Datentyp
PART_KEY	INTEGER
COLOR	CHAR(6)
QUANTITY	INTEGER
PRICE	DECIMAL(10,2)

Spaltenname	Datentyp
TAX	REAL
ORDER_KEY	INTEGER

SHIP_TAB

Spaltenname	Datentyp
DATE	DATE
MODE	CHAR(6)
COMMENT	VARCHAR(128)
PART_KEY	INTEGER

Planung

Bevor Sie mit dem XML Extender Ihre Dokumente zusammensetzen, müssen Sie die Struktur des XML-Dokuments festlegen und definieren, wie es der Struktur der Daten in der Datenbank entspricht. Dieser Abschnitt bietet einen Überblick zu der von der Serviceabteilung angeforderten XML-Dokumentstruktur, zu der DTD, mit der Sie das Dokument überprüfen, und zu der Zuordnung zu den Spalten mit den Daten, die die Dokumente füllen.

Dokumentstruktur festlegen

Die XML-Dokumentstruktur ruft Informationen für eine bestimmte Bestellung aus verschiedenen Tabellen ab und erstellt ein XML-Dokument für die Bestellung. Diese Tabellen enthalten zugehörige Informationen zu der Bestellung; sie können entsprechend ihren Schlüsselspalten kombiniert werden. Die Serviceabteilung möchte ein Dokument, das auf der Ausgangsebene nach der Nummer der Bestellung strukturiert ist und anschließend nach Kunden-, Teile- und Versandinformationen. Die Dokumentstruktur soll intuitiv und flexibel sein, wobei die Elemente die Daten beschreiben statt der Struktur des Dokuments. (Der Name des Kunden soll beispielsweise in einem Element mit dem Namen „customer" gespeichert sein statt in einem Abschnitt.) Entsprechend der Anforderung soll die hierarchische Struktur der DTD und des XML-Dokuments wie die in Abb. 5 auf Seite 32 aussehen.

Nachdem Sie die Dokumentstruktur festgelegt haben, müssen Sie eine DTD zur Beschreibung der Struktur des XML-Dokuments erstellen. Dieses Lernprogramm bietet Ihnen ein XML-Dokument und eine DTD. Die DTD-Datei ist in „Anhang B. Beispiele" auf Seite 285 dargestellt. Sie sehen, daß sie der Struktur in Abb. 5 auf Seite 32 entspricht.

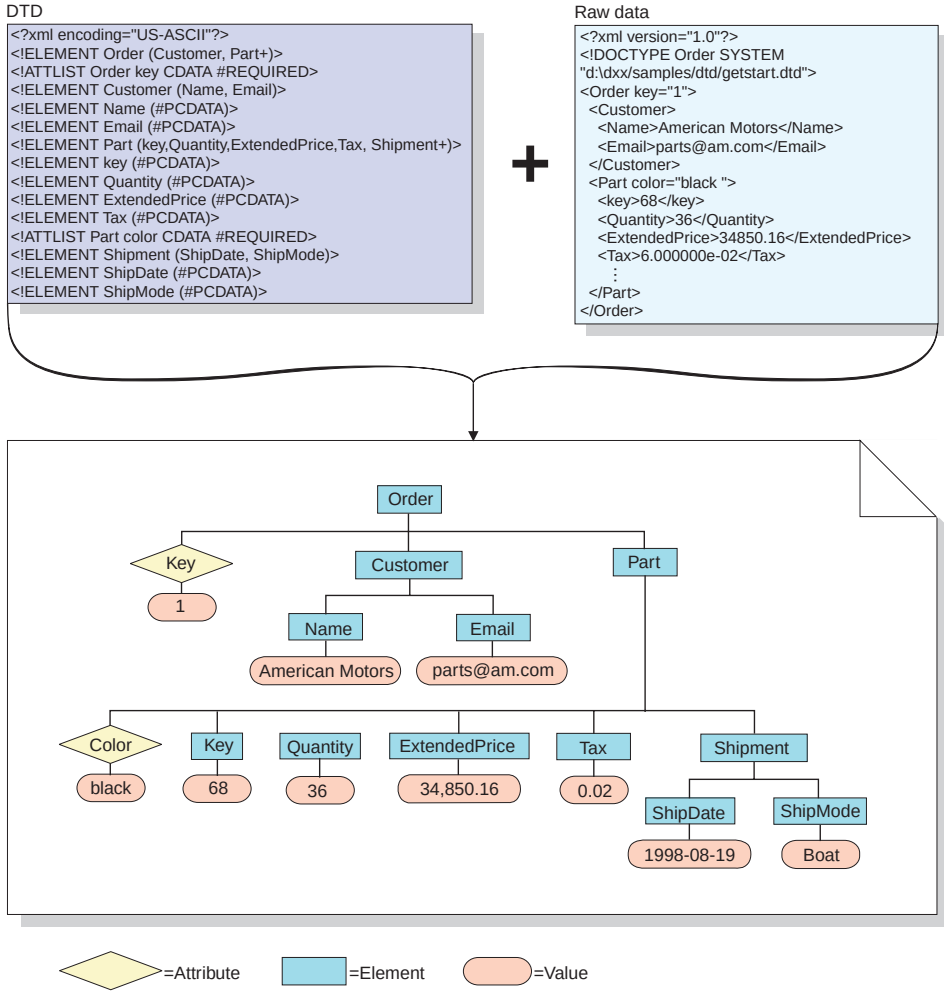


Abbildung 5. Die hierarchische Struktur der DTD und des XML-Dokuments

Beziehung XML-Dokument und Datenbank zuordnen

Nachdem Sie die Struktur festgelegt und die DTD erstellt haben, müssen Sie zeigen, wie die Struktur der Dokumente sich zu den DB2-Tabellen verhält, mit denen Sie die Elemente und Attribute füllen werden. Sie können die hierarchische Struktur spezifischen Spalten in den relationalen Tabellen zuordnen, wie in Abb. 6 auf Seite 33 gezeigt.

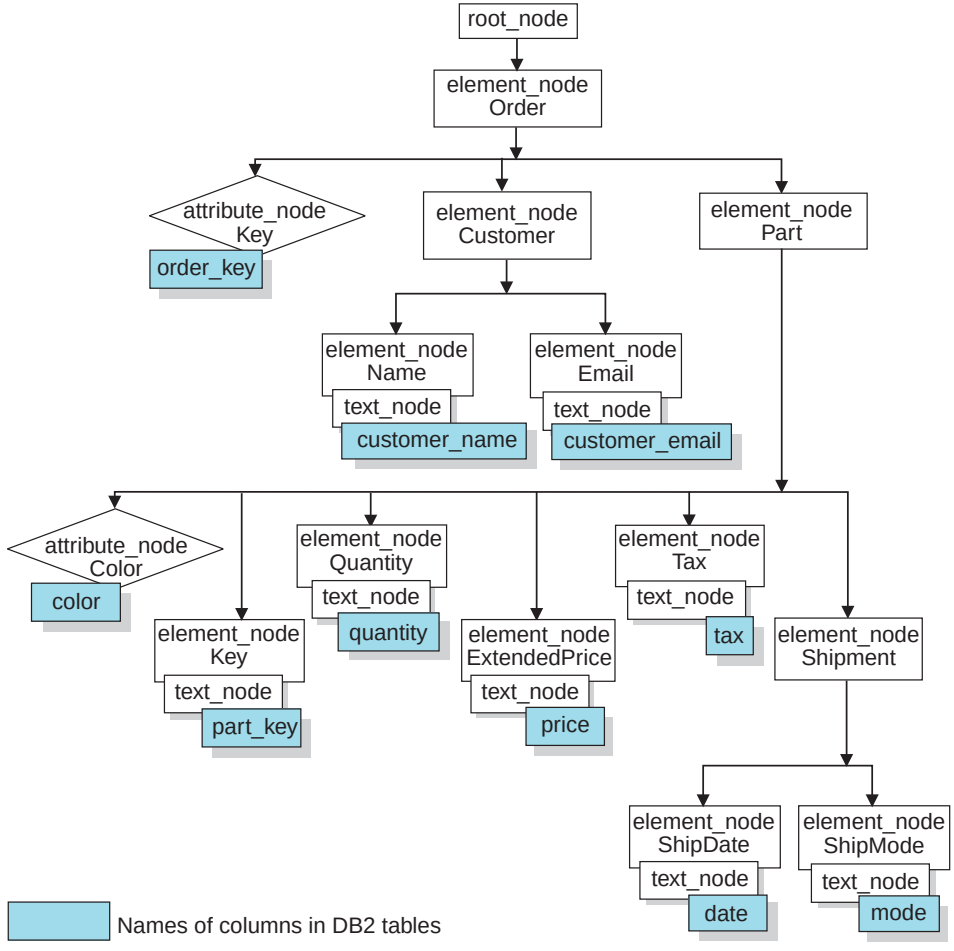


Abbildung 6. Relationalen Tabellenspalten zugeordnetes XML-Dokument

Verwenden Sie diese Beziehungsbeschreibung zum Erstellen von DAD-Dateien, die die Beziehung zwischen den relationalen Daten und der XML-Dokumentstruktur definieren.

Zum Erstellen der DAD-Datei für die XML-Objektgruppe müssen Sie wissen, wie sich das XML-Dokument zu der Datenbankstruktur verhält (siehe Abb. 6), um beschreiben zu können, aus welchen Tabellen und Spalten die XML-Dokumentstruktur Daten für Elemente und Attribute ableitet. Sie verwenden diese Informationen zum Erstellen der DAD-Datei für die XML-Objektgruppe.

Für dieses Lernprogramm wurde eine Reihe von Prozeduren bereitgestellt, mit denen Sie Ihre Umgebung einrichten können. Diese Prozeduren befinden sich im Verzeichnis*DXX_INSTALL*\samples\cmd (*DXX_INSTALL* gibt hierbei das Lauf-

werk und das Verzeichnis an, in dem Sie den XML Extender installiert haben, z. B. `c:\dxx\samples\cmd`); sie heißen wie folgt:

getstart_db.cmd
Erstellt die Datenbank und füllt die vier Tabellen.

getstart_prep.cmd
Bindet die Datenbank an die gespeicherten Prozeduren des XML Extender und die DB2-CLI.

getstart_stp.cmd
Führt die gespeicherte Prozedur zum Zusammensetzen der XML-Objektgruppe aus.

getstart_exportXML.cmd
Exportiert das XML-Dokument aus der Datenbank zur Verwendung in einer Anwendung.

getstart_clean.cmd
Bereinigt die Lernprogrammumgebung.

Einstellen

Datenbank erstellen

In diesem Abschnitt richten Sie mit einem entsprechenden Befehl die Datenbank ein. Dieser Befehl erstellt eine Musterdatenbank, stellt die Verbindung zu dieser Datenbank her, erstellt die Tabellen für die Daten und fügt die Daten anschließend ein.

Wichtig: Wenn Sie die Lektion zu "XML-Spalte" abgeschlossen und Ihre Umgebung noch nicht bereinigt haben, können Sie diesen Schritt eventuell überspringen. Überprüfen Sie, ob Sie eine Datenbank SALES_DB haben.

So erstellen Sie die Datenbank:

1. Wechseln Sie zum Verzeichnis *DXX_INSTALL*`\samples\cmd`, wobei *DXX_INSTALL* das Laufwerk und Verzeichnis angibt, in dem der XML Extender installiert wurde. In diesen Lektionen wird von dem Verzeichnis `c:\dxx` ausgegangen. Passen Sie diese Angaben entsprechend an, wenn Sie ein anderes Laufwerk und Verzeichnis verwenden.
2. Öffnen Sie das DB2-Befehlsfenster über das Windows NT-Startmenü, oder geben Sie den folgenden Befehl über die Windows NT-Eingabeaufforderung ein:
   ```
   DB2CMD
   ```
3. Führen Sie über das DB2-Befehlsfenster aus den folgenden Befehl aus:
   ```
   getstart_db.cmd
   ```

Datenbank aktivieren

Zum Speichern von XML-Informationen in der Datenbank müssen Sie diese für den XML Extender aktivieren. Wenn Sie eine Datenbank für XML aktivieren, führt der XML Extender folgende Schritte aus:

- Erstellt alle benutzerdefinierten Typen (UDTs) und benutzerdefinierten Funktionen (UDFs).
- Erstellt die Steuertabellen und füllt sie mit den erforderlichen Metadaten für den XML Extender
- Erstellt das db2xml-Schema und ordnet die erforderlichen Berechtigungen zu.

Wichtig: Wenn Sie die Lektion zu "XML-Spalte" abgeschlossen und Ihre Umgebung noch nicht bereinigt haben, können Sie diesen Schritt eventuell überspringen.

So aktivieren Sie die Datenbank für XML:

Führen Sie vom DB2-Befehlsfenster aus die folgende Prozedur aus, um die Datenbank SALES_DB zu aktivieren:

```
getstart_prep.cmd
```

Diese Prozedur bindet die Datenbank an die gespeicherten Prozeduren des XML Extender und die DB2-CLI. Außerdem führt diese Prozedur die Befehlsoption **dxxadm** aus, mit der die Datenbank aktiviert wird:

```
dxxadm enable_db SALES_DB
```

XML-Objektgruppe erstellen: DAD-Datei vorbereiten

Da die Daten bereits in verschiedenen Tabellen vorhanden sind, erstellen Sie eine XML-Objektgruppe, die die Tabellen dem XML-Dokument zuordnet. Zum Erstellen einer XML-Objektgruppe definieren Sie die Objektgruppe durch Vorbereiten einer DAD-Datei.

In „Planung" auf Seite 31 haben Sie festgelegt, welche Spalten in der relationalen Datenbank mit den Daten vorhanden sind und wie die Daten aus den Tabellen in einem XML-Dokument strukturiert werden. In diesem Abschnitt erstellen Sie das Zuordnungsschema in der DAD-Datei, die die Beziehung zwischen den Tabellen und der Struktur des XML-Dokuments angibt.

In den folgenden Schritten werden die Elemente in der DAD als *Befehle* und die Elemente Ihrer XML-Dokumentstruktur als *Elemente* bezeichnet. Ein Muster einer DAD-Datei ähnlich der von Ihnen erstellten finden Sie in `c:\dxx\samples\dad\getstart_xcollection.dad`. Dieses Muster weist einige kleine Unterschiede gegenüber der in den folgenden Schritten generierten Datei auf. Wenn Sie das Muster für die Lektion verwenden, beachten Sie, daß die Dateipfade für Ihre Umgebung anders sein können.

So erstellen Sie eine DAD-Datei zum Zusammensetzen eines XML-Dokuments:

1. Öffnen Sie vom Verzeichnis `c:\dxx\samples\cmd` aus einen Texteditor, und erstellen Sie eine Datei mit dem Namen `getstart_xcollection.dad`.
2. Erstellen Sie die DAD-Kopfzeilen mit dem folgenden Text:

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   Der XML Extender geht davon aus, daß Sie das Produkt im Verzeichnis `c:\dxx` installiert haben. Ändern Sie ggf. hier und in den folgenden Schritten diese Angabe ggf. auf das Laufwerk und das Verzeichnis, das Sie bei der Installation dieses Produkts angegeben haben.
3. Fügen Sie die Befehle <DAD></DAD> ein. Alle weiteren Befehle befinden sich innerhalb dieser Befehle.
4. Geben Sie die Befehle <validation> </validation> an, um anzugeben, ob der XML Extender die XML-Dokumentstruktur mit der DTD überprüft, die Sie in der DTD-Repository-Tabelle eingefügt hatten.

   ```
   <validation>NO</validation>
   ```
5. Verwenden Sie die Befehle <Xcollection> </Xcollection> zum Definieren der Zugriffs- und Speichermethode als XML-Objektgruppe. Die Zugriffs- und Speichermethode geben an, daß die XML-Daten in einer Objektgruppe von DB2-Tabellen gespeichert sind.

   ```
   <Xcollection>
   </Xcollection>
   ```
6. Geben Sie eine SQL-Anweisung zur Definition der Tabellen und Spalten für die XML-Objektgruppe an. Diese Methode wird als SQL-Zuordnung bezeichnet; sie stellt eine der beiden Möglichkeiten dar, relationale Daten der XML-Dokumentstruktur zuzuordnen. (Weitere Informationen zu Zuordnungsschemata finden Sie im Abschnitt „Arten von Zuordnungsschemata” auf Seite 64.) Geben Sie die folgende Anweisung ein:

   ```
   <SQL_stmt>
      SELECT o.order_key, customer_name, customer_email, p.part_key, color,
      quantity, price, tax, ship_id, date, mode from order_tab o, part_tab p,
      table (select substr(char(timestamp(generate_unique())),16)
        as ship_id, date, mode, part_key from ship_tab) s
            WHERE o.order_key = 1 and
                  p.price > 20000 and
                  p.order_key = o.order_key and
                  s.part_key = p.part_key
            ORDER BY order_key, part_key, ship_id
   </SQL_stmt>
   ```

   Diese SQL-Anweisung verwendet die folgenden Richtlinien bei Verwendung der SQL-Zuordnung. Schlagen Sie die Dokumentstruktur in Abb. 6 auf Seite 33 nach.

- Spalten werden von oben nach unten angegeben entsprechend der Hierarchie der XML-Dokumentstruktur. Die Spalten für die Bestellung Elemente ″order″ und ″customer″ sind beispielsweise die ersten, das Element ″part″ ist das zweite und ″shipment″ das dritte.
- Die Spalten für eine Entität sind gruppiert, und jede Gruppe hat eine Objekt-ID-Spalte: ORDER_KEY, PART_KEY und SHIP_ID.
- Die Objekt-ID-Spalte ist in jeder Gruppe die erste Spalte. O.ORDER-_KEY steht beispielsweise vor den Spalten zum Schlüsselattribut, und p.PART_KEY steht vor dem Element ″Part″.
- Die Tabelle SHIP_TAB hat keine einzelne Schlüsselbedingungsspalte; daher wird die integrierte DB2-Funktion generate_unique zum Generieren der Spalte SHIP_ID verwendet.
- Die Objekt-ID-Spalten werden anschließend von oben nach unten in Anweisungen ORDER BY aufgelistet. Die Spalten in ORDER BY sollten nicht über ein Schema und einen Tabellennamen qualifiziert werden und sollten den Spaltennamen in der SELECT-Klausel entsprechen.

Die Anforderungen zum Schreiben einer SQL-Anweisung sind im Abschnitt „Voraussetzung für das Zuordnungsschema” auf Seite 66 beschrieben.

7. Fügen Sie die folgenden Prologinformationen zur Verwendung in dem zusammengesetzten XML-Dokument hinzu.

   ```
   <prolog>?xml version="1.0"?</prolog>
   ```

   Dieser Text ist in genau dieser Form für alle DAD-Dateien erforderlich.
8. Fügen Sie die Befehle <doctype></doctype> zur Verwendung in dem zusammengesetzten XML-Dokument ein. Der Befehl <doctype> enthält den Pfad zu der auf dem Client gespeicherten DTD.

   ```
   <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
   ```
9. Definieren Sie das Root-Element des XML-Dokuments mit den Befehlen <root_node></root_node>. Innerhalb des Root-Knotens geben Sie die Elemente und Attribute an, aus denen das XML-Dokument besteht.
10. Ordnen Sie mit den folgenden drei Arten von Knoten die XML-Dokumentstruktur der Struktur der relationalen DB2-Tabelle zu:

    element_node
    (Elementknoten) Gibt das Element in dem XML-Dokument an. Elementknoten können untergeordnete Elementknoten enthalten.

    attribute_node
    (Attributknoten) Gibt das Attribut eines Elements in dem XML-Dokument an.

text_node
(Textknoten) Gibt den Textinhalt des Elements und die Spaltendaten in einer relationalen Tabelle für die Elementknoten der untersten Ebene an.

Weitere Informationen zu diesen Knoten finden Sie im Abschnitt „Die DAD-Datei" auf Seite 61. Abb. 6 auf Seite 33 zeigt die hierarchische Struktur des XML-Dokuments und der DB2-Tabellenspalten und gibt an, welche Arten von Knoten verwendet werden. Die schraffierten Kästchen kennzeichnen die Namen der DB2-Tabellenspalten, as denen die Daten beim Zusammensetzen des XML-Dokuments extrahiert werden.

Mit den folgenden Schritten fügen Sie nacheinander die einzelnen Knotenarten hinzu.

a. Definieren Sie einen Befehl <element_node> für jedes Element in dem XML-Dokument.

```
<root_node>
<element_node name="Order">
  <element_node name="Customer">
     <element_node name="Name">
     </element_node>
     <element_node name="Email">
     </element_node>
  </element_node>
  <element_node name="Part">
    <element_node name="key">
     </element_node>
    <element_node name="Quantity">
      </element_node>
    <element_node name="ExtendedPrice">
      </element_node>
    <element_node name="Tax">
      </element_node>
     <element_node name="Shipment" multi_occurrence="YES">
      <element_node name="ShipDate">
      </element_node>
      <element_node name="ShipMode">
      </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

Beachten Sie, daß das untergeordnete Element <Shipment> das Attribut multi_occurrence="YES" aufweist. Dieses Attribut wird für Elemente ohne Attribute verwendet, die in dem Dokument wiederholt werden. Das Element <Part> verwendet das Attribut "multi-occurrence" nicht, da es durch das Attribut "Color" eindeutig ist.

b. Definieren Sie einen Befehl <attribute_node> für jedes Attribut in Ihrem XML-Dokument. Diese Attribute sind innerhalb ihres element-_node verschachtelt. Die hinzugefügten attribute_nodes sind in Fettschrift hervorgehoben:

```
<root_node>
<element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
  <element_node name="Customer">
     <element_node name="Name">
     </element_node>
     <element_node names"Email">
     </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
    <element_node name="key">
    </element_node>
    <element_node name="Quantity">
    </element_node>

...

  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

c. Definieren Sie für jeden element_node auf der untersten Ebene die Befehle <text_node>, um zu kennzeichnen, daß das XML-Element Zeichendaten enthält, die beim Zusammensetzen des Dokuments aus DB2 extrahiert werden können.

```
<root_node>
<element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node>
      </text_node>
    </element_node>
    <element_node name="Email">
      <text_node>
      </text_node>
    </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
    <element_node name="key">
       <text_node>
       </text_node>
    </element_node>
    <element_node name="Quantity">
```

```
      <text_node>
      </text_node>
    </element_node>
    <element_node name="ExtendedPrice">
      <text_node>
      </text_node>
    </element_node>
    <element_node name="Tax">
      <text_node>
      </text_node>
    </element_node>
    <element_node name="Shipment" multi-occurrence="YES">
      <element_node name="ShipDate">
        <text_node>
        </text_node>
      </element_node>
      <element_node name="ShipMode">
        <text_node>
        </text_node>
      </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

d. Definieren Sie für jeden element_node auf der untersten Ebene einen Befehl <column>. Diese Befehle geben an, aus welcher Spalte beim Zusammensetzen des XML-Dokuments die Daten extrahiert werden sollen; sie befinden sich normalerweise innerhalb der Befehle <attribute_node> oder <text_node>. Denken Sie daran, daß sich die hier definierten Spalten innerhalb der Klausel <SQL_stmt> SELECT befinden müssen.

```
<root_node>
<element_node name="Order">
  <attribute_node name="key">
    <column name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node>
        <column name="customer_name"/>
      </text_node>
    </element_node>
    <element_node name="Email">
      <text_node>
        <column name="customer_email"/>
     </text_node>
    </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
       <column name="color"/>
    </attribute_node>
    <element_node name="key">
```

```
      <text_node>
        <column name="part_key"/>
      </text_node>
    <element_node name="Quantity">
      <text_node>
        <column name="quantity"/>
      </text_node>
    </element_node>
    <element_node name="ExtendedPrice">
      <text_node>
        <column name="price"/>
      </text_node>
    </element_node>
    <element_node name="Tax">
      <text_node>
        <column name="tax"/>
      </text_node>
    </element_node>
    <element_node name="Shipment" multi-occurrence="YES">
      <element_node name="ShipDate">
        <text_node>
          <column name="date"/>
        </text_node>
      </element_node>
      <element_node name="ShipMode">
        <text_node>
          <column name="mode"/>
        </text_node>
      </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

11. Stellen Sie sicher, daß Sie einen Endbefehl </root_node> nach dem letzten Befehl </element_node> verwenden.
12. Stellen Sie sicher, daß Sie einen Endbefehl </Xcollection> nach dem Befehl </root_node> verwenden.
13. Stellen Sie sicher, daß Sie einen Endbefehl </DAD> nach dem Befehl </Xcollection> verwenden.
14. Speichern Sie die Datei unter dem Namen `getstart_xcollection.dad`.

Sie können die eben erstellte Datei mit der Beispieldatei `c:\dxx\samples\dad\getstart_xcollection.dad` vergleichen. Diese Datei ist eine Arbeitskopie der zum Zusammensetzen des XML-Dokuments erforderlichen DAD-Datei. Die Beispieldatei enthält Pfadanweisungen, die für eine erfolgreiche Ausführung eventuell an Ihre Umgebung angepaßt werden müssen. Wenn Sie in Ihrer Anwendung eine XML-Objektgruppe häufig zum Zusammensetzen von Dokumenten verwenden, können Sie durch Aktivieren der Objektgruppe einen Gruppennamen definieren. Durch Aktivieren der Objektgruppe wird diese in der Tabelle XML_USAGE registriert; durch die

Angabe des Gruppennamens (statt des Namens der DAD-Datei) bei der Ausführung gespeicherter Prozeduren kann somit die Leistung verbessert werden. In diesen Lektionen aktivieren Sie die Objektgruppe nicht. Weitere Informationen zum Aktivieren von Objektgruppen finden Sie im Abschnitt „XML-Objektgruppen aktivieren" auf Seite 120.

XML-Dokument zusammensetzen

In diesen Schritt verwenden Sie die gespeicherte Prozedur dxxGenXML() zum Zusammensetzen des über die DAD-Datei angegebenen XML-Dokuments. Diese gespeicherte Prozedur gibt das Dokument als einen UDT XMLVARCHAR zurück.

So stellen Sie das XML-Dokument zusammen:

1. Geben Sie über das DB2-Befehlsfenster den folgenden Befehl zur Ausführung der gespeicherten Prozedur ein:

   ```
   getstart_stp.cmd
   ```

   Das XML-Dokument wurde zusammengesetzt und wird in der Tabelle RESULT_TAB gespeichert.

   Beispiele zu den gespeicherten Prozeduren, die in diesem Schritt verwendet werden können, finden Sie in den folgenden Dateien:
   - `c:\dxx\samples\c\tests2x.sqc` zeigt, wie die gespeicherte Prozedur über das eingebettete SQL aufgerufen und die ausführbare Datei `texts2x` generiert wird, die von `getstart_stp.cmd` verwendet wird.
   - `c:\dxx\samples\cli\sql2xml.c` zeigt, wie die gespeicherte Prozedur über das CLI aufgerufen wird.
2. Exportieren Sie das XML-Dokument aus der Tabelle in eine Datei; verwenden Sie hierzu die XML Extender-Abruffunktion Content():

   ```
   DB2 CONNECT TO SALES_DB

   DB2 SELECT db2xml.Content(db2xml.xmlvarchar(doc),
      'c:\dxx\samples\cmd\getstart.xml') FROM RESULT_TAB
   ```

   Alternativ dazu können Sie auch die folgende Befehlsdatei zum Exportieren der Datei ausführen:

   ```
   getstart_exportXML.cmd
   ```

   In dieser Lektion erfahren Sie, wie Sie ein oder mehrere zusammengesetzte XML-Dokumente über die Ergebnisgruppenfunktion der gespeicherten Prozeduren von DB2 abrufen, so daß Sie jede Zeile und damit jedes Dokument abrufen können. Wenn Sie jede Zeile des Dokuments abrufen, können Sie sie in eine Datei exportieren; dies ist die einfachste Möglichkeit, diese Funktion zu demonstrieren. Effektivere Möglichkeiten zum Abrufen von Daten finden Sie in den CLI-Beispielen in `c:\dxx\samples\cli`.

Lernprogrammumgebung bereinigen

Zum Bereinigen der Lernprogrammumgebung können Sie die Datei `getstart_clean.cmd` ausführen. Diese Datei führt folgende Aktionen aus:

- Inaktivieren der XML-Spalte ORDER
- Freigeben der in dem Lernprogramm erstellten Tabellen
- Löschen der DTD aus der DTD-Referenztabelle

Diese Befehlsdatei führt kein Inaktivieren oder Freigeben der Datenbank SALES_DB durch; die Datenbank ist weiterhin zur Verwendung mit XML Extender verfügbar. Eventuell werden Fehlernachrichten angezeigt, wenn Sie nicht beide Lektionen dieses Kapitels ausgeführt haben. Sie können diese Nachrichten ignorieren.

So bereinigen Sie die Lernprogrammumgebung:

1. Führen Sie über das DB2-Befehlsfenster aus den folgenden Befehl aus:

   ```
   getstart_clean.cmd
   ```

2. Wenn Sie die Datenbank inaktivieren wollen, können Sie den folgenden XML Extender-Befehl über das DB2-Befehlsfenster ausführen:

   ```
   dxxadm disable_db SALES_DB
   ```

   Dieser Befehl gibt die Verwaltungssteuertabellen DTD_REF und XML_USAGE frei und entfernt die benutzerdefinierten Typen und Funktionen, die vom XML Extender bereitgestellt wurden.

3. Wenn Sie die Datenbank freigeben wollen, können Sie den folgenden Befehl über das DB2-Befehlsfenster ausführen:

   ```
   db2
   drop database SALES_DB
   ```

   Dieser Befehl gibt SALES_DB frei.

Teil 2. Verwaltung

Dieser Teil des Handbuchs beschreibt die Ausführung von Verwaltungs-Tasks für den XML Extender.

Kapitel 3. XML Extender vorbereiten: Verwaltung

In diesem Kapitel werden die Anforderungen zur Einstellung und Planung des XML Extender beschrieben.

Voraussetzungen zum Einstellen

In den folgenden Abschnitten werden die Voraussetzungen für die Einstellung des XML Extender beschrieben.

Softwarevoraussetzungen

Der XML Extender ist unter AIX, Windows NT und Sun Solaris verfügbar.

Erforderliche Software: Der XML Extender erfordert DB2 Universal Database Version 7.1 oder eine höhere Version.

Wahlweise installierbare Software:

- Für eine strukturelle Textsuche der DB2 Universal Database Text Extender Version 7.1 oder höher
- Für das Verwaltungs-Tool des XML Extender:
  - DB2 UDB JDBC (verfügbar mit DB2 UDB Version 7.1 oder höher)
  - JDK 1.1.7 oder JRE 1.1.7 (verfügbar mit der DB2 UDB-Steuerzentrale)
  - JFC 1.1 mit Swing 1.1 (verfügbar mit mit der DB2 UDB-Steuerzentrale)

Installationsvoraussetzungen

In der Informationsdatei (README) für Ihr Betriebssystem finden Sie Informationen zu folgenden Tasks:

- Binden des XML Extender an Ihre DB2 UDB-Datenbank.

  Aus Gründen der Sicherheit müssen Sie den XML Extender an jede Datenbank binden. Ausführliche Informationen zur Ausführung dieses Vorgangs finden Sie im Abschnitt „Vorbereitungen" auf Seite 223; ein Beispiel hierzu finden Sie in der Datei

  DXX_INSTALL\samples\cmd\getstart_prep.cmd
- Anzeigen der Anweisungen zum Einstellen unter UNIX.
- Erstellen einer Datenbank für den XML-Zugriff.

Voraussetzungen für die Berechtigung

Zur Ausführung von Verwaltungs-Tasks müssen Sie die DB2ADM-Berechtigung haben.

Verwaltungs-Tools

Der XML Extender bietet drei Methoden für die Verwaltung: den XML Extender-Verwaltungsassistenten, den XML Extender-Verwaltungsbefehl und die *gespeicherten Prozeduren* des XML Extender.

- Der Verwaltungsassistent zeigt Eingabeaufforderungen für die Verwaltungs-Tasks an; dieses Programm stellt die empfohlene Verwaltungsmethode dar. Die Verwendung dieses Tools für die Verwaltungs-Tasks wird in „Kapitel 4. XML-Daten verwalten" auf Seite 73 beschrieben.
- Der Verwaltungsbefehl **dxxadm** enthält Optionen für die verschiedenen Verwaltungs-Tasks. Die Verwendung dieses Befehls für die Verwaltungs-Tasks ist in „Kapitel 4. XML-Daten verwalten" auf Seite 73 und in „Kapitel 7. XML Extender-Verwaltungsbefehl: **dxxadm**" auf Seite 169 beschrieben.
- Die gespeicherten Prozeduren für die Verwaltung enthalten ebenfalls Optionen für verschiedene Verwaltungs-Tasks. Diese gespeicherten Prozeduren sind im Abschnitt „Gespeicherte Prozeduren zur Verwaltung" auf Seite 223 beschrieben.

Planung der Verwaltung

Beim Planen einer Anwendung, die XML-Dokumente verwendet, müssen Sie zunächst folgende Entscheidungen zum Design treffen:

- Ob Sie XML-Dokumente aus Daten in der Datenbank zusammensetzen wollen
- Ob Sie vorhandene XML-Dokumente speichern wollen und ob diese Dokumente als intakte XML-Dokumente in einer Spalte oder zerlegt als reguläre DB2-Daten gespeichert werden sollen

Nachdem Sie diese Entscheidungen getroffen haben, können Sie Ihre weiteren Verwaltungs-Tasks planen:

- Ob Sie Ihre XML-Dokumente überprüfen wollen
- Ob Sie die XML-Spaltendaten für die Schnellsuche und zum Abrufen indexieren wollen
- Wie die Struktur des XML-Dokuments den relationalen DB2-Tabellen zugeordnet werden soll

Die Verwendung des XML Extender hängt von den Anforderungen der jeweiligen Anwendung ab. Wie in „Kapitel 1. Einführung in den XML Extender" auf Seite 3 beschrieben, können Sie XML-Dokumente aus vorhandenen DB2-Daten zusammensetzen und XML-Dokumente in DB2 speichern, entweder als intakte Dokumente oder als DB2-Daten. Für jede dieser Speicher- und Zugriffsmethoden gelten andere Planungsvoraussetzungen. In den folgenden Abschnitten werden diese Überlegungen zur Planung beschrieben.

Zugriffs- und Speichermethode auswählen

Der XML Extender bietet zwei Zugriffs- und Speichermethoden zur Verwendung von DB2 als XML-Repository: XML-Spalte und XML-Objektgruppe. Zunächst müssen Sie entscheiden, welche dieser Methoden den Anforderungen Ihrer Anwendung beim Aufrufen und Bearbeiten von XML-Daten am besten gerecht wird.

XML-Spalte
XML-Dokumente werden als DB2-Spaltendaten gespeichert und abgerufen. Die XML-Daten werden als XM-Spalte dargestellt.

XML-Objektgruppe
Zerlegt XML-Dokumente in eine Objektgruppe relationaler Tabellen oder setzt XML-Dokumente aus einer Objektgruppe relationaler Tabellen zusammen.

Von den Merkmalen Ihrer Anwendung hängt es ab, welche Zugriffs- und Speichermethode am besten geeignet ist und wie die XML-Daten strukturiert werden sollten. Die folgenden Szenarien beschreiben Situationen für die verschiedenen Zugriffs- und Speichermethoden.

Situationen für die Verwendung von XML-Spalten

Verwenden Sie XML-Spalten in folgenden Situationen:

- Die XML-Dokumente sind bereits vorhanden oder stammen aus externen Datenquellen, und Sie wollen die Dokumente im nativen XML-Format speichern. Sie wollen aus Gründen der Datenintegrität sowie zu Archivierungs- und Prüfzwecken die Daten in DB2 speichern.
- Die XML-Dokumente werden allgemein gelesen, aber nicht aktualisiert.
- Sie wollen Dateiname-Datentypen verwenden, um XML-Dokumente extern in DB2 in dem lokalen oder fernen Dateisystem zu speichern und um DB2 für Verwaltungs- und Suchoperationen verwenden zu können.
- Sie brauchen eine Bereichssuche entsprechend den Werten der XML-Elemente oder -Attribute, und Sie wissen, welche Elemente bzw. Attribute häufig als Suchargumente verwendet werden.
- Die Dokumente enthalten Elemente mit großen Textblöcken, und Sie wollen den DB2 Text Extender für eine strukturelle Textsuche verwenden und dabei die gesamten Dokumente intakt lassen.

Situationen für die Verwendung von XML-Objektgruppen

Verwenden Sie XML-Objektgruppen in folgenden Situationen:

- Sie haben Daten in Ihren vorhandenen relationalen Tabellen und wollen entsprechend einer bestimmten DTD XML-Dokumente zusammensetzen.
- Sie haben XML-Dokumente, die mit Objektgruppe von Daten gespeichert werden müssen, die sich gut auf relationale Tabellen abbilden lassen.
- Sie wollen verschiedene Sichten Ihrer relationalen Daten mit verschiedenen Zuordnungsschemata erstellen.
- Sie haben XML-Dokumente, die aus anderen Datenquellen stammen. Sie legen Wert auf die Daten, jedoch nicht auf die Befehle, und wollen die reinen Daten in der Datenbank speichern. Sie wollen die Flexibilität haben, selbst zu entscheiden, ob die Daten in vorhandenen oder in neuen Tabellen gespeichert werden sollen.
- Eine kleine Untergruppe Ihrer XML-Dokumente muß häufig aktualisiert werden, und die Leistung bei der Aktualisierung ist von großer Bedeutung.
- Sie müssen die Daten aus vollständigen ankommenden XML-Dokumenten speichern, wollen häufig jedoch nur eine Untermenge davon abrufen.
- Ihre XML-Dokumente sind größer als 2 Gigabyte, und Sie müssen sie zerlegen.

Sie verwenden die Dokumentzugriffsdefinition (DAD) zum Zuordnen von XML-Daten zu DB2-Tabellen über diese beiden Zugriffs- und Speichermethoden. Abb. 7 auf Seite 51 zeigt, wie die DAD die Zugriffs- und Speichermethode angibt.

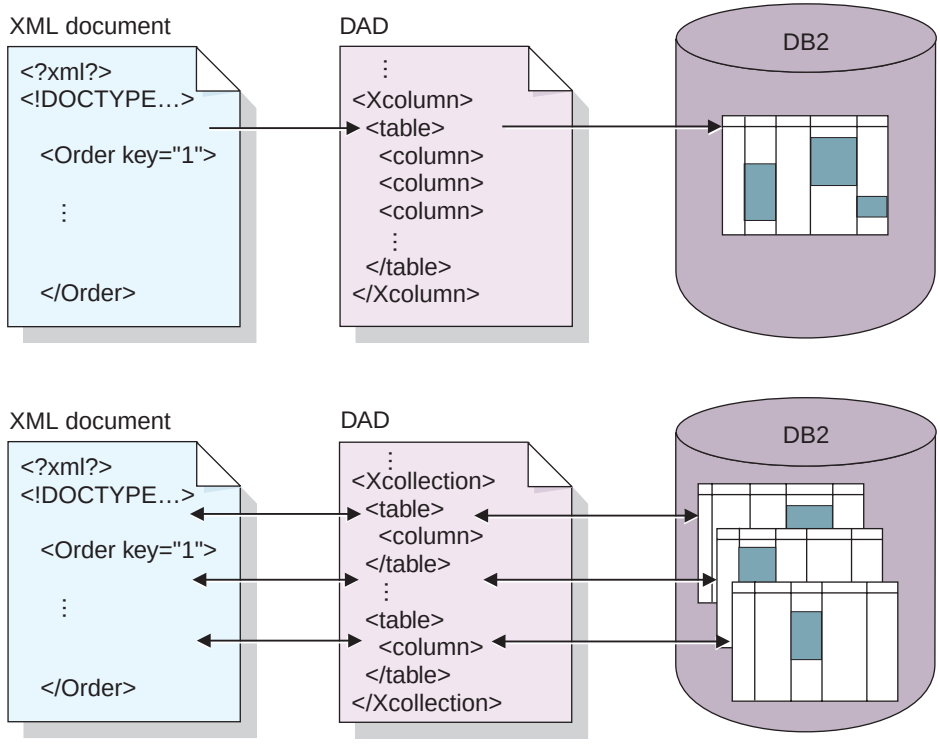


Abbildung 7. Die DAD-Datei ordnet die XML-Dokumentstruktur DB2 zu und legt die Zugriffs- und Speichermethode fest.

Die DAD-Datei ist eine wichtige Komponente bei der Verwaltung des XML Extender. Sie definiert den Standort der Schlüsseldateien wie der DTD und gibt die Beziehung zwischen der XML-Dokumentstruktur und den DB2-Daten an. Vor allem aber definiert sie die in der Anwendung verwendeten Zugriffs- und Speichermethoden.

XML-Spalten planen

In den folgenden Abschnitten werden die Planungsaufgaben für XML-Spalten beschrieben.

Gültigkeitsprüfung

Nachdem Sie eine Zugriffs- und Speichermethode ausgewählt haben, können Sie festlegen, ob Ihre Daten *geprüft* werden sollen. Sie prüfen XML-Daten durch Verwendung einer DTD, um sicherzustellen, daß das XML-Dokument gültig ist und zur Ausführung einer strukturellen Suche in Ihren XML-Daten. Die DTD wird im DTD-Repository gespeichert, oder sie kann in dem Dateisystem gespeichert werden, auf das der DB2-Server Zugriff hat.

Sie können Dokumente in derselben XML-Spalte mit verschiedenen DTDs prüfen. Mit anderen Worten: Sie können Dokumente haben, die eine ähnliche Struktur aufweisen und ähnliche Elemente und Attribute enthalten und die doch unterschiedliche DTDs aufrufen. Halten Sie sich beim Verweisen auf verschiedene DTDs an die folgenden Regeln:

- Die System-ID des XML-Dokuments in der DOCTYPE-Definition muß die DTD-Datei mit einem vollständigen Pfadnamen angeben.
- Sie müssen für die Überprüfung in der DAD-Datei YES angeben.
- Mindestens eine der DTDs muß in der Tabelle DTD_REF gespeichert werden. Alle DTDs können in dieser Tabelle gespeichert werden.
- Die DTDs müssen eine gemeinsame Struktur aufweisen, die sich nur in den Unterelementen unterscheidet.
- Die DAD-Datei darf nur Elemente oder Attribute angeben, die allen DTDs gemeinsam sind, auf die von Dokumenten in dieser Spalte verwiesen wird.

Wichtig: Entscheiden Sie vor dem Einfügen von XML-Daten in DB2, ob die Daten geprüft werden sollen. Der XML Extender unterstützt die Überprüfung von Daten nicht, die bereits in DB2 eingefügt wurden.

Überlegungen:

- Es wird empfohlen, die XML-Daten mit einer DTD zu prüfen, es sei denn, Sie speichern XML-Dokumente zu Archivierungszwecken. Zur Prüfung müssen Sie eine DTD im XML Extender-Repository haben. Im Abschnitt „Eine DTD im DTD-Repository speichern" auf Seite 78 finden Sie Informationen dazu, wie Sie eine DTD in das Repository einfügen können.
- Sie brauchen keine DTD zum Speichern oder Archivieren von XML-Dokumenten.
- Die Überprüfung Ihrer XML-Daten kann eine gewisse Auswirkung auf den Durchsatz haben.
- Sie können mehrere DTDs verwenden, jedoch nur gemeinsame Elemente und Attribute indexieren.
- Wenn Sie ein Dokument prüfen, wird die in dem XML-Dokument angegebene DTD nicht verarbeitet. Es ist wichtig, daß DTDs verarbeitet werden, um Entitäten und Attributwerte aufzulösen; dies gilt auch bei der Verarbeitung von Dokumentteilen, die nicht geprüft werden können.

Benutzerdefinierte XML-Typen

Sie speichern ein XML-Dokument in einer XML-Spalte als UDT. Eine Übersicht über die verfügbaren UDTs finden Sie in Tabelle 4.

Tabelle 4. Die XML Extender-UDTs

Spalte benutzerdefinierter Typ	Quellendatentyp	Beschreibung der Verwendung
XMLVARCHAR	VARCHAR(*varchar_len*)	Speichert ein vollständiges XML-Dokument als VARCHAR in DB2.
XMLCLOB	CLOB(*clob_len*)	Speichert ein vollständiges XML-Dokument als *CLOB* in DB2.
XMLFILE	VARCHAR(1024)	Speichert den Dateinamen eines XML-Dokuments DB2 und speichert das XML-Dokument in einer lokalen Datei auf dem DB2-Server.

Seitentabellen

Bei der Planung von Seitentabellen müssen Sie berücksichtigen, wie diese Tabellen organisiert sind, wie viele Tabellen erstellt werden sollen und ob eine Standardsicht für die Seitentabellen erstellt werden soll. Diese Entscheidungen basieren auf verschiedenen Punkten: ob die Elemente und Attribute mehrfach vorkommen können sowie auf den Anforderungen zur Leistung der Abfrage.

Mehrfaches Vorkommen: Wenn ein Dokument mehrere auftretende Standortpfade ausweist, fügt XML Extender eine Spalte DXX_SEQNO des Typs INTEGER in jeder Seitentabelle hinzu, um die Reihenfolge der mehrfach auftretenden Elemente aufzuzeichnen. Mit DXX_SEQNO können Sie eine Liste der Elemente abrufen mit der gleichen Reihenfolge wie im ursprünglichen XML-Dokument; geben Sie hierzu ORDER BY DXX_SEQNO in einer SQL-Abfrage an.

Standardsichten und Abfrageleistung: Wenn Sie eine XML-Spalte aktivieren, können Sie eine Standardsicht mit Lesezugriff angeben, die die Anwendungstabelle mit den Seitentabellen über eine eindeutige Kennung, die ROOT ID, verknüpft. Mit der Standardsicht können Sie XML-Dokumente durch Abfrage der Seitentabellen durchsuchen. Beispiel: Sie haben die Anwendungstabelle SALES_TAB und die Seitentabellen ORDER_TAB, PART_TAB und SHIP_TAB:

```
SELECT sales_person FROM sales_order_view
WHERE price > 2500.00
```

Die SQL-Anweisung gibt den Namen der Vertriebsmitarbeiter (sales_person) in SALES_TAB zurück, die Bestellungen in Spalte ORDER gespeichert haben, für die ein PRICE über 2500.00 eingetragen ist.

Der Vorteil der Abfrage der Standardsicht liegt darin, daß sie eine virtuelle einzelne Sicht der Anwendungstabelle und der Seitentabellen darstellt. Je mehr Seitentabellen erstellt werden, desto aufwendiger ist allerdings die Abfrage. Das Erstellen der Standardsicht wird daher nur empfohlen, wenn die Gesamtzahl der Seitentabellen klein ist. Anwendungen können ihre eigenen Sichten erstellen und dabei die wichtigen Spalten der Seitentabellen verknüpfen.

Indizes für XML-Spaltendaten

Eine wichtige Entscheidung bei der Planung ist, ob Ihr XML-Spaltendokument indexiert werden soll. Diese Entscheidung hängt davon ab, wie häufig auf die Daten zugegriffen werden soll und wie wichtig der Durchsatz bei einer strukturellen Suche ist.

Beim Arbeiten mit XML-Spalten, die vollständige XML-Dokumente enthalten, können Sie Seitentabellen erstellen, die Spalten von XML-Element- oder -Attributwerten enthalten. Anschließend können Sie Indizes zu diesen Spalten erstellen. Sie müssen ermitteln, für welche Elemente und Attribute der Index erstellt werden soll.

Die XML-Spaltenindexierung ermöglicht ein Indexieren von häufig abgefragten Daten mit allgemeinen Datentypen wie Integer, Decimal oder Date, über die native DB2-Indexunterstützung der Datenbanksteuerkomponente. Der XML Extender extrahiert die Werte der XML-Elemente oder Attribute aus XML-Dokumenten und speichert sie in den *Seitentabellen*, so daß Sie Indizes zu diesen Seitentabellen erstellen können.

Sie können jede Spalte einer Seitentabelle mit einem Standortpfad, der ein XML-Element oder -Attribut und einen SQL-Datentyp kennzeichnet, angeben. Abb. 8 auf Seite 55 zeigt eine XML-Spalte mit Seitentabellen.

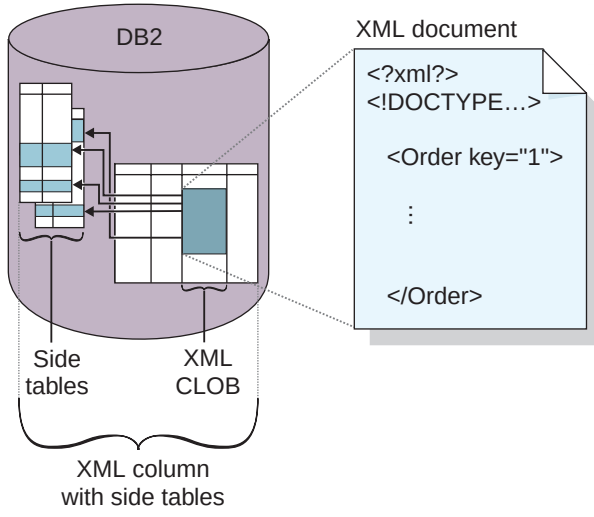


Abbildung 8. Eine XML-Spalte mit Seitentabellen

Der XML Extender füllt die Seitentabelle automatisch, wenn Sie XML-Dokumente in der XML-Spalte speichern.

Erstellen Sie für eine Schnellsuche Indizes dieser Spalten mit der Technologie der DB2 *B-Baumstruktur-Indexierung*. Die zum Erstellen eines Index verwendeten Methoden unterscheiden sich je nach dem verwendeten Betriebssystem, und der XML Extender unterstützt diese Methoden.

Überlegungen:

- Wenn in einem XML-Dokument Elemente oder Attribute *mehrfach vorkommen*, müssen Sie wegen der komplexen Struktur der XML-Dokumente für jedes XML-Element bzw. -Attribut eine separate Seitentabelle erstellen.

  Sie können beispielsweise einen Index über `/Order/Part/ExtendedPrice` erstellen und `/Order/Part/ExtendedPrice` als Datentyp REAL angeben. In diesem Fall speichert der XML Extender den Wert von `/Order/Part/ExtendedPrice` in der Spalte PRICE in einer Seitentabelle.

- Sie können mehrere Indizes zu einer XML-Spalte erstellen. Entsprechend dem vorherigen Beispiel können Sie zwei Spalten in zwei Seitentabellen erstellen, eine für `ExtendedPrice` und eine für `ShipDate`.
- Sie können Seitentabellen über die ROOT ID zuordnen. ROOT ID ist der Spaltenname des Primärschlüssels in der Anwendungstabelle und eine eindeutige Kennung, die alle Seitentabellen der Anwendungstabelle zuordnet. Sie können angeben, daß der Primärschlüssel der Anwendungstabelle die ROOT ID sein soll; diese ID kann jedoch nicht als zusammengesetzter Schlüssel verwendet werden. Diese Methode wird empfohlen.

  Wenn der einzelne Primärschlüssel in der Anwendungstabelle nicht vorhanden ist oder Sie ihn aus irgendeinem Grund nicht verwenden wollen, ändert der XML Extender die Anwendungstabelle und fügt eine Spalte DXXROOT_ID hinzu, in der eine eindeutige ID gespeichert wird, die beim Einfügen erstellt wird. Alle Seitentabellen enthalten eine Spalte DXXROOT_ID mit der eindeutigen ID. Wenn der Primärschlüssel als ROOT ID verwendet wird, haben alle Seitentabellen eine Spalte mit dem gleichen Namen und Typ wie der Primärschlüssel in der Anwendungstabelle, und die Werte des Primärschlüssels werden gespeichert.
- Wenn Sie eine XML-Spalte für den DB2 Text Extender aktivieren, können Sie auch die Funktion für strukturellen Text des Text Extender verwenden. Der Text Extender unterstützt die *Abschnittsuche* ("section search"), die die Möglichkeit einer konventionellen Volltextsuche erweitert und den Vergleich von Suchbegriffen in einem spezifischen Dokumentkontext, der über Standortpfade angegeben wird, ermöglicht. Der *Index für den strukturellen Text* kann mit der Indexierung des XML Extender bei allgemeinen SQL-Datentypen verwendet werden.

Standortpfad

Ein *Standortpfad* ist eine Folge von *XML-Befehlen*, die ein XML-Element oder -attribut kennzeichnen. Der XML Extender verwendet den Standortpfad in den folgenden Situationen:

- Beim Extrahieren der UDFs zur Identifikation der zu extrahierenden Elemente und Attribute
- Zum Angeben der Zuordnungsdatei zwischen einem XML-Element bzw. -attribut und einer DB2-Spalte beim Definieren des Indexierungsschemas in der DAD für XML-Spalten
- Vom Text Extender für eine strukturierte Textsuche

Abb. 9 auf Seite 57 zeigt ein Beispiel eines Standortpfads und seiner Beziehung zur Struktur des XML-Dokuments.

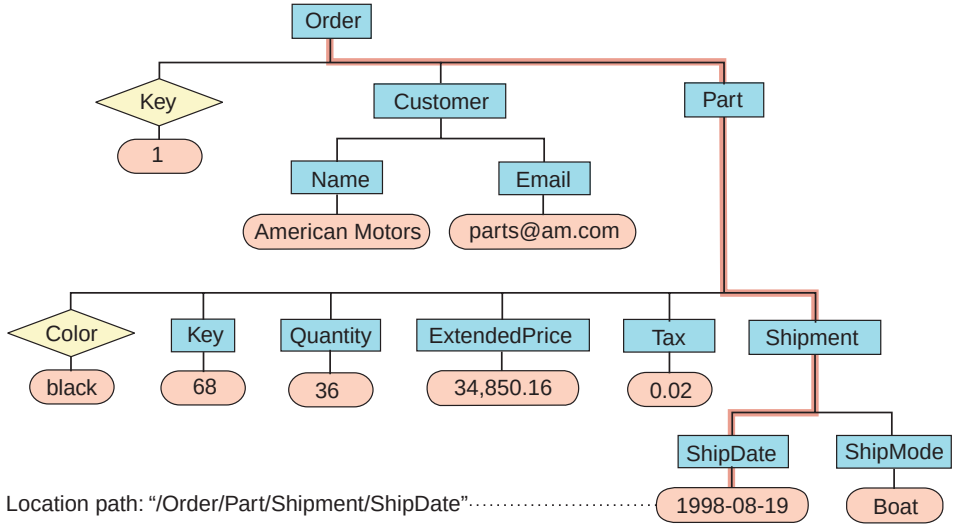


Abbildung 9. Speichern von Dokumenten als strukturierte XML-Dokumente in einer DB2-Tabellenspalte

Syntax des Standortpfads: Die folgende Liste beschreibt die Syntax des vom XML Extender unterstützten Standortpfads. Ein einfacher Schrägstrich (/) als Pfad gibt an, daß der Kontext das gesamte Dokument ist.

1. / Steht für das XML-*Stammelement*.

2. */tag1*
Steht für das Element *tag1* unter Root.

3. */tag1/tag2/..../tagn*
Steht für ein Element mit dem Namen *tagn* als untergeordnetes Element der absteigenden Kette von Root, *tag1*, *tag2* bis zu *tagn*-1.

4. *//tagn*
Steht für ein beliebiges Element mit dem Namen *tagn*, wobei doppelte Schrägstriche (//) auf keine oder mehrere Zufallsbefehle verweisen.

5. */tag1//tagn*
Steht für ein beliebiges Element mit dem Namen *tagn*, einem untergeordneten Element eines Elements mit dem Namen *tag1* unter Root, wobei doppelte Schrägstriche auf keine oder mehrere Zufallsbefehle verweisen.

6. */tag1/tag2/@attr1*
Steht für das Attribut *attr1* eines Elements mit dem Namen *tag2*, eines untergeordneten Elements des Elements *tag1* unter Root.

7. */tag1/tag2[@attr1="5"]*
   Steht für ein Element mit dem Namen *tag2*, dessen Attribut *attr1* den Wert 5 hat. *tag2* ist ein untergeordnetes Element mit dem Namen *tag1* unter Root.

8. */tag1/tag2[@attr1="5"]/.../tagn*
   Steht für ein Element mit dem Namen *tagn*, einem untergeordneten Element der absteigenden Kette von Root, *tag1*, *tag2* bis zu *tagn*-1, wobei das Attribut *attr1* von *tag2* den Wert 5 hat.

Platzhalterzeichen: Sie können einen Stern für ein Element in einem Standortpfad verwenden; dieser Stern steht dann für eine beliebige Zeichenfolge an dieser Stelle.

Einfacher Standortpfad: *Einfacher Standortpfad* ist die Syntax zu dem Standortpfad, mit dem Elemente und Attribute für Seitentabellen angegeben werden, die in der DAD-Datei mit XML-Spalten definiert sind. Der einfache Standortpfad ist eine Folge von Elementtypnamen, die durch einen einzelnen Schrägstrich (/) verbunden sind. Die Attributwerte sind hinter dem Elementtyp in eckigen Klammern angegeben. Tabelle 5 faßt die Syntax für den einfachen Standortpfad zusammen.

Tabelle 5. Syntax zum einfachen Standortpfad

Element	Standortpfad	Beschreibung
XML-Element	*/tag1/tag2/..../tagn-1/tagn*	Ein Elementinhalt, der von dem Element mit dem Namen *tagn* und seinen übergeordneten Elementen gekennzeichnet ist.
XML-Attribut	*/tag_1/tag_2/..../tag_n-1/tag_n/@attr1*	Ein Attribut mit dem Namen *attr1* des Elements, das durch *tagn* und seine übergeordneten Elemente gekennzeichnet ist.

XML Extender-Einschränkungen: Der XML Extender weist bei der Definition der Elemente oder Attribute in der DAD gewisse Einschränkungen zur Verwendung des Standortpfads auf. Da der XML Extender eine Eins-zu-Eins-Zuordnung zwischen einem Element bzw. Attribut und einer DB2-Spalte aufweist, schränkt er die in der DAD-Datei und in Funktionen zulässigen Syntaxregeln ein. Tabelle 6 auf Seite 59 beschreibt die Einschränkungen für den Standortpfad. Die in der vom Standortpfad unterstützten Spalte angegebenen Nummern beziehen sich auf die Syntaxdarstellungen in „Syntax des Standortpfads" auf Seite 57.

Tabelle 6. Einschränkungen des XML Extender zur Verwendung des Standortpfads

Verwendung des Standortpfads	Unterstützter Standortpfad
Element in der XML-Spalten-DAD-Zuordnung für Seitentabellen	3, 6 (einfacher Standortpfad, siehe Beschreibung in Tabelle 5 auf Seite 58)
Extraktions-UDFs	1-8[1]
Aktualisierungs-UDF	1-8[1]
Such-UDF des Text Extender	1-8

[1] Die Extraktions- und Aktualisierungs-UDFs unterstützen Standortpfade, die Prädikate mit Attribute haben, nicht jedoch Elemente.

Die DAD-Datei

Für XML-Spalten gibt die DAD in erster Linie an, wie die in einer XML-Spalte gespeicherten Dokumente indexiert werden sollen. Die DAD ist ein XML-formatiertes Dokument, das auf dem Client gespeichert ist. Wenn Sie XML-Dokumente mit einer DTD prüfen wollen, kann die DAD-Datei dieser DTD zugeordnet werden. Die DAD-Datei hat den Datentyp CLOB. Diese Datei kann bis zu 100 KB umfassen.

Die DAD-Datei für XML-Spalten enthält eine XML-Kopfzeile, gibt die Verzeichnispfade auf dem Client für die DAD-Datei und die DTD an und bietet eine Übersicht über die XML-Daten, die für die Indexierung in den Seitentabellen gespeichert werden sollen.

Zum Angeben der Zugriffs- und Speichermethode "XML-Spalte" verwenden Sie den folgenden Befehl in der DAD-Datei.

<Xcolumn>

Gibt an, daß die XML-Daten als vollständige XML-Dokumente in für XML-Daten aktivierten DB2-Spalten gespeichert bzw. daraus abgerufen werden sollen.

Eine für XML aktivierte Spalte hat den XML Extender-UDT. Anwendungen können die Spalte in einer beliebigen *Benutzertabelle* enthalten. Sie greifen auf die XML-Spaltendaten vor allem über SQL-Anwendungen und die UDFs des XML Extender zu.

Sie können den XML Extender-Verwaltungsassistenten oder einen Editor zum Erstellen und Aktualisieren der DAD verwenden.

XML-Objektgruppen planen

Beim Planen der XML-Objektgruppen müssen Sie unterschiedliche Punkte bedenken, je nachdem, ob Sie Dokumente aus DB2-Daten zusammensetzen, XML-Dokumente in DB2-Daten zerlegen oder beides. In den folgenden Abschnitten werden Fragen zur Planung von XML-Objektgruppen, zur Zusammenstellung und zum Zerlegen von Adressen erläutert.

Gültigkeitsprüfung

Nachdem Sie eine Zugriffs- und Speichermethode ausgewählt haben, können Sie festlegen, ob Ihre Daten geprüft werden sollen. Für die Prüfung der XML-Daten verwenden Sie eine DTD. Die Verwendung einer DTD stellt sicher, daß das XML-Dokument gültig ist, und sie ermöglicht die Ausführung einer strukturellen Suche in Ihren XML-Daten. Die DTD wird im DTD-Repository gespeichert.

Empfehlung: Prüfen Sie XML-Daten mit einer DTD. Zur Prüfung müssen Sie eine DTD im XML Extender-Repository haben. Im Abschnitt „Eine DTD im DTD-Repository speichern" auf Seite 78 finden Sie Informationen dazu, wie Sie eine DTD in das Repository einfügen. Die DTD-Anforderungen sind unterschiedlich, je nachdem, ob Sie XML-Dokumente zusammensetzen oder zerlegen.

- Zum Zusammensetzen können Sie die generierten XML-Dokumente nur anhand einer DTD prüfen. Die zu verwendende DTD ist in der DAD-Datei angegeben.
- Beim Zerlegen können Sie die zum Zusammensetzen zu verwendenden Dokumente mit verschiedenen DTDs prüfen. Mit anderen Worten, Sie können Dokumente mit der gleichen DAD-Datei zerlegen, aber dabei unterschiedliche DTDs aufrufen. Halten Sie sich beim Verweisen auf verschiedene DTDs an die folgenden Regeln:
  - Mindestens eine der DTDs muß in der Tabelle DTD_REF gespeichert werden. Alle DTDs können in dieser Tabelle gespeichert werden.
  - Die DTDs müssen eine gemeinsame Struktur aufweisen, die sich nur in den Unterelementen unterscheidet.
  - Sie müssen die Überprüfung in der DAD-Datei auswählen.
  - Die System-ID des XML-Dokuments muß die DTD-Datei mit einem vollständigen Pfadnamen angeben.
  - Die DAD-Datei enthält die Angabe, wie das Dokument zerlegt werden soll; Sie können daher nur gemeinsame Elemente und Attribute beim Zerlegen angeben. Elemente und Attribute, die innerhalb einer DTD eindeutig sind, können nicht zerlegt werden.

Wichtig: Entscheiden Sie vor dem Einfügen von XML-Daten in DB2, ob die Daten geprüft werden sollen. Der XML Extender unterstützt die Überprüfung von Daten nicht, die bereits in DB2 eingefügt wurden.

Überlegungen:

- Wenn Sie XML als Austauschformat verwenden wollen, müssen eine DTD verwenden.
- Die Überprüfung Ihrer XML-Daten kann eine gewisse Auswirkung auf den Durchsatz haben.

- Wenn Sie beim Zerlegen mehrere DTDs verwenden, können Sie nur gemeinsame Elemente und Attribute zerlegen.
- Wenn Sie eine einzige DTD verwenden, können Sie alle Elemente und Attribute zerlegen.
- Beim Zusammensetzen von Dokumenten können Sie nur eine DTD verwenden.

Die DAD-Datei

Für XML-Objektgruppen ordnet die DAD-Datei die Struktur des XML-Dokuments den DB2-Tabellen zu, aus denen Sie entweder das Dokument zusammensetzen oder in die Sie das Dokument zerlegen.

Wenn Sie beispielsweise ein Element mit dem Namen <Tax> in Ihrem XML-Dokument haben, müssen Sie <Tax> eventuell einer Spalte TAX zuordnen. Sie definieren die Beziehung zwischen den XML-Daten und den relationalen Daten in der DAD.

Die DAD-Datei wird entweder beim Aktivieren einer Objektgruppe oder beim Verwenden der DAD-Datei in den *gespeicherten Prozeduren* der XML-Objektgruppe angegeben. Die DAD ist ein XML-formatiertes Dokument, das auf dem Client gespeichert ist. Wenn Sie XML-Dokumente mit einer DTD prüfen wollen, kann die DAD-Datei dieser DTD zugeordnet werden. Wenn die DAD-Datei als Eingabeparameter der gespeicherten Prozeduren des XML Extender verwendet wird, hat sie den Datentyp CLOB. Diese Datei kann bis zu 100 KB umfassen.

Zum Angeben der Zugriffs- und Speichermethode ″XML-Objektgruppen″ verwenden Sie den folgenden Befehl in der DAD-Datei:

<Xcollection>

Gibt an, daß XML-Daten entweder aus XML-Dokumenten in eine Objektgruppe relationaler Tabellen zerlegt werden oder aus einer Objektgruppe relationaler Tabellen zu XML-Dokumenten zusammengesetzt werden sollen.

Eine XML-Objektgruppe ist ein virtueller Name für eine Gruppe relationaler Tabellen, die XML-Daten enthalten. Anwendungen können eine XML-Objektgruppe beliebiger Benutzertabellen aktivieren. Diese Benutzertabellen können vorhandene Tabellen mit unternehmensspezifischen Geschäftsdaten sein oder Tabellen, die der XML Extender vor kurzem erstellt hat. Sie greifen auf XML-Objektgruppendaten vor allem über die von XML Extender bereitgestellten gespeicherten Prozeduren zu.

Die DAD-Datei definiert die Baumstruktur der XML-Dokumente anhand der folgenden Knotenarten:

root_node
(Stammknoten) Gibt das Stammelement des Dokuments an.

element_node
(Elementknoten) Kennzeichnet ein Element, bei dem es sich um das Stammelement oder ein untergeordnetes Element handeln kann.

text_node
(Textknoten) Steht für den CDATA-Text eines Elements.

attribute_node
(Attributknoten) Steht für ein Attribut eines Elements.

Abb. 10 zeigt ein Fragment der Zuordnung, die in einer DAD-Datei verwendet wird. Die Knoten ordnen den Inhalt des XML-Dokuments den Tabellenspalten in einer relationalen Tabelle zu.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  ...
<Xcollection>
<SQL_stmt>
       ...
</SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE DAD SYSTEM "c:\dxx\sample\dtd\getstart.dtd"</doctype>
<root_node>
 <element_node name="Order">       --> Kennzeichnet das Element <Order>
 <attribute_node name="key">       --> Kennzeichnet das Attribut "key"
    <column name="order_key"/>     --> Definiert den Namen der Spalte,
                                       "order_key", der Element und
                                       Attribut zugeordnet werden
   </attribute_node>
   <element_node name="Customer">  --> Kennzeichnet ein untergeordnetes Element
                                            von <Order> als <Customer>
    <text_node>                    --> Gibt den CDATA-Text für das Element
                                            <Customer>
      <column name="customer">     --> Definiert den Namen der Spalte,
                                       "customer", der das untergeordnete
                                       Element zugeordnet ist
        </text_node>
     </element_node>
        ...
   </element_node>
      ...
<root_node>
</Xcollection>
</DAD>
```

Abbildung 10. Knotendefinitionen

In diesem Beispiel wurden den ersten beiden Spalten in der SQL-Anweisung Elemente und Attribute zugeordnet.

Der XML Extender unterstützt auch Verarbeitungsanweisungen für Formatvorlagen über das Element <stylesheet>. Das Element muß sich innerhalb des Stammknotens der DAD-Datei befinden, wobei der Dokumenttyp und der Prolog für das XML-Dokument definiert sind. Beispiel:

```
<Xcollection>
...
<prolog>...</prolog>
<doctype>...</doctype>
<stylesheet>?xml-stylesheet type="text/css" href="order.css"?</stylesheet>
<root_node>...</root_node>
...

</Xcollection>
```

Sie können den XML Extender-Verwaltungsassistenten oder einen Editor zum Erstellen und Aktualisieren der DAD-Datei verwenden. Das Element <stylesheet> wird derzeit vom XML Extender Verwaltungsassistenten nicht unterstützt.

Schemata für die Zuordnung von XML-Objektgruppen

Wenn Sie eine XML-Objektgruppe verwenden, müssen Sie ein *Zuordnungsschema* auswählen, das festlegt, wie XML-Daten in einer relationalen Datenbank dargestellt werden. Da XML-Objektgruppen einer hierarchischen Struktur entsprechen müssen, die in XML-Dokumenten mit einer relationalen Struktur verwendet wird, müssen Sie die Gemeinsamkeiten und Unterschiede der beiden Strukturen kennen. Abb. 11 auf Seite 64 zeigt, wie die hierarchische Struktur auf die relationalen Tabellenspalten abgebildet werden kann.

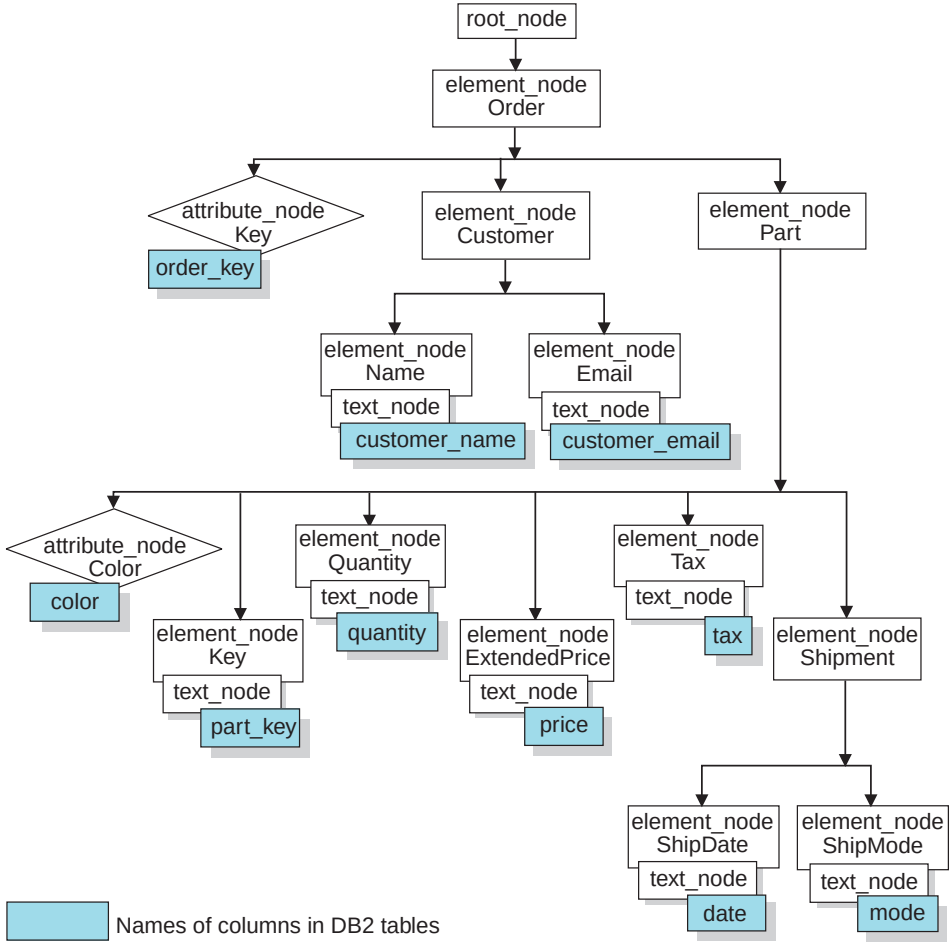


Abbildung 11. XML-Dokumentstruktur auf relationale Tabellenspalten abgebildet

Der XML Extender verwendet das Zuordnungsschema beim Zusammensetzen oder Zerlegen von XML-Dokumenten in verschiedenen relationalen Tabellen. Der XML Extender enthält einen Assistenten, der das Erstellen der DAD-Datei unterstützt. Bevor Sie jedoch die DAD-Datei erstellen, müssen Sie darüber nachdenken, wie Ihre XML-Daten der XML-Objektgruppe zugeordnet werden sollen.

Arten von Zuordnungsschemata: Das Zuordnungsschema wird im Element <Xcollection> in der DAD-Datei angegeben. Der XML Extender bietet zwei Arten von Zuordnungsschemata: *SQL-Zuordnung* und *Zuordnung über die relationale Datenbank (RDB_node)*. Beide Methoden verwenden das XSLT-Modell zum Definieren der Hierarchie des XML-Dokuments.

SQL-Zuordnung

Ermöglicht ein einfaches und direktes Zuordnen von relationalen Daten zu XML-Dokumenten über eine einzelne SQL-Anweisung und das *XSLT-Datenmodell*. Die SQL-Zuordnung wird für das Zusammensetzen verwendet, nicht jedoch beim Zerlegen. Die SQL-Zuordnung wird im Element SQL_stmt in der DAD-Datei definiert. Der Inhalt von SQL_stmt ist eine gültige SQL-Anweisung. SQL_stmt ordnet die Spalten in der SELECT-Klausel XML-Elementen oder Attributen zu, die in dem XML-Dokument verwendet werden. Sofern sie für das Zusammensetzen von XML-Dokumenten definiert sind, werden die Spaltennamen in der SELECT-Klausel der SQL-Anweisung verwendet, um den Wert eines *attribute_node* oder des Inhalts von *text_node* zu definieren. Die FROM-Klausel definiert die Tabellen mit den Daten; die WHERE-Klausel gibt die *join-* und Such*bedingung* an.

Die SQL-Zuordnung gibt DB2-Benutzern die Möglichkeit, mit SQL Daten zuzuordnen. Bei Verwendung der SQL-Zuordnung müssen Sie in der Lage sein, alle Tabellen in einer SELECT-Anweisung zu verbinden und damit eine Abfrage zu bilden. Wenn eine SQL-Anweisung nicht ausreicht, sollten Sie eine RDB_node-Zuordnung in Betracht ziehen. Zum Verbinden aller Tabellen wird die Beziehung *Primärschlüssel* und *Fremdschlüssel* zwischen diesen Tabellen empfohlen.

RDB_node-Zuordnung

Definiert den Standort des Inhalts eines XML-Elements oder des Werts eines XML-Attributs, so daß der XML Extender feststellen kann, wo die XML-Daten gespeichert bzw. von wo sie abgerufen werden sollen.

Der *RDB_node* enthält eine oder mehrere Knotendefinitionen für Tabellen, wahlweise auszuführende Spalten und wahlfreie Bedingungen. Die Tabellen und Spalten werden verwendet, um festzulegen, wie die XML-Daten in der Datenbank gespeichert werden sollen. Die Bedingung gibt die Kriterien zur Auswahl der XML-Daten an oder dazu, wie die XML-Objektgruppentabelle verbunden werden soll.

Zum Definieren eines Zuordnungsschemas erstellen Sie eine DAD mit einem Element <Xcollection>. Abb. 12 auf Seite 66 zeigt ein Fragment einer Muster-DAD-Datei mit einer SQL-Zuordnung "XML-Objektgruppe", die eine Gruppe von XML-Dokumenten aus Daten in drei relationalen Tabellen zusammensetzt.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  <dtdid>c:\dxx\samples\dad\getstart.dtd</dtdid>
<validation>YES</validation>
<Xcollection>
<SQL_stmt>
       SELECT o.order_key, customer, p.part_key, quantity, price, tax, date,
              mode, comment
         FROM order_tab o, part_tab p,
               table(select substr(char(timestamp(generate_unique())),
               as ship_id, date, mode, from ship_tab) as s
        WHERE p.price > 2500.00 and s.date > "1996-06-01" AND
              p.order_key = o.order_key and s.part_key = p.part_key
</SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
    <doctype>!DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
<root_node>
   <element_node name="Order">
  <attribute_node name="key">
<column_name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
  <text_node>
  <column name="customer"/>
  </text_node>
  <element_node>

...

  </element_node><!-end Part->
   </element_node><!-end Order->
</root_node>
</Xcollection>
</DAD>
```

Abbildung 12. SQL-Zuordnungsschema

Der XML Extender bietet verschiedene gespeicherte Prozeduren zur Verwaltung von Daten in einer XML-Objektgruppe. Diese gespeicherten Prozeduren unterstützen beide Arten der Zuordnung, erfordern jedoch, daß die DAD-Datei entsprechend den in „Voraussetzung für das Zuordnungsschema" definierten Regeln aufgebaut ist.

Voraussetzung für das Zuordnungsschema: In den folgenden Abschnitten werden die Voraussetzungen für die verschiedenen Arten der Zuordnungsschemata für XML-Objektgruppen beschrieben.

Voraussetzungen bei der Verwendung der SQL-Zuordnung

In diesem Zuordnungsschema müssen Sie das Element SQL_stmt im DAD-Element <Xcollection> angeben. SQL_stmt muß eine einzelne SQL-Anweisung enthalten, die mehrere relationale Tabellen über das Abfrage*prädikat* verbindet. Darüber hinaus sind die folgenden Klauseln erforderlich:

- **SELECT-Klausel**
  - Stellen Sie sicher, daß der Name der Spalte eindeutig ist. Wenn zwei Tabellen den gleichen Spaltennamen haben, verwenden Sie das Schlüsselwort AS, um für eine der Spalten einen Aliasnamen zu erstellen.
  - Gruppieren Sie die Spalten der gleichen Tabelle, und verwenden Sie die logische hierarchische Ebene der relationalen Tabellen. Dies bedeutet: Gruppieren Sie die Tabellen entsprechend ihrer Wichtigkeit, da sie der hierarchischen Struktur Ihres XML-Dokuments zugeordnet werden. In der SELECT-Klausel müssen die Spalten der Tabellen der höheren Ebene vor den Spalten der Tabellen der niedrigeren Ebenen stehen. Das folgende Beispiel verdeutlicht die hierarchische Beziehung zwischen den Tabellen:

    ```
    SELECT o.order_key, customer, p.part_key, quantity, price, tax,
           ship_id, date, mode
    ```

    In diesem Beispiel weisen `order_key` und `customer` aus der Tabelle ORDER_TAB die höchste relationale Stufe auf, da sie im hierarchischen Baum des XML-Dokuments höher angesiedelt sind. `ship_id`, `date` und `mode` aus der Tabelle SHIP_TAB befinden sich auf der niedrigsten relationalen Stufe.
  - Verwenden Sie zum Beginnen jeder Stufe einen Kandidatenschlüssel mit einer einzigen Spalte. Wenn in einer Tabelle kein solcher Schlüssel verfügbar ist, sollte die Abfrage mit einem Tabellenausdruck und der integrierten Funktion generate_unique() einen Schlüssel für diese Tabelle generieren. In dem obigen Beispiel ist `o.order_key` der Primärschlüssel für ORDER_TAB, und `part_key` ist der Primärschlüssel für PART_TAB. Diese Schlüssel erscheinen am Anfang ihrer eigenen auszuwählenden Spaltengruppen. Da die Tabelle SHIP_TAB keinen Primärschlüssel hat, muß einer generiert werden; in diesem Fall `ship_id`. Dieser Schlüssel wird als erste Spalte der Tabellengruppe SHIP_TAB aufgelistet. Verwenden Sie die FROM-Klausel zum Generieren des Primärschlüssels, wie im folgenden Beispiel gezeigt.

- **FROM-Klausel**
  - Verwenden Sie einen Tabellenausdruck und die integrierte Funktion generate_unique() zum Generieren eines einzelnen Schlüssels für Tabellen ohne einzelnen Primärschlüssel. Beispiel:

    ```
    FROM order_tab as o, part_tab as p,
           table(select substr(char(timestamp(generate_unique())),16)
           as ship_id, date, mode from ship_tab) as s
    ```

    In diesem Beispiel wird die Funktion generate_unique() auf einen CHAR-Datentyp von TIMESTAMP abgebildet, und sie erhält den Aliasnamen `ship_id`.
  - Verwenden Sie einen Aliasnamen, wenn eine Spalte eindeutig gekennzeichnet werden muß. Sie können beispielsweise `o` für ORDER_TAB verwenden, `p` für PART_TAB und `s` für SHIP_TAB.
- **WHERE-Klausel**
  - Geben Sie einen Primärschlüssel und einen Fremdschlüssel als Verknüpfungsbedingung an, die Tabellen in der Objektgruppe verbindet. Beispiel:

    ```
    WHERE p.price > 2500.00 AND s.date > "1996-06-01" AND
          p.order_key = o.order_key AND s.part_key = p.part_key
    ```
  - Geben Sie eine weitere Suchfunktion in dem Prädikat an. Jedes gültige Prädikat kann verwendet werden.
- **ORDER BY-Klausel**
  - Definieren Sie die ORDER BY-Klausel am Ende von SQL_stmt.
  - Stellen Sie sicher, daß die Spaltennamen den Spaltennamen in der SELECT-Klausel entsprechen.
  - Geben Sie die Spaltennamen oder Kennungen an, die die Entitäten im Konzept der Entitätsbeziehungen der Datenbank eindeutig kennzeichnen. Eine Kennung kann mit Hilfe eines Tabellenausdrucks und der integrierten Funktion generate_unique oder einer benutzerdefinierten Funktion (UDF) generiert werden.
  - Behalten Sie die Top-Down-Reihenfolge der Entitäten-Hierarchie bei. Die in der ORDER BY-Klausel angegebene Spalte muß für jede Entität die erste aufgelistete Spalte sein. Durch die Beibehaltung der Reihenfolge wird sichergestellt, daß die zu generierenden XML-Dokumente keine ungültigen Duplikate enthalten.
  - Qualifizieren Sie die Spalten in ORDER BY nicht durch ein Schema oder einen Tabellennamen.

Auch wenn für SQL_stmt die oben angeführten Einschränkungen gelten, ist es doch eine sehr leistungsstarke Funktion, da Sie in Ihrer WHERE-Klausel ein beliebiges Prädikat angeben können, sofern der Ausdruck im Prädikat die Spalten in den Tabellen verwendet.

Voraussetzungen bei der Verwendung der RDB_node-Zuordnung

Verwenden Sie mit dieser Zuordnungsmethode nicht das Element SQL_stmt im Element <Xcollection> der DAD-Datei. Verwenden Sie statt dessen das Element RDB_node element in allen Knoten der Ausgangsebene für *element_node* und für jeden attribute_node und text_node.

- **RDB_node für den Anfangs-element_node**

  Der *Anfangs-element_node* in der DAD-Datei steht für das Stammelement des XML-Dokuments. Geben Sie einen RDB_node für den Anfangs-element_node wie folgt an:

  – Geben Sie alle Tabellen an, die den XML-Dokumenten zugeordnet sind. Die folgende Zuordnung gibt beispielsweise drei Tabellen im RDB_node des element_node <Order> (des Anfangs-element_node) an:

  ```
  <element_node name="Order">
      <RDB_node>
          <table name="order_tab"/>
          <table name="part_tab"/>
          <table name="ship_tab"/>
          <condition>
             order_tab.order_key = part_tab.order_key AND
             part_tab.part_key = ship_tab.part_key
          </condition>
      </RDB_node>
  ```

  Das Bedingungselement kann leer sein oder fehlen, wenn in der Objektgruppe nur eine Tabelle vorhanden ist.

  – Wenn Sie ein Dokument zerlegen oder die über die DAD-Datei angegebene XML-Objektgruppe aktivieren, müssen Sie für jede Tabelle einen Primärschlüssel angeben. Der Primärschlüssel kann aus einer einzelnen Spalte oder mehreren Spalten, dem zusammengesetzten Schlüssel, bestehen. Der Primärschlüssel wird durch Hinzufügen eines Attributs zum Tabellenelement des RDB_node angegeben. Bei Angabe eines zusammengesetzten Schlüssels wird das Schlüsselattribut über die Namen der Schlüsselspalten, durch ein Leerzeichen getrennt, angegeben. Beispiel:

  ```
  <table name="part_tab" key="part_key, price"/>
  ```

  Die für das Zerlegen angegebenen Informationen werden beim Zusammensetzen eines Dokuments ignoriert.

– Verwenden Sie das Attribut orderBy, um XML-Dokumente, die Elemente oder Attribute mit mehrfachem Vorkommen enthalten, mit ihrer ursprünglichen Struktur wieder zusammenzusetzen. Dieses Attribut ermöglicht die Angabe des Namens der Spalte, die als Schlüssel zur Beibehaltung der Reihenfolge des Dokuments verwendet wird. Das Attribut orderBy ist Teil des Tabellenelements in der DAD-Datei; es ist ein wahlfreies Attribut.

Sie müssen den Tabellennamen und den Spaltennamen explizit ausschreiben.

- **RDB_node für jeden attribute_node und text_node**

  In diesem Zuordnungsschema sind die Daten in dem attribute_node und text_node für jeden element_node abgelegt. Der XML Extender muß daher wissen, wo in der Datenbank die Daten zu finden sind. Sie müssen einen RDB_node für jeden attribute_node und text_node angeben und so der gespeicherten Prozedur mitteilen, aus welcher Tabelle, welcher Spalte und mit welcher Abfragebedingung die Daten abgerufen werden können. Sie müssen Werte für die Tabelle und die Spalte angeben; der Bedingungswert ist wahlfrei.

  – Geben Sie den Namen der Tabelle an, die die Spaltendaten enthält. Der Tabellenname muß im RDB_node des Anfangselement_node angegeben sein. In diesem Beispiel ist für text_node des Elements <Price> die Tabelle als PART_TAB angegeben.

```
<element_node name="Price">
    <text_node>
        <RDB_node>
            <table name="part_tab"/>
            <column name="price"/>
            <condition>
                 price > 2500.00
            </condition>
        </RDB_node>
    </text_node>
</element_node>
```

  – Geben Sie den Namen der Spalte ein, die die Daten für den Elementtext enthält. Im vorigen Beispiel wurde die Spalte als PRICE angegeben.

  – Geben Sie eine Bedingung an, wenn Sie die XML-Dokumente über die Abfragebedingung erstellen wollen. Im obigen Beispiel ist die Bedingung als `price > 2500.00` angegeben. Nur die Daten, die diese Bedingung erfüllen, werden in die generierten XML-Dokumente einbezogen. Die Bedingung muß eine gültige WHERE-Klausel sein.

– Wenn Sie ein Dokument zerlegen oder die über die DAD-Datei angegebene XML-Objektgruppe aktivieren, müssen Sie den Spaltentyp für jeden attribute_node und text_node angeben. Auf diese Weise wird der richtige Datentyp für jede Spalte sichergestellt, wenn beim Aktivieren einer XML-Objektgruppe neue Tabellen erstellt werden. Spaltentypen werden durch Hinzufügen des Attributtyps zum Spaltenelement angegeben. Beispiel:

```
<column name="order_key" type="integer"/>
```

Die für das Zerlegen angegebenen Informationen werden beim Zusammensetzen eines Dokuments ignoriert.

Mit dem Ansatz über die RDB_node-Zuordnung brauchen Sie keine SQL-Anweisungen anzugeben. Das Aufstellen komplexer Abfragebedingungen im Element RDB_node ist jedoch unter Umständen schwieriger. Die Verwendung eines Ausdrucks bzw. einer Operation *union* ist beispielsweise weniger leistungsstark als der Ansatz SQL-zu-XML.

Anforderungen für die Tabellengröße beim Zerlegen

Beim Zerlegen wird die RDB_node-Zuordnung verwendet zur Angabe, wie ein XML-Dokument in DB2-Tabellen zerlegt werden soll durch Extrahieren der Element- und Attributwerte in Tabellenzeilen. Die Werte aus den einzelnen XML-Dokumenten werden in einer oder mehreren DB2-Tabellen gespeichert. Jede Tabelle kann maximal 1024 Zeilen aus jedem Dokument enthalten.

Wenn ein XML-Dokument beispielsweise in fünf Tabellen zerlegt wird, kann jede dieser fünf Tabellen bis zu 1024 Zeilen für ein bestimmtes Dokument haben. Wenn die Tabelle Zeilen für mehrere Dokumente enthält, kann sie bis zu 1024 Zeilen für jedes Dokument enthalten. Wenn die Tabelle 20 Dokumente enthält, kann sie also 20.480 Zeilen enthalten, 1024 pro Dokument.

Die Verwendung mehrfach vorkommender Elemente (Elemente mit Standortpfaden, die mehr als einmal in der XML-Struktur vorkommen) wirkt sich auf die Anzahl der Zeilen aus. Ein Dokument, das beispielsweise ein Element <Part> 20 mal enthält, kann als 20 Zeilen in eine Tabelle zerlegt werden. Beachten Sie bei Verwendung mehrfach vorkommender Elemente diese Einschränkung der Tabellengröße.

Kapitel 4. XML-Daten verwalten

Die Verwaltungs-Tasks für den XML Extender umfassen das Aktivieren der Datenbank und der Tabellenspalten für XML und das Zuordnen der XML-Daten zu den relationalen DB2-Strukturen. Der XML Extender bietet verschiedene Verwaltungs-Tools, je nachdem, ob Sie eine Anwendung entwickeln oder einfach mit einem Assistentenprogramm (Wizard) arbeiten wollen. Sie können die Verwaltungs-Tasks für den XML Extender mit den folgenden Tools ausführen:

- XML Extender-Verwaltungsassistent
- Befehl **dxxadm**
- Gespeicherte Prozeduren des XML Extender zur Verwaltung

In diesem Kapitel sind die Verwaltungs-Tasks für den Verwaltungsassistenten und dem Befehl **dxxadm** beschrieben. Die gespeicherten Prozeduren zur Verwaltung sind im Abschnitt „Gespeicherte Prozeduren zur Verwaltung" auf Seite 223 beschrieben.

Zur Ausführung der Tasks in diesem Kapitel sollten Sie mit den im Abschnitt „Planung der Verwaltung" auf Seite 48 beschriebenen Konzepten und Planungs-Tasks vertraut sein.

In den folgenden Abschnitten werden die Verwaltungs-Tasks zu XML Extender beschrieben:

1. „Verwaltungsassistent starten"
2. „Datenbank für XML aktivieren" auf Seite 77
3. „Eine DTD im DTD-Repository speichern" auf Seite 78
4. „XML-Spalten oder -Objektgruppen definieren" auf Seite 81
5. „Mit XML-Spalten arbeiten" auf Seite 81
6. „Mit XML-Objektgruppen arbeiten" auf Seite 93

Verwaltungsassistent starten

Dieser Abschnitt enthält Informationen zum Einstellen und Aufrufen des XML Extender-Verwaltungsassistenten.

Verwaltungsassistent einstellen

Stellen Sie sicher, daß Sie die in der Informationsdatei zu Ihrem Betriebssystem beschriebenen Schritte zur Installation und Einstellung des Verwaltungsassistenten ausgeführt haben. Hierzu gehört auch das Sicherstellen, daß die

Bind-Anweisung ausgeführt und die erforderliche Software in den CLASSPATH-Anweisungen eingebunden wurde.

- Die bind-Anweisungen sind in den Readme-Dateien zu dem Assistenten und in der Einführungs-Beispieldatei enthalten:

  /dxx_install/samples/cmd/getstart_prep.cmd

- Die CLASSPATH-Anweisung sollte ungeführt wie folgt aussehen (die Zeilenumbrüche sind nur zur übersichtlicheren Darstellung eingefügt):

```
.;C:\java\db2java.zip;C:\java\runtime.zip;C:\java\sqlj.zip;
C:\dxx\dxxadmin\dxxadmin.jar;C:\dxx\dxxadmin\dxxadmin.cmd;
C:\dxx\dxxadmin\html\dxxahelp*.htm;C:\java\jdk\lib\classes.zip;
C:\java\swingall.jar
```

  Wichtig: Für den Assistenten muß ein Pfadname ohne Leerzeichen angegeben werden. Wenn Sie die IBM DB2 Universal Database V7.1 Standardinstallation verwenden, befindet sich SQLLIB\java unter dem Verzeichnis Program Files. *Kopieren* Sie in diesem Fall den Java-Code in einen einfacheren Pfad. Verschieben Sie den Java-Code nicht, und ändern Sie CLASSPATH; das Control Center erfordert die Angabe des CLASSPATH während der Installation.

Der XML Extender Verwaltungsassistent verwendet eine Klassendatei. Der vollständige Dateiname der Hauptklassendatei der XML Extender-Verwaltung lautet:

```
com.ibm.dxx.admin.Admin.
```

Ändern Sie diese Datei für Ihr System, um den Assistenten aufzurufen.

- Zum Aufrufen über das JDK geben Sie ein:

```
java -classpath klassenpfad com.ibm.dxx.admin.Admin
```

- Zum Aufrufen über JRE geben Sie ein:

```
jre -classpath klassenpfad com.ibm.dxx.admin.Admin
```

  klassenpfad gibt hierbei an:

  – Die Umgebungsvariable %CLASSPATH% zur Angabe, wo sich die Klassendateien für den Verwaltungsassistenten befinden. Bei Verwendung dieser Option muß Ihr System-CLASSPATH auf das Verzeichnis *dxx_install*/dxxadmin verweisen, das die folgenden Dateien enthält: dxxadmin.jar, xml4j.jar und db2java.zip. Beispiel:

```
java -classpath %CLASSPATH% com.ibm.dxx.admin.Admin
```

  – Ein Überschreiben der Umgebungsvariablen %CLASSPATH% mit Zeigern auf die Dateien im Verzeichnis *dxx_install*/dxxadmin, von dem aus Sie den XML Extender-Verwaltungsassistenten aufrufen. Beispiel:

```
java -classpath dxxadmin.jar;xml4j.jar;db2java.zip
com.ibm.dxx.admin.Admin url=jdbc:db2:mydb userid=db2xml
password=db2xml driver=COM.ibm.db2.jdbc.app.DB2Driver
```

Optional können Sie zur Laufzeit die folgenden Parameter zur Laufzeit angeben:

url Vollständig qualifizierter URL-Pfad auf die IBM DB2 UDB-Datenquelle, zu der eine Verbindung hergestellt werden soll. Beispiel: jdbc:db2://dxx.stl.ibm.com:8080/guidb. Im Assistent als „Adresse" beschriftet.

userid Benutzer-ID, die für den Zugriff auf die obige Datenquelle verwendet werden soll. Beispiel: db2guest.

password
Kennwort für die obige Benutzer-ID. Beispiel: guest.

driver JDBC-Treibername für die obige URL. Standardwert: COM.ibm.db2.jdbc.net.DB2Driver. Im Assistent als „JDBC-Treiber" beschriftet.

Weitere Informationen zu diesen Angaben finden Sie im Abschnitt „Verwaltungsassistent aufrufen".

Verwaltungsassistent aufrufen

Führen Sie zum Aufrufen des XML Extender-Verwaltungsassistenten die folgenden Schritte aus.

1. Rufen Sie den Assistenten auf.

   Für Windows NT:

   Klicken Sie doppelt auf das Symbol des XML Extender-Verwaltungsassistenten im Desktop.

   Für AIX, Sun Solaris und Linux:

   Führen Sie die Datei `dxxadmin` aus.

   Das Anmeldungsfenster des Verwaltungsassistenten wird geöffnet.

Wenn Sie den XML Extender-Verwaltungsassistenten aufrufen, wird das Anmeldefenster angezeigt. Melden Sie sich bei der Datenbank an, die Sie beim Arbeiten mit den XML-Daten verwenden wollen. XML Extender stellt die Verbindung zu dem aktuellen Exemplar her.

2. Geben Sie im Feld **Adresse** die vollständig qualifizierte JDBC-URL zu der IBM DB2 UDB-Datenquelle ein, zu der eine Verbindung hergestellt werden soll. Die Adresse hat die folgende Syntax:

   Für eigenständige Konfigurationen (empfohlen):

   `jdbc:db2:`*`datenbankname`*

   Hierbei gilt folgendes:

   datenbankname
   Die Datenbank, zu der Sie eine Verbindung herstellen und in der Sie XML-Dokumente speichern wollen.

Beispiel:

```
jdbc:db2:sales_db
```

Für Netzkonfigurationen:

```
jdbc:db2://servername:port_number/datenbank_name
```

Hierbei gilt folgendes:

servername
Der Name des Servers, auf dem sich der XML Extender befindet.

anschlußnummer
Die für die Verbindung mit dem Server verwendete Anschlußnummer. Geben Sie zum Ermitteln der Anschlußnummer den folgenden Befehl in einer DB2-Befehlszeile auf der Server-Maschine ein:

```
db2jstrt anschlußnummer
```

Benutzer von Windows NT können die Anschlußnummer in der Datei \winnt\system32\driver\etc\services überprüfen.

datenbankname
Die Datenbank, zu der Sie eine Verbindung herstellen und in der Sie XML-Dokumente speichern wollen.

Beispiel:

```
jdbc:db2://host1.ibm.com:8080/sales_db
```

3. Geben Sie die DB2-Benutzer-ID und das Kennwort für die gewünschte Datenbank in den Feldern **Benutzer-ID** und **Kennwort** bzw. überprüfen Sie diese Angaben.
4. Überprüfen Sie im Feld **JDBC-Treiber** den JDBC-Treibernamen für die angegebene Adresse mit den folgenden Werten:

   Für eigenständige Konfigurationen (Standardwert und empfohlen):

   ```
   COM.ibm.db2.jdbc.app.DB2DRIVER
   ```

   Für Netzkonfigurationen:

   ```
   COM.ibm.db2.jdbc.net.DB2DRIVER
   ```

5. Klicken Sie **Beenden** an, um eine Verbindung zum Assistenten herzustellen, und fahren Sie fort mit dem LaunchPad-Fenster.

Das LaunchPad-Fenster bietet Zugriff auf fünf Verwaltungsassistenten. Mit diesen Assistenten haben Sie folgende Möglichkeiten:

- Aktivieren einer Datenbank
- Hinzufügen einer DTD zum DTD-Repository

- Arbeiten mit DAD-Dateien für:
  - XML-Spalten
  - XML-Objektgruppen
- Arbeiten mit XML-Spalten
- Mit XML-Objektgruppen arbeiten

Datenbank für XML aktivieren

Zum Speichern oder Abrufen von XML-Dokumenten aus DB2 mit XML Extender aktivieren Sie die Datenbank für XML. XML Extender aktiviert die Datenbank, zu der Sie eine Verbindung haben, mit dem aktuellen Exemplar.

Wenn Sie eine Datenbank für XML aktivieren, führt der XML Extender folgende Schritte aus:

- Erstellt alle benutzerdefinierten Typen (UDTs) und benutzerdefinierten Funktionen (UDFs)
- Erstellt und füllt die Steuertabellen mit den erforderlichen Metadaten für den XML Extender
- Erstellt das db2xml-Schema und ordnet die erforderlichen Berechtigungen zu

  Der vollständige Name einer XML-Funktion lautet *schemaname.funktionsname*, wobei *schemaname* eine Kennung ist, die eine logische Gruppierung für SQL-*Objekte* bietet. Sie können den vollständigen Namen bei jedem Verweis auf eine UDF oder einen UDT verwenden. Sie können den Schemanamen beim Verweis auf eine UDF oder einen UDT auch weglassen; in diesem Fall verwendet DB2 den Funktionspfad zum Ermitteln des gewünschten Funktions- oder Datentyps.

Verwaltungsassistent verwenden

Führen Sie zum Aktivieren einer Datenbank für XML-Daten die folgenden Schritte aus:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Datenbank aktivieren** im LaunchPad-Fenster an, um die aktuelle Datenbank zu aktivieren.

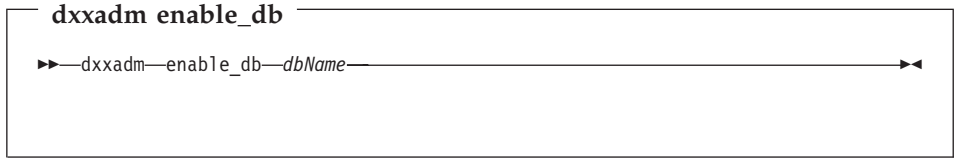
   Wenn eine Datenbank bereits aktiviert ist, kann nur **Datenbank inaktivieren** ausgewählt werden.

   Wenn die Datenbank aktiviert ist, kehren Sie zurück zum LaunchPad-Fenster.

Von der DB2-Befehls-Shell aus

Geben Sie **dxxadm** in der Befehlszeile ein, und geben Sie dabei die zu aktivierende Datenbank an.

Syntax:



Parameter:

dbName
Der Name der zu aktivierenden Datenbank.

Beispiel: Aktiviert eine vorhandene Datenbank mit dem Namen SALES_DB.

```
dxxadm enable_db SALES_DB
```

Eine DTD im DTD-Repository speichern

Sie können mit einer DTD die XML-Daten in einer XML-Spalte oder in einer XML-Objektgruppe überprüfen. Die DTD überprüft die XML-Spalte und wird zum Definieren von DAD-Dateien verwendet, die für die strukturelle Suche und zum Zusammensetzen und Zerlegen von Objektgruppen verwendet werden.

Alle DTDs werden in dem DTD-Repository gespeichert, einer DB2-Tabelle mit dem Namen DTD_REF. Sie hat den Schemanamen db2xml. Jede DTD in der Tabelle DTD_REF hat eine eindeutige ID. Wenn Sie eine Datenbank für XML aktivieren, erstellt der XML Extender die Tabelle DTD_REF.

Weitere Informationen zu DTDs finden Sie in den Abschnitten „XML-Spalten planen" auf Seite 51 und „XML-Objektgruppen planen" auf Seite 59.

Sie können die DTD über die DB2-Befehls-Shell einfügen oder über den Verwaltungsassistenten.

Verwaltungsassistent verwenden

Führen Sie zum Einfügen einer DTD die folgenden Schritte aus:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **DTD importieren** im LaunchPad-Fenster an, um eine vorhandene DTD-Datei in das DTD-Repository der aktuellen Datenbank zu importieren. Das Fenster **Importieren einer DTD** wird angezeigt.
3. Geben Sie den Namen der DTD-Datei im Feld **DTD-Dateiname** ein, oder klicken Sie **...** an, um eine vorhandene DTD-Datei zu suchen.
4. Geben Sie die DTD-ID im Feld **DTD-ID** ein.

   Die DTD-ID ist eine Kennung der DTD; dies kann der Pfad sein, der den Standort der DTD auf dem lokalen System angibt. Die DTD-ID muß dem Wert entsprechen, der in der DAD-Datei für das Element <DTD-ID> angegeben ist.
5. Geben Sie wahlweise den Namen des Autors der DTD im Feld **Autor** ein.

   Der XML Extender zeigt automatisch den Namen des Autors an, wenn er in der DTD angegeben ist.
6. Klicken Sie **Beenden** an, um die DTD in die DTD-Repository-Tabelle DB2XML.DTD_REF einzufügen, und kehren Sie zurück zum LaunchPad-Fenster.

Von der DB2-Befehls-Shell aus

Geben Sie eine SQL-Anweisung INSERT für die Tabelle DTD_REF anhand des Schemas in Tabelle 7 ein:

Tabelle 7. Das Schema für die Tabelle DTD_REF DTD

Spaltenname	Datentyp	Beschreibung
DTD-ID	VARCHAR(128)	Primärschlüssel (eindeutig und nicht NULL). Der Primärschlüssel wird zur Kennzeichnung der DTD verwendet; er muß mit der SYSTEM ID in der Zeile DOCTYPE in jedem XML-Dokument übereinstimmen, wenn eine Gültigkeitsprüfung verwendet wird. Wenn der Primärschlüssel in der DAD-Datei angegeben ist, muß die DAD-Datei entsprechend dem in der DTD definierten Schema aufgebaut sein.
CONTENT	XMLCLOB	Der Inhalt der DTD.
USAGE_COUNT	INTEGER	Die Anzahl der XML-Spalten und XML-Objektgruppen in der Datenbank, die diese DTD zur Definition einer DAD verwenden.
AUTHOR	VARCHAR(128)	Autor der DTD, wahlfreie Informationen, die der Benutzer eingeben kann.
CREATOR	VARCHAR(128)	Die Benutzer-ID, über die die erste Einfügung vorgenommen wird.
UPDATOR	VARCHAR(128)	Die Benutzer-ID, über die die erste Aktualisierung vorgenommen wird.

Beispiel:

```
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'benutzer1',
    'benutzer1', 'benutzer1')
```

Wichtig für XML-Objektgruppen: Die DTD-ID ist ein Pfad, der den Standort der DTD auf dem lokalen System angibt. Die DTD-ID muß dem Wert entsprechen, der in der DAD-Datei für das Element <DTD-ID> angegeben ist.

XML-Spalten oder -Objektgruppen definieren

In den folgenden Abschnitten wird beschrieben, wie Ihre Datenbank für XML-Spalten und -Objektgruppen eingestellt und definiert wird und wie die erforderlichen Schemata für die Datenzuordnung vorbereitet werden.

Diese Abschnitte sind:

- „Mit XML-Spalten arbeiten"
- „Mit XML-Objektgruppen arbeiten" auf Seite 93

Mit XML-Spalten arbeiten

Zum Einstellen der XML-Spalten müssen Sie die DAD-Datei für den Zugriff auf Ihre XML-Daten definieren und die Spalten für XML-Daten in einer XML-Tabelle aktivieren. Ein wichtiges Konzept beim Erstellen der DAD ist das Verständnis der Pfadsyntax, da über diese Syntax die Element- und Attributwerte zugeordnet werden, die Sie in DB2-Tabellen indexieren wollen. Weitere Informationen zum Standortpfad und seiner Syntax finden Sie im Abschnitt „Standortpfad" auf Seite 56.

DAD-Datei erstellen oder editieren

Beim Angeben einer DAD-Datei definieren Sie die Attribute und Schlüsselelemente Ihrer zu durchsuchenden Daten. Der XML Extender verwendet diese Informationen zum Erstellen von Seitentabellen, so daß Sie Ihre Daten indexieren können, um sie schneller zu finden. Informationen zur Planung beim Erstellen der DAD-Datei finden Sie im Abschnitt „Die DAD-Datei" auf Seite 59.

Vorbereitungen

- Machen Sie sich mit der hierarchischen Struktur Ihrer XML-Daten vertraut, damit Sie die Schlüsselelemente und Attribute für die Indexierung und die Schnellsuche definieren können.
- Bereiten Sie die DTD des XML-Dokuments vor und fügen Sie sie in die Tabelle DTD_REF ein. Dieser Schritt ist für die Gültigkeitsprüfung erforderlich.

Verwaltungsassistent verwenden

Führen Sie zum Erstellen einer DAD-Datei die folgenden Schritte aus:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.

2. Klicken Sie **Mit DAD-Dateien arbeiten** im LaunchPad-Fenster an, um eine XML DAD-Datei zu editieren oder zu erstellen. Das Fenster **Eine DAD angeben** wird geöffnet.
3. Wählen Sie aus, ob Sie eine bestehende DAD-Datei editieren oder eine neue DAD-Datei erstellen wollen.
   - **So editieren Sie eine vorhandene DAD:**
     a. Klicken Sie im Kontextmenü **...** an, um nach vorhandenen DAD-Dateien zu suchen, oder geben Sie den Namen der DAD-Datei im Feld **Dateiname** ein.
     b. Vergewissern Sie sich, daß der Assistent die angegebene DAD-Datei erkennt.
        – Wenn der Assistent die angegebene DAD-Datei erkennt, können Sie **Weiter** auswählen, und **XML-Spalte** wird im Feld **Typ** angezeigt.
        – Wenn der Assistent die angegebene DAD-Datei nicht erkennt, kann **Weiter** nicht ausgewählt werden. Geben Sie entweder den DAD-Dateinamen im Feld **Dateiname** erneut ein, oder klicken Sie **Öffnen** an, um erneut nach vorhandenen DAD-Dateien zu suchen. Fahren Sie mit dieser Prozedur fort, bis **Weiter** ausgewählt werden kann.
     c. Klicken Sie **Weiter** an.
   - **So erstellen Sie eine neue DAD:**
     a. Lassen Sie das Feld **Dateiname** leer.
     b. Klicken Sie im Menü **Typ** die Option **XML-Spalte** an.
     c. Klicken Sie **Weiter** an.
4. Wählen Sie im Fenster **Gültigkeitsprüfung auswählen** aus, ob Ihre XML-Dokumente mit einer DTD geprüft werden sollen.
   - So führen Sie eine Überprüfung durch:
     a. Klicken Sie **Prüfen Sie XML-Dokumente mit der DTD** an.
     b. Wählen Sie die für die Gültigkeitsprüfung zu verwendende DTD im Menü **DTD-ID** aus.

     Wenn Sie keine DTDs in das DTD-Repository für Ihre Datenbank importiert haben, können Sie keine Gültigkeitsprüfung für Ihre XML-Dokumente durchführen.
   - Klicken Sie **Nicht prüfen** an, um fortzufahren, ohne Ihre XML-Dokumente zu überprüfen.
5. Klicken Sie **Weiter** an.
6. Wählen Sie im Fenster **Seitentabellen** aus, ob eine neue Seitentabelle hinzugefügt, eine vorhandene Seitentabelle editiert oder eine vorhandene Seitentabelle entfernt werden soll.

- **So fügen Sie eine neue Seitentabelle oder eine Seitentabellenspalte hinzu:**

  Zum Hinzufügen einer neuen Seitentabelle definieren Sie die Spalten in der Tabelle. Führen Sie für jede Spalte in einer Seitentabelle die folgenden Schritte aus.

  a. Füllen Sie die Felder des Bereichs **Details** im Fenster **Seitentabellen** aus.

     1) **Tabellenname**: Geben Sie den Namen der Tabelle an, die die Spalte enthält. Beispiel:

        ```
        ORDER_SIDE_TAB
        ```

     2) **Spaltenname**: Geben Sie den Namen der Spalte ein. Beispiel:

        ```
        CUSTOMER_NAME
        ```

     3) **Typ**: Wählen Sie in dem Menü den Typ der Spalte aus. Beispiel:

        ```
        XMLVARCHAR
        ```

     4) **Länge (Nur Typ VARCHAR)**: Geben Sie die maximale Anzahl von VARCHAR-Zeichen ein. Beispiel:

        ```
        30
        ```

     5) **Pfad**: Geben Sie den Standortpfad des Elements oder Attributs ein. Beispiel:

        ```
        /ORDER/CUSTOMER/NAME
        ```

        Die Syntax für den Standortpfad ist im Abschnitt „Standortpfad" auf Seite 56 beschrieben.

     6) **Mehrfaches Vorkommen**: Wählen Sie **Nein** oder **Ja** in dem Menü aus.

        Gibt an, ob der Standortpfad dieses Elements bzw. Attributs in einem Dokument mehrfach verwendet werden kann.

        Wichtig Wenn Sie für eine Spalte ein mehrfaches Vorkommen angeben, können Sie nur eine Spalte in der Seitentabelle angeben, die die Spalte enthält.

  b. Klicken Sie **Hinzufügen** an, um die Spalte hinzuzufügen.

  c. Fahren Sie mit dem Hinzufügen, Editieren oder Entfernen von Spalten für die Seitentabellen fort, oder klicken Sie **Weiter** an.

- **So editieren Sie eine vorhandene Seitentabellenspalte:**

  Sie können eine Seitentabelle aktualisieren, indem Sie die Definitionen der vorhandenen Spalten ändern.

  a. Klicken Sie die Seitentabelle und den Namen der Spalte an, die Sie editieren wollen.

  b. Editieren Sie die Felder im Bereich **Details**.

  c. Klicken Sie **Ändern** an, um die Änderungen zu speichern.

d. Fahren Sie mit dem Hinzufügen, Editieren oder Entfernen von Spalten für alle Seitentabellen fort, oder klicken Sie **Weiter** an.

- **So entfernen Sie eine vorhandene Seitentabellenspalte:**
  a. Klicken Sie die Seitentabelle und die Spalte an, die Sie entfernen wollen.
  b. Klicken Sie **Entfernen** an.
  c. Fahren Sie mit dem Hinzufügen, Editieren oder Entfernen von Seitentabellenspalten fort, oder klicken Sie **Weiter** an.
- **So entfernen Sie eine vorhandene Seitentabelle:**

  Zum Entfernen einer gesamten Seitentabelle löschen Sie alle Spalten in der Tabelle.
  a. Klicken Sie alle Seitentabellenspalten für die zu entfernende Tabelle an.
  b. Klicken Sie **Entfernen** an.
  c. Fahren Sie mit dem Hinzufügen, Editieren oder Entfernen von Seitentabellenspalten fort, oder klicken Sie **Weiter** an.

7. Geben Sie einen Ausgabedateinamen für die geänderte DAD-Datei im Feld **Dateiname** des Fensters **Eine DAD angeben** ein.
8. Klicken Sie **Beenden** an, um die DAD-Datei zu speichern und zum LaunchPad-Fenster zurückzukehren.

Von der DB2-Befehls-Shell aus

Die DAD-Datei ist eine XML-Datei, die in einem beliebigen Texteditor erstellt werden kann.

Führen Sie zum Erstellen einer DAD-Datei die folgenden Schritte aus:

1. Öffnen Sie einen Texteditor.
2. Erstellen Sie die DAD-Kopfzeilen mit der folgenden Syntax:

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "pfad\dtd\dad.dtd"> --> Pfad- und
   Dateiname der DTD für die DAD-Datei
   ```
3. Fügen Sie die Befehle <DAD></DAD> ein.
4. Geben Sie in dem Befehl <DAD> wahlweise die DTD-ID-Kennung an, die die DAD-Datei mit der DTD des XML-Dokuments für die Gültigkeitsprüfung verbindet:

   ```
   <dtdid>pfad\dtd_name.dtd</dtdid> --> Pfad- und Dateiname der DTD
   für Ihre Anwendung
   ```

   Die DTD-ID ist für die Gültigkeitsprüfung erforderlich; sie muß dem Wert der DTD-ID entsprechen, der beim Einfügen der DTD in die DTD-Referenztabelle (db2xml.DTD_REF) verwendet wurde.
5. Geben Sie an, ob eine Gültigkeitsprüfung durchgeführt werden soll (ob über eine DTD sichergestellt werden soll, daß das XML-Dokument ein gültiges XML-Dokument ist).

Beispiel:

```
<validation>YES</validation>  --> geben Sie YES oder NO an
```

Wenn Sie YES angeben, müssen Sie im vorigen Schritt eine DTD-ID angegeben und eine DTD in die Tabelle DTD_REF eingefügt haben.

6. Verwenden Sie das Element <Xcolumn> zum Definieren der Zugriffs- und Speichermethode als XML-Spalte.

   ```
   <Xcolumn>
   </Xcolumn>
   ```

7. Definieren Sie jede Seitentabelle und die wichtigen Elemente und Attribute für die Indexierung, um eine strukturelle Suche zu ermöglichen. Führen Sie für jede Tabelle die folgenden Schritte aus. In den folgenden Schritten werden Beispiele aus einer DAD-Musterdatei verwendet; diese Datei ist im Abschnitt „DAD-Datei: XML-Spalte" auf Seite 287 abgebildet:

   a. Fügen Sie die Befehle <TABLE></TABLE> und das Namensattribut ein.

      ```
      <table name="order_tab">
      </table>
      ```

   b. Fügen Sie nach dem Befehl <TABLE> einen Befehl <COLUMN> und die entsprechenden Attribute für jede Spalte in die Tabelle ein:
      - ***name***: Der Name der Spalte
      - ***type***: Der Typ der Spalte
      - ***path***: Der Standortpfad des Elements bzw. Attributs. Die Syntax für den Standortpfad ist im Abschnitt „Standortpfad" auf Seite 56 beschrieben.
      - ***multi_occurrence***: Gibt an, ob dieses Element bzw. Attribut in einem Dokument mehrfach verwendet werden kann

      ```
      <table ...>
         <column name="order_key"
                 type="integer"
                 path="/Order/@key"
                 multi_occurrence="NO"/>
         <column name="customer"
                 type="varchar(50)"
                 path="/Order/Customer/Name"
                 multi_occurrence="NO"/>
      </table>
      ```

8. Stellen Sie sicher, daß Sie einen Endbefehl </TABLE> nach der letzten Spaltendefinition verwenden.
9. Stellen Sie sicher, daß Sie einen Endbefehl </Xcolumn> nach dem letzten Befehl </TABLE> verwenden.
10. Stellen Sie sicher, daß Sie einen Endbefehl </DAD> nach dem letzten Befehl </Xcolumn> verwenden.

XML-Tabelle erstellen oder ändern

Zum Speichern intakter XML-Dokumente in einer Tabelle müssen Sie eine Tabelle erstellen oder ändern, so daß sie eine Spalte mit einem benutzerdefinierten XML-Typ (UDT) enthält. Die Tabelle wird als *XML-Tabelle* bezeichnet; eine solche Tabelle enthält XML-Dokumente. Bei der Tabelle kann es sich um eine neue oder um eine geänderte Tabelle handeln. Wenn eine Tabelle eine Spalte des Typs XML enthält, können Sie die Spalte für XML aktivieren.

Sie können eine vorhandene Tabelle mit einer Spalte des XML-Typs mit dem Verwaltungsassistenten ändern oder über die DB2-Befehls-Shell.

Verwaltungsassistent verwenden

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit XML-Spalten arbeiten** im LaunchPad-Fenster an. Das Fenster **Eine Task auswählentask** wird geöffnet.
3. Klicken Sie **Eine XML-Spalte hinzufügen** an. Das Fenster **Eine XML-Spalte hinzufügen** wird geöffnet.
4. Wählen Sie den Namen der Tabelle im Aktionsfenstermenü **Tabellenname** aus, oder geben Sie den Namen der Tabelle ein, die Sie ändern wollen. Beispiel:

   ```
   SALES_DB
   ```

5. Geben Sie den Namen der Spalte, die der Tabelle hinzugefügt werden soll, im Feld **Spaltenname** ein. Beispiel:

   ```
   ORDER
   ```

6. Wählen Sie den UDT für die Spalte im Aktionsfenstermenü **Spaltentyp** aus. Beispiel:

   ```
   XMLVARCHAR
   ```

7. Klicken Sie **Beenden** an, um die Spalte des XML-Typs hinzuzufügen.

Von der DB2-Befehls-Shell aus

Erstellen (″CREATE″) oder ändern (″ALTER″) Sie eine Tabelle mit einer Spalte eines XML-Typs in der column-Klausel der Anweisung CREATE TABLE bzw. ALTER TABLE.

Beispiel: Sie wollen in der Anwendung sales eine XML-formatierte Bestellzeile in einer Spalte ORDER einer Anwendungstabelle SALES_TAB speichern. Diese Tabelle enthält auch die beiden Spalten INVOICE_NUM und SALES_PERSON. Da es sich um eine kleine Bestellung handelt, speichern Sie sie mit dem Typ XMLVARCHAR. Der Primärschlüssel ist INVOICE_NUM.

Die folgende Anweisung CREATE TABLE erstellt die Tabelle mit einer Spalte des Typs XML:

```
CREATE TABLE sales_tab(
             invoice_num  char(6) NOT NULL PRIMARY KEY,
             sales_person     varchar(20),
             order            XMLVarchar);
```

XML-Spalten aktivieren

Zum Speichern eines XML-Dokuments in einer DB2-Datenbank müssen Sie eine Spalte für XML aktivieren. Durch das Aktivieren wird die Spalte für die Indexierung vorbereitet, so daß sie schnell durchsucht werden kann. Sie können eine Spalte mit dem XML Extender-Verwaltungsassistenten aktivieren oder über die DB2-Befehls-Shell. Die Spalte muß den Typ XML haben.

Beim Aktivieren einer XML-Spalte führt der XML Extender folgende Aktionen aus:

- Lesen der DAD und wahlweise Ausführung folgender Aktionen:
  - Prüfen der DAD-Datei anhand der DTD für die DAD-Datei.
  - Abrufen der DTD-ID aus der Tabelle DTD_REF, sofern angegeben.
  - Erstellen von Seitentabellen zum Indexieren der XML-Spalte.
  - Vorbereiten der Spalte für XML-Daten.
- Wahlweise Erstellen einer *Standardsicht* der XML-Tabelle und der Seitentabellen, sofern Seitentabellen definiert sind.
- Angeben eines Wert ROOT ID an, falls kein solcher Wert angegeben war.

Nach dem Aktivieren der XML-Spalte haben Sie folgende Möglichkeiten:

- Erstellen von Indizes zu den Seitentabellen
- Einfügen von XML-Dokumenten in der XML-Spalte
- Abfragen, Aktualisieren oder Durchsuchen der XML-Dokumente in der XML-Spalte

Vorbereitungen

Erstellen Sie eine XML-Tabelle, indem Sie eine DB2-Tabelle mit einer Spalte des XML-UDT erstellen oder ändern.

Verwaltungsassistent verwenden

Führen Sie zum Aktivieren von XML-Spalten die folgenden Schritte aus:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit XML-Spalten arbeiten** im LaunchPad-Fenster an, um die Tasks zu XML Extender-Spalten anzuzeigen. Das Fenster **Eine Task auswählen** wird geöffnet.

3. Klicken Sie **Eine Spalte aktivieren** und anschließend **Weiter** an, um eine vorhandene Tabellenspalte in der Datenbank zu aktivieren.
4. Wählen Sie im Feld **Tabellenname** die Tabelle aus, die die XML-Spalte enthält. Beispiel:

   ```
   SALES_TAB
   ```

5. Wählen Sie die zu aktivierende Spalte im Feld **Spaltenname** aus. Beispiel:

   ```
   ORDER
   ```

   Die Spalte muß bereits vorhanden sein und den Typ XML haben.
6. Geben Sie den DAD-Pfad- und Dateinamen im Feld **DAD-Dateiname** ein, oder klicken Sie **...** an, um nach einer vorhandenen DAD-Datei zu suchen. Beispiel:

   ```
   c:\dxx\samples\dad\getstart.dad
   ```

7. Wahlweise können Sie den Namen eines vorhandenen Tabellenbereichs im Feld **Tabellenbereich** eingeben.

   Der Tabellenbereich enthält Seitentabellen, die vom XML Extender erstellt wurden. Wenn Sie einen Tabellenbereich eingeben, werden die Seitentabellen in dem angegebenen Tabellenbereich erstellt. Geben Sie keinen Tabellenbereich an, werden die Seitentabellen in dem Standardtabellenbereich erstellt.
8. Wahlweise können Sie den Namen der Standardsicht im Feld **Standardsicht** angeben.

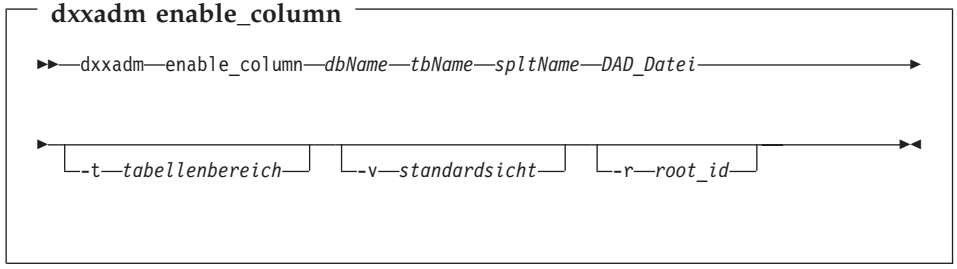
   Sofern sie angegeben ist, wird die Standardsicht beim Aktivieren der Spalte automatisch erstellt und verknüpft die XML-Tabelle und alle dazugehörigen Seitentabellen.
9. Wahlweise können Sie auch den Spaltennamen des Primärschlüssels in der Anwendungstabelle im Feld **Root-ID** angeben. Diese Vorgehensweise wird empfohlen.

   Der XML Extender verwendet den Wert ROOT ID als eindeutige Kennung für die Zuordnung aller Seitentabellen zu der Anwendungstabelle. Fehlt diese Angabe, fügt der XML Extender der Anwendungstabelle die Spalte DXXROOT_ID hinzu und generiert eine Kennung.
10. Klicken Sie **Beenden** an, um die XML-Spalte zu aktivieren, die Seitentabellen zu erstellen und zum LaunchPad-Fenster zurückzukehren.
    - Wenn die Spalte erfolgreich aktiviert wurde, erscheint die Nachricht `Spalte erfolgreich aktiviert`.
    - Wurde die Spalte nicht erfolgreich aktiviert, erscheint eine Fehlernachricht. Korrigieren Sie die Werte des Eingabefelds, bis die Spalte erfolgreich aktiviert wurde.

Von der DB2-Befehls-Shell aus

Geben Sie zum Aktivieren einer XML-Spalte den folgenden Befehl ein:

Syntax:



Parameter:

dbName
Der Name der Datenbank.

tbName
Der Name der Tabelle, die die zu aktivierende Spalte enthält.

spltName
Der Name der zu aktivierenden XML-Spalte.

DAD_Datei
Der Name der Datei, die die Dokumentzugriffsdefinition (DAD) enthält.

tabellenbereich
Ein zuvor erstellter Tabellenbereich, der die vom XML Extender erstellten Seitentabellen enthält. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.

standardsicht
Wahlfrei. Der Name der Standardsicht, die der XML Extender zum Verknüpfen einer Anwendungstabelle und aller zugehörigen Seitentabellen erstellt hat.

root_id Wahlfrei. Der Spaltenname des Primärschlüssels in der Anwendungstabelle und eine eindeutige Kennung, die alle Seitentabellen der Anwendungstabelle zuordnet. Der XML Extender verwendet den Wert *root_id* als eindeutige Kennung für die Zuordnung aller Seitentabellen zu der Anwendungstabelle. Die Angabe der ROOT ID wird empfohlen. Wenn die ROOT ID nicht angegeben ist, fügt der XML Extender die Spalte DXXROOT_ID der Anwendungstabelle hinzu und generiert eine Kennung.

Einschränkung: Wenn die Anwendungstabelle einen Spaltennamen DXXROOT_ID enthält, diese Spalte jedoch nicht den Wert für *root_id* enthält, müssen Sie den Parameter *root_id* angeben; andernfalls tritt ein Fehler auf.

Beispiel: Das folgende Beispiel aktiviert eine Spalte über die DB2-Befehls-Shell. Die DAD-Datei und das XML-Dokument finden Sie in „Anhang B. Beispiele" auf Seite 285.

```
dxxadm enable_column SALES_DB sales_tab order getstart.dad
            -v sales_order_view -r invoice_num
```

In diesem Beispiel wird die Spalte ORDER in der Tabelle SALES_DB.SALE-S_TAB aktiviert. Die DAD-Datei ist `getstart.dad`, die Standardsicht ist sales-_order_view und die ROOT-ID ist INVOICE_NUM.

In diesem Beispiel hat die Tabelle SALES_TAB das folgende Schema:

Spaltenname	INVOICE_NUM	SALES_PERSON	ORDER
Datentyp	CHAR(6)	VARCHAR(20)	XMLVARCHAR

Die folgenden Seitentabellen werden entsprechend der DAD-Spezifikation erstellt:

ORDER_SIDE_TAB:

Spaltenname	ORDER_KEY	CUSTOMER	INVOICE_NUM
Datentyp	INTEGER	VARCHAR(50)	CHAR(6)
Pfadausdruck	`/Order/@key`	`/Order/Customer/Name`	N/A

PART_SIDE_TAB:

Spaltenname	PART_KEY	PRICE	INVOICE_NUM
Datentyp	INTEGER	DOUBLE	CHAR(6)
Pfadausdruck	`/Order/Part/@key`	`/Order/Part/ExtendedPrice`	N/A

SHIP_SIDE_TAB:

Spaltenname	DATE	INVOICE_NUM
Datentyp	DATE	CHAR(6)
Pfadausdruck	`/Order/Part/Shipment/ShipDate`	N/A

Alle Seitentabellen enthalten die Spalte INVOICE_NUM desselben Typs, da die ROOT ID vom Primärschlüssel INVOICE_NUM in der Anwendungstabelle angegeben wurde. Nach dem Aktivieren der Spalte wird der Wert von INVOICE_NUM in die Seitentabellen eingefügt. Die Angabe des Parameters *default_view* beim Aktivieren der XML-Spalte ORDER bewirkt, daß eine Standardsicht sales_order_view erstellt wird. Die Sicht verknüpft die obigen Tabellen mit der folgenden Anweisung:

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                             order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_tab.order_key, order_tab.customer,
            part_tab.part_key, part_tab.price,
            ship_tab.date
     FROM sales_tab, order_tab, part_tab, ship_tab
     WHERE sales_tab.invoice_num = order_tab.invoice_num
                AND sales_tab.invoice_num = part_tab.invoice_num
                AND sales_tab.invoice_num = ship_tab.invoice_num
```

Wenn der Tabellenbereich im Befehl **enable_column** angegeben ist, werden die Seitentabellen in dem angegebenen Tabellenbereich erstellt. Ist der Tabellenbereich nicht angegeben, werden die Seitentabellen in dem Standardtabellenbereich erstellt.

Seitentabellen indexieren

Nachdem Sie eine XML-Spalte aktiviert und Seitentabellen erstellt haben, können Sie die Seitentabellen indexieren. Seitentabellen enthalten die XML-Daten in den Spalten, die Sie beim Erstellen der DAD-Datei angegeben haben. Das Indexieren dieser Tabellen hilft, die Leistung der Abfragen zu XML-Dokumenten zu verbessern.

Vorbereitungen

- Erstellen Sie eine DAD-Datei, die die Seitentabellen für die XML-Dokumentstruktur angibt.
- Aktivieren Sie die XML-Spalte mit der DAD-Datei; dadurch werden die Seitentabellen erstellt.

Der Befehl DB2 CREATE INDEX

Verwenden Sie den Befehl DB2 CREATE INDEX.

Beispiel:

Das folgende Beispiel erstellt Indizes zu vier Seitentabellen:

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)

DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)
```

```
DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)

DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

XML-Spalten inaktivieren

Inaktivieren Sie eine Spalte, wenn Sie eine DAD-Datei für die XML-Spalte aktualisieren oder die XML-Spalte oder die Tabelle, die die Spalte enthält, löschen wollen. Nach dem Inaktivieren der Spalte können Sie die Spalte mit der aktualisierten DAD-Datei erneut aktivieren, die Spalte löschen oder andere Tasks ausführen. Sie können eine Spalte mit dem XML Extender-Verwaltungsassistenten oder über die DB2-Befehls-Shell inaktivieren.

Beim Aktivieren einer XML-Spalte führt der XML Extender folgende Aktionen aus:

- Löschen des Spalteneintrags aus der Tabelle XML_USAGE.
- Freigeben der Seitentabellen, die dieser Spalte zugeordnet sind.

Wichtig: Wenn Sie eine Tabelle mit einer XML-Spalte freigeben, ohne die Spalte zunächst zu inaktivieren, kann der XML Extender keine der XML-Spalte zugeordnete Seitentabellen freigeben, was zu unerwarteten Ergebnissen führen kann.

Vorbereitungen

Stellen Sie sicher, daß die zu inaktivierende XML-Spalte in der aktuellen DB2-Datenbank vorhanden ist.

Verwaltungsassistent verwenden

Führen Sie zum Inaktivieren von XML-Spalten die folgenden Schritte aus:

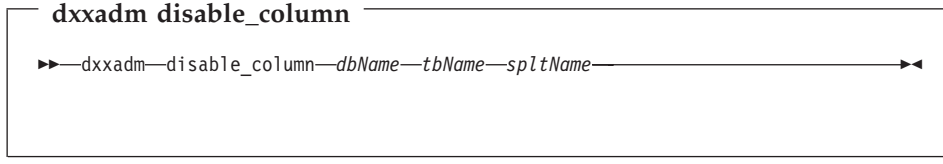
1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit XML-Spalten arbeiten** im LaunchPad-Fenster an, um die Tasks zu XML Extender-Spalten anzuzeigen. Das Fenster **Eine Task auswählen** wird geöffnet.
3. Klicken Sie **Inaktivieren einer Spalte** und anschließend **Weiter** an, um eine vorhandene Tabellenspalte in der Datenbank zu inaktivieren.
4. Wählen Sie im Feld **Tabellenname** die Tabelle aus, die die XML-Spalte enthält.
5. Wählen Sie die zu inaktivierende Spalte im Feld **Spaltenname** aus.
6. Klicken Sie **Beenden** an.
   - Wenn die Spalte erfolgreich inaktiviert wurde, erscheint die Nachricht `Disabled column is successful`.

- Wurde die Spalte nicht erfolgreich inaktiviert, erscheint eine Fehlernachricht. Korrigieren Sie die Werte des Eingabefelds, bis die Spalte erfolgreich inaktiviert wurde.

Von der DB2-Befehls-Shell aus

Geben Sie zum Inaktivieren einer XML-Spalte den folgenden Befehl ein:

Syntax:



Parameter:

dbName
Der Name der Datenbank.

tbName
Der Name der Tabelle, die die zu inaktivierende Spalte enthält.

spltName
Der Name der zu inaktivierenden XML-Spalte.

Beispiel: Das folgende Beispiel inaktiviert eine Spalte über die DB2-Befehls-Shell. Die DAD-Datei und das XML-Dokument finden Sie in „Anhang B. Beispiele" auf Seite 285.

```
dxxadm disable_column SALES_DB sales_tab order
```

In diesem Beispiel wird die Spalte ORDER in der Tabelle SALES_DB.SALES_TAB inaktiviert.

Wenn die Spalte inaktiviert ist, werden die Seitentabellen freigegeben.

Mit XML-Objektgruppen arbeiten

Zum Einstellen der XML-Objektgruppen muß ein Zuordnungsschema erstellt und wahlweise die Objektgruppe mit einem virtuellen Namen aktiviert werden, der die DB2-Tabellen einer DAD-Datei zuordnet.

Das Aktivieren der XML-Objektgruppe ist zwar nicht erforderlich, es ermöglicht jedoch eine höhere Leistung.

DAD-Datei für das Zuordnungsschema erstellen oder editieren

Das Erstellen einer DAD-Datei ist bei Verwendung der XML-Objektgruppen erforderlich. Eine DAD-Datei definiert die Beziehung zwischen den XML-Daten und mehreren relationalen Tabellen. Der XML Extender verwendet die DAD-Datei für folgende Aktionen:

- Zusammensetzen eines XML-Dokuments aus relationalen Daten
- Zerlegen eines XML-Dokuments in relationale Daten

Sie können eine der beiden Methoden zum Zuordnen der Daten zwischen den XML-Tabellen und der DB2-Tabelle verwenden: SQL-Zuordnung und RDB_node-Zuordnung.

SQL-Zuordnung
: Verwendet eine SQL-Anweisung zur Angabe der SQL-Abfrage für Tabellen und Spalten, die die XML-Daten enthalten. Die SQL-Zuordnung kann ausschließlich zum Zusammensetzen von XML-Dokumenten verwendet werden.

RDB_node-Zuordnung
: Verwendet ein spezielles XML Extender-Element, den relationalen Datenbankknoten oder ″RDB_node″, der Tabellen, Spalten, Bedingungen und die Reihenfolge der XML-Daten angibt. Die RDB_node-Zuordnung unterstützt komplexere Zuordnungen, als sie mit SQL-Anweisungen möglich sind. Die RDB_node-Zuordnung kann zum Zusammensetzen und zum Zerlegen von XML-Dokumenten verwendet werden.

Beide Zuordnungsmethoden verwenden das *XPath-Datenmodell*; dieses Modell ist im Abschnitt „Die DAD-Datei” auf Seite 61 beschrieben.

Vorbereitungen

- Ordnen Sie die Beziehung zwischen Ihren DB2-Tabellen und dem XML-Dokument zu. Dieser Schritt umfaßt das Zuordnen der Hierarchie des XML-Dokuments und die Angabe, wie die Daten in dem Dokument einer DB2-Tabelle zugeordnet sind.
- Wenn Sie eine Gültigkeitsprüfung mit den XML-Dokumenten ausführen wollen, fügen Sie die DTD für das zusammenzusetzende bzw. zu zerlegende XML-Dokument in die DTD-Referenztabelle db2xml.DTD_REF ein.

XML-Dokumente mit SQL-Zuordnung zusammensetzen

Verwenden Sie die SQL-Zuordnung, wenn Sie XML-Dokumente zusammensetzen und SQL verwenden wollen.

Verwaltungsassistent verwenden: Führen Sie zum Erstellen einer DAD-Datei mit einer SQL-Zuordnung für XML-Objektgruppen die folgenden Schritte aus:

So erstellen Sie eine DAD-Datei zum Zusammensetzen mit einer SQL-Zuordnung:

Verwenden Sie die SQL-Zuordnung, wenn Sie XML-Dokumente zusammensetzen und eine SQL-Anweisung zum Definieren der Tabelle und der Spalten verwenden wollen, aus dem die Daten in dem XML-Dokument abgeleitet werden.

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit DAD-Dateien arbeiten** im LaunchPad-Fenster an. Das Fenster **Eine DAD angeben** wird angezeigt.
3. Wählen Sie aus, ob Sie eine bestehende DAD-Datei editieren oder eine neue DAD-Datei erstellen wollen.

   So erstellen Sie eine neue DAD-Datei:

   a. Lassen Sie das Feld **Dateiname** leer.
   b. Wählen Sie im Menü **Typ** die Option **SQL-Zuordnung von XML-Objektgruppen** aus.
   c. Klicken Sie **Weiter** an, um das Fenster **Gültigkeitsprüfung auswählen** zu öffnen.

   So editieren Sie eine vorhandene DAD-Datei:

   a. Geben Sie den DAD-Dateinamen im Feld **Dateiname** ein, oder klicken Sie **...** an, um nach einer vorhandenen DAD-Datei zu suchen.
   b. Vergewissern Sie sich, daß der Assistent die angegebene DAD-Datei erkennt.
      - Wenn der Assistent die angegebene DAD-Datei erkennt, können Sie **Weiter** auswählen, und **SQL-Zuordnung von XML-Objektgruppen** wird im Feld **Typ** angezeigt.
      - Wenn der Assistent die angegebene DAD-Datei nicht erkennt, kann **Weiter** nicht ausgewählt werden. Geben Sie entweder den DAD-Dateinamen erneut ein, oder klicken Sie **...** an, um erneut nach vorhandenen DAD-Dateien zu suchen. Korrigieren Sie die Werte des Eingabefelds, bis **Weiter** ausgewählt werden kann.
   c. Klicken Sie **Weiter** an, um das Fenster **Gültigkeitsprüfung auswählen** zu öffnen.

4. Wählen Sie im Fenster **Gültigkeitsprüfung auswählen** aus, ob Ihre XML-Dokumente mit einer DTD geprüft werden sollen.
   - So führen Sie eine Überprüfung durch:
     a. Klicken Sie **Prüfen Sie XML-Dokumente mit der DTD** an.
     b. Wählen Sie die für die Gültigkeitsprüfung zu verwendende DTD im Menü **DTD-ID** aus.

     Wenn Sie keine DTDs in das DTD-Repository für Ihre Datenbank importiert haben, können Sie keine Gültigkeitsprüfung für Ihre XML-Dokumente durchführen.
   - Klicken Sie **Nicht prüfen** an, um fortzufahren, ohne Ihre XML-Dokumente zu überprüfen.
5. Klicken Sie **Weiter** an, um das Fenster **Text angeben** zu öffnen.
6. Geben Sie den Prologna men im Feld **Prolog** ein, um den Prolog des zusammenzusetzenden XML-Dokuments anzugeben.

   ```
   <?xml version="1.0"?>
   ```

   Wenn Sie eine vorhandene DAD editieren, wird der Prolog automatisch im Feld **Prolog** angezeigt.
7. Geben Sie den Dokumenttyp des XML-Dokuments im Feld **Doctype** des Fensters **Text angeben** ein, um auf die DTD für das XML-Dokument zu verweisen. Beispiel:

   ```
   ! DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"
   ```

   Wenn Sie eine vorhandene DAD editieren, wird der Dokumenttyp automatisch im Feld **Doctype** angezeigt.
8. Klicken Sie **Weiter** an, um das Fenster **SQL-Anweisung angeben** zu öffnen.
9. Geben Sie eine gültige SQL-Anweisung SELECT im Feld **SQL-Anweisung** ein. Beispiel:

   ```
   SELECT o.order_key, customer_name, customer_email, p.part_key, color,
   quantity, price, tax, ship_id, date, mode from order_tab o, part_tab p,
   table (select substr(char(timestamp(generate_unique())),16)
   as ship_id, date, mode, part_key from ship_tab) s
       WHERE o.order_key = 1 and
             p.price > 20000 and
             p.order_key = o.order_key and
             s.part_key = p.part_key
       ORDER BY order_key, part_key, ship_id
   ```

   Wenn Sie eine vorhandene DAD editieren, wird die SQL-Anweisung automatisch im Feld **SQL-Anweisung** angezeigt.

10. Klicken Sie **SQL testen** an, um die Gültigkeit der SQL-Anweisung zu testen.
    - Wenn Ihre SQL-Anweisung gültig ist, werden Musterergebnisse im Feld **Beispielergebnisse** angezeigt.
    - Ist Ihre SQL-Anweisung nicht gültig, erscheint eine Fehlernachricht im Feld **Beispielergebnisse**. Die Fehlernachricht fordert Sie auf, Ihre SQL-Anweisung SELECT zu korrigieren und die Aktion erneut zu versuchen.
11. Klicken Sie **Weiter** an, um das Fenster **SQL-Zuordnung** zu öffnen.
12. Wählen Sie einen Element- oder Attributknoten für die Zuordnung aus, indem Sie in dem Feld links vom Fenster **SQL-Zuordnung** darauf klicken.

    Ordnen Sie die Elemente und Attribute in dem XML-Dokument den Element- und Attributknoten zu, die den DB2-Daten entsprechen. Diese Knoten bieten einen Pfad von den XML-Daten zu den DB2-Daten.
    - **So fügen Sie den Root-Knoten hinzu:**
      a. Wählen Sie das Symbol **Root** aus.
      b. Klicken Sie **Neues Element** an, um einen neuen Knoten zu definieren.
      c. Geben Sie im Bereich **Details** für **Knotenart** die Auswahl **Element** an.
      d. Geben Sie den Namen des Knotens der höchsten Ebene im Feld **Knotenname** ein.
      e. Klicken Sie **Hinzufügen** an, um den neuen Knoten zu erstellen.

         Sie haben den Root-Knoten oder das Element erstellt, das allen anderen Element- und Attributknoten in der Hierarchie übergeordnet ist. Sie können diesem Knoten jetzt neue untergeordnete Elemente und Attribute hinzufügen.
    - **So fügen Sie einen untergeordneten Element- oder Attributknoten hinzu:**
      a. Klicken Sie einen übergeordneten Knoten in dem Feld links an, um ein untergeordnetes Element oder Attribut hinzuzufügen.

         Wenn Sie keinen übergeordneten Knoten ausgewählt haben, kann **Neues Element** nicht ausgewählt werden.
      b. Klicken Sie **Neues Element** an.

c. Wählen Sie die Knotenart im Menü **Knotenart** im Bereich **Details** aus.

   Das Menü **Knotenart** zeigt nur die Knotenarten an, die an dieser Stelle in der Zuordnung gültig sind:

   Element
   Steht für ein in der DTD definiertes XML-Element, das dem XML-Dokument zugeordnet wurde. Wird zum Zuordnen des XML-Elements zu einer Spalte in einer DB2-Tabelle verwendet. Ein Elementknoten kann Attributknoten, untergeordnete Elementknoten oder Textknoten haben. Einem Knoten auf der untersten Ebene der Baumstruktursicht ist ein Textknoten und ein Spaltenname zugeordnet.

   Attribut
   Steht für ein in der DTD definiertes XML-Attribut, das dem XML-Dokument zugeordnet wurde. Wird zum Zuordnen des XML-Attributs zu einer Spalte in einer DB2-Tabelle verwendet. Einem Attributknoten kann ein Textknoten und ein Spaltenname in der Baumstruktursicht zugeordnet sein.

   Text Gibt den Textinhalt für einen Element- oder Attributknoten an, dessen Inhalt einer relationalen Tabelle zugeordnet werden soll. Einem Textknoten ist ein Spaltenname in der Baumstruktursicht zugeordnet.

   Tabelle
   Gibt den Tabellennamen für einen Element- oder Attributwert an, der einer relationalen Tabelle zugeordnet werden soll.

   Spalte Gibt den Spaltennamen für einen Element- oder Attributwert an, der einer relationalen Tabelle zugeordnet werden soll.

   Bedingung
   Gibt eine Bedingung für die Spalte an.

d. Geben Sie den Knotennamen im Feld **Knotenname** im Bereich **Details** ein. Beispiel:

   ```
   Order
   ```

e. Wenn Sie **Attribut**, **Element** oder **Text** für ein Element der untersten Ebene als Knotenart angegeben haben, wählen Sie eine Spalte im Feld **Spalte** im Bereich **Details** aus. Beispiel:

   ```
   Customer_Name
   ```

Einschränkung: Mit dem Verwaltungsassistenten können keine neuen Spalten erstellt werden. Wenn Sie **Spalte** als Knotenart auswählen, können Sie nur eine Spalte auswählen, die in Ihrer DB2-Datenbank bereits vorhanden ist.

f. Klicken Sie **Hinzufügen** an, um den neuen Knoten hinzuzufügen.

Sie können einen Knoten später ändern, indem Sie ihn in dem Feld links anklicken und alle gewünschten Änderungen in dem Bereich **Details** vornehmen. Klicken Sie **Ändern** an, um das Element zu aktualisieren.

Sie können dem Knoten auch untergeordnete Elemente oder Attribute hinzufügen, indem Sie den Knoten hervorheben und den Hinzufügeprozeß wiederholen.

g. Fahren Sie mit dem Editieren der SQL-Zuordnung fort, oder klicken Sie **Weiter** an, um das Fenster **Eine DAD angeben** zu öffnen.

- **So entfernen Sie einen Knoten:**
  a. Klicken Sie einen Knoten in dem Feld auf der linken Seite an.
  b. Klicken Sie **Entfernen** an.
  c. Fahren Sie mit dem Editieren der SQL-Zuordnung fort, oder klicken Sie **Weiter** an, um das Fenster **Eine DAD angeben** zu öffnen.

Wenn Sie einen Knoten der untersten Ebene entfernen, wird ein anderes Element zum Knoten der untersten Ebene; eventuell muß dann für dieses Element ein Spaltenname definiert werden.

13. Geben Sie einen Ausgabedateinamen für die geänderte DAD-Datei im Feld **Dateiname** des Fensters **Eine DAD angeben** ein.
14. Klicken Sie **Beenden** an, um zum LaunchPad-Fenster zurückzukehren.

Von der DB2-Befehls-Shell aus: Verwenden Sie die SQL-Zuordnung, wenn Sie XML-Dokumente zusammensetzen und SQL verwenden wollen.

Die DAD-Datei ist eine XML-Datei, die Sie mit einem beliebigen Texteditor erstellen können. Die folgenden Schritte zeigen Teile des Muster-Anhangs, „Dokumentzugriffsdefinitionsdateien" auf Seite 286. In diesen Beispielen finden Sie ausführliche Informationen und den jeweiligen Kontext.

1. Öffnen Sie einen Texteditor.
2. Erstellen Sie die DAD-Kopfzeilen:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "pfad\dad.dtd"> --> Pfad- und
      Dateiname der DTD für die DAD
```

3. Fügen Sie die Befehle <DAD></DAD> ein.
4. Geben Sie nach dem Befehl <DAD> die DTD-ID an, die die DAD-Datei mit der DTD des XML-Dokuments verbindet.

```
<dtdid>pfad\dtd_name.dtd>
--> Pfad- und Dateiname
       der DTD für Ihre Anwendung
```

5. Geben Sie an, ob eine Gültigkeitsprüfung durchgeführt werden soll (ob über eine DTD sichergestellt werden soll, daß das XML-Dokument ein gültiges XML-Dokument ist). Beispiel:

   ```
   <validation>NO</validation>  --> Geben Sie YES oder NO an
   ```

6. Verwenden Sie das Element <Xcollection> zum Definieren der Zugriffs- und Speichermethode als XML-Objektgruppe. Die Zugriffs- und Speichermethode geben an, daß das XML-Dokument einen Inhalt haben wird, der aus den in DB2-Tabellen gespeicherten Daten abgeleitet wurde.

   ```
   <Xcollection>
   </Xcollection>
   ```

7. Geben Sie eine oder mehrere SQL-Anweisungen an, um Daten aus DB2-Tabellen abzufragen oder in DB2-Tabellen einzufügen. Richtlinien hierzu finden Sie im Abschnitt „Voraussetzung für das Zuordnungsschema" auf Seite 66. Sie können beispielsweise eine einzelne SQL-Abfrage wie die folgende angeben:

   ```
    <SQL_stmt>
      SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
      price, tax, ship_id, date, mode from order_tab o, part_tab p,
      table (select substr(char(timestamp(generate_unique())),16)
        as ship_id, date, mode, part_key from ship_tab) s
            WHERE o.order_key = 1 and
                  p.price > 20000 and
                  p.order_key = o.order_key and
                  s.part_key = p.part_key
            ORDER BY order_key, part_key, ship_id
   </SQL_stmt>
   ```

8. Fügen Sie die folgenden Prologinformationen hinzu:

   ```
   <prolog>?xml version="1.0"?</prolog>
   ```

   Dieser Text ist in genau dieser Form erforderlich.

9. Fügen Sie die Befehle <doctype></doctype> hinzu. Beispiel:

   ```
   <doctype>! DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
   ```

10. Definieren Sie den Root-Knoten mit den Befehlen <root_node></root_node>. Innerhalb des Root-Knotens geben Sie die Elemente und Attribute an, aus denen das XML-Dokument besteht.
11. Ordnen Sie die Elemente und Attribute in dem XML-Dokument den Element- und Attributknoten zu, die den DB2-Daten entsprechen. Diese Knoten bieten einen Pfad von den XML-Daten zu den DB2-Daten.
    a. Definieren Sie einen <element_node> für jedes Element in Ihrem XML-Dokument, das einer Spalte in einer DB2-Tabelle zugeordnet ist.

       ```
       <element_node name="name"></element_node>
       ```

Ein element_node kann folgende Knoten enthalten.

- attribute_node
- untergeordneten element_node
- text_node

b. Definieren Sie einen <attribute_node> für jedes Attribut in Ihrem XML-Dokument, das einer Spalte in einer DB2-Tabelle zugeordnet ist. Beispiele zu DTDs für die SQL-Zuordnung finden Sie am Anfang dieses Abschnitts; die DTD für die DAD-Datei mit der vollständigen Syntax für die DAD-Datei ist in „Anhang A. DTD für die DAD-Datei" auf Seite 277 beschrieben.

Sie benötigen beispielsweise einen Attributschlüssel für ein Element <Order>. Der Wert des Schlüssels ist in einer Spalte PART_KEY gespeichert.

DAD-Datei: Erstellen Sie in der DAD-Datei einen Attributknoten für den Schlüssel, und geben Sie die Tabelle an, in der der Wert 1 gespeichert werden soll.

```
<attribute_node name="key">
    <column name="part_key"/>
</attribute_node>
```

Zusammengesetztes XML-Dokument: Der Wert des Schlüssels stammt aus der Spalte PART_KEY.

```
<Order key="1">
```

12. Erstellen Sie einen <text_node> für jedes Element oder Attribut, das einen Inhalt hat, der aus einer DB2-Tabelle abgeleitet wird. Der Textknoten hat ein Element <column>, das angibt, aus welcher Spalte der Inhalt bereitgestellt wird.

Sie können beispielsweise ein XML-Element <Tax> mit einem Wert haben, der aus einer Spalte mit dem Namen TAX stammt:

DAD-Element:

```
<element_node name="Tax">
   <text_node>
   <column name="tax"/>
   </text_node>
</element_node>
```

Der Spaltenname muß in der SQL-Abweisung am Anfang der DAD-Datei stehen.

Zusammengesetztes XML-Dokument:

```
<Tax>0.02</Tax>
```

Der Wert `0.02` wird aus der Spalte TAX abgeleitet.

13. Stellen Sie sicher, daß Sie einen Endbefehl </root_node> nach dem letzten Befehl </element_node> verwenden.
14. Stellen Sie sicher, daß Sie einen Endbefehl </Xcollection> nach dem Befehl </root_node> verwenden.
15. Stellen Sie sicher, daß Sie einen Endbefehl </DAD> nach dem Befehl </Xcollection> verwenden.

XML-Dokumente mit RDB_node-Zuordnung zusammensetzen

Verwenden Sie die RDB_node-Zuordnung zum Zusammensetzen von XML-Dokumenten mit einer XML-ähnlichen Struktur.

Diese Methode verwendet <RDB_node> zur Angabe von DB2-Tabellen, Spalten und Bedingungen für einen Element- oder Attributknoten. Der <RDB_node> verwendet die folgenden Elemente:

- <table>: definiert die Tabelle, die dem Element entspricht
- <column>: definiert die Spalte, die das entsprechende Element enthält.
- <condition>: definiert wahlweise eine Bedingung zu der Spalte

Die im <RDB_node> verwendeten untergeordneten Elemente hängen vom Kontext des Knotens ab; es gelten dabei folgende Regeln:

Für die Knotenart:	**RDB untergeordnetes Element wird verwendet:**		
	Tabelle	**Spalte**	**Bedingung[1]**
Stammelement	J	N	J
Attribut	J	J	wahlfrei
Text	J	J	wahlfrei

(1) Erforderlich bei mehreren Tabellen

Verwaltungsassistent verwenden: So erstellen Sie eine DAD zum Zusammensetzen mit der RDB_node-Zuordnung:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit DAD-Dateien arbeiten** im LaunchPad-Fenster an. Das Fenster **Eine DAD angeben** wird angezeigt.
3. Wählen Sie aus, ob Sie eine bestehende DAD-Datei editieren oder eine neue DAD erstellen wollen.

   So editieren Sie eine vorhandene DAD:

   a. Geben Sie den DAD-Dateinamen im Feld **Dateiname** ein, oder klicken Sie **...** an, um nach einer vorhandenen DAD zu suchen.

   b. Vergewissern Sie sich, daß der Assistent die angegebene DAD-Datei erkennt.

- Wenn der Assistent die angegebene DAD-Datei erkennt, können Sie **Weiter** auswählen, und die RDB_node-Zuordnung "XML-Objektgruppe" wird im Feld **Typ** angezeigt.
- Wenn der Assistent die angegebene DAD-Datei nicht erkennt, kann **Weiter** nicht ausgewählt werden. Geben Sie entweder den DAD-Dateinamen im Feld **Dateiname** erneut ein, oder klicken Sie **...** an, um erneut nach vorhandenen DAD-Dateien zu suchen. Fahren Sie mit diesen Schritten fort, bis **Weiter** ausgewählt werden kann.

c. Klicken Sie **Weiter** an, um das Fenster **Gültigkeitsprüfung auswählen** zu öffnen.

So erstellen Sie eine neue DAD:

a. Lassen Sie das Feld **Dateiname** leer.
b. Wählen Sie die RDB_node-Zuordnung "XML-Objektgruppe" im Menü **Typ** aus.
c. Klicken Sie **Weiter** an, um das Fenster **Gültigkeitsprüfung auswählen** zu öffnen.

4. Wählen Sie im Fenster **Gültigkeitsprüfung auswählen** aus, ob Ihre XML-Dokumente mit einer DTD geprüft werden sollen.
   - So führen Sie eine Überprüfung durch:
     a. Klicken Sie **Prüfen Sie XML-Dokumente mit der DTD** an.
     b. Wählen Sie die für die Gültigkeitsprüfung zu verwendende DTD im Menü **DTD-ID** aus.

     Wenn Sie keine DTDs in das DTD-Repository für Ihre Datenbank importiert haben, können Sie keine Gültigkeitsprüfung für Ihre XML-Dokumente durchführen.
   - Klicken Sie **Nicht prüfen** an, um fortzufahren, ohne Ihre XML-Dokumente zu überprüfen.
5. Klicken Sie **Weiter** an, um das Fenster **Text angeben** zu öffnen.
6. Geben Sie den Prologna men im Feld **Prolog** des Fensters **Text angeben** ein.

   ```
   <?xml version="1.0"?>
   ```

   Wenn Sie eine vorhandene DAD editieren, wird der Prolog automatisch im Feld **Prolog** angezeigt.
7. Geben Sie den Dokumenttyp des XML-Dokuments im Feld **Doctype** des Fensters **Text angeben** ein.

   Wenn Sie eine vorhandene DAD editieren, wird der Dokumenttyp automatisch im Feld **Doctype** angezeigt.
8. Klicken Sie **Weiter** an, um das Fenster **RDB-Zuordnung** zu öffnen.

9. Wählen Sie einen Element- oder Attributknoten für die Zuordnung aus, indem Sie in dem Feld links vom Fenster **RDB-Zuordnung** darauf klicken.

   Ordnen Sie die Elemente und Attribute in dem XML-Dokument den Element- und Attributknoten zu, die den DB2-Daten entsprechen. Diese Knoten bieten einen Pfad von den XML-Daten zu den DB2-Daten.

10. **So fügen Sie den Root-Knoten hinzu:**
    a. Wählen Sie das Symbol **Root** aus.
    b. Klicken Sie **Neues Element** an, um einen neuen Knoten zu definieren.
    c. Geben Sie im Bereich **Details** für **Knotenart** die Auswahl **Element** an.
    d. Geben Sie den Namen des Knotens der höchsten Ebene im Feld **Knotenname** ein.
    e. Klicken Sie **Hinzufügen** an, um den neuen Knoten zu erstellen.

       Sie haben den Root-Knoten oder das Element erstellt, das allen anderen Element- und Attributknoten in der Hierarchie übergeordnet ist. Der Root-Knoten hat untergeordnete Tabellenelemente und eine Verknüpfungsbedingung.
    f. Fügen Sie Tabellenknoten für jede Tabelle hinzu, die Teil der Objektgruppe ist.
       1) Heben Sie den Root-Knotennamen hervor, und wählen Sie **Neues Element** aus.
       2) Geben Sie im Bereich **Details** für **Knotenart** die Auswahl **Tabelle** an.
       3) Wählen Sie den Namen der Tabelle unter **Tabellenname** aus. Die Tabelle muß bereits vorhanden sein.
       4) Klicken Sie **Hinzufügen** an, um den Tabellenknoten hinzuzufügen.
       5) Wiederholen Sie diese Schritte für jede Tabelle.
    g. Fügen Sie eine Verknüpfungsbedingung für die Tabellenknoten hinzu.
       1) Heben Sie den Root-Knotennamen hervor, und wählen Sie **Neues Element** aus.
       2) Geben Sie im Bereich **Details** für **Knotenart** die Auswahl **Bedingung** an.
       3) Geben Sie im Feld **Bedingung** die Verknüpfungsbedingung mit der folgenden Syntax ein:

          ```
          table_name.table_column = table_name.table_column AND
          table_name.table_column = table_name.table_column ...
          ```

       4) Klicken Sie **Hinzufügen** an, um die Bedingung hinzuzufügen.

11. **So fügen Sie einen Element- oder Attributknoten hinzu:**
    a. Klicken Sie einen übergeordneten Knoten in dem Feld links an, um ein untergeordnetes Element oder Attribut hinzuzufügen.

b. Klicken Sie **Neues Element** an. Wenn Sie keinen übergeordneten Knoten ausgewählt haben, kann **Neues Element** nicht ausgewählt werden.

c. Wählen Sie eine Knotenart im Menü **Knotenart** im Bereich **Details** aus.

   Das Menü **Knotenart** zeigt nur die Knotenarten an, die an dieser Stelle in der Zuordnung gültig sind. **Element** oder **Attribut**.

d. Geben Sie einen Knotennamen im Feld **Knotenname** an.

e. Klicken Sie **Hinzufügen** an, um den neuen Knoten hinzuzufügen.

f. **So ordnen Sie den Inhalt eines Element- oder Attributknotens einer relationalen Tabelle zu:**

   1) Geben Sie einen Textknoten an.
      a) Klicken Sie den übergeordneten Knoten an.
      b) Klicken Sie **Neues Element** an.
      c) Wählen Sie im Feld **Knotenart** die Option **Text** aus.
      d) Wählen Sie **Hinzufügen** aus, um den Knoten auszuwählen.
   2) Fügen Sie einen Tabellenknoten hinzu.
      a) Wählen Sie den eben erstellten Textknoten aus, und klicken Sie **Neues Element** an.
      b) Wählen Sie im Feld **Knotenart** die Option **Tabelle** aus, und geben Sie einen Tabellennamen für das Element ein.
      c) Klicken Sie **Hinzufügen** an, um den Knoten hinzuzufügen.
   3) Fügen Sie einen Spaltenknoten hinzu.
      a) Wählen Sie den Textknoten erneut aus, und klicken Sie **Neues Element** an.
      b) Wählen Sie im Feld **Knotenart** die Option **Spalte** aus, und geben Sie einen Spaltennamen für das Element ein.
      c) Klicken Sie **Hinzufügen** an, um den Knoten hinzuzufügen.

      Einschränkung: Mit dem Verwaltungsassistenten können keine neuen Spalten erstellt werden. Wenn Sie **Spalte** als Knotenart auswählen, können Sie nur eine Spalte auswählen, die in Ihrer DB2-Datenbank bereits vorhanden ist.
   4) Fügen Sie wahlweise eine Bedingung für die Spalte hinzu.
      a) Wählen Sie den Textknoten erneut aus, und klicken Sie **Neues Element** an.
      b) Wählen Sie im Feld **Knotenart** die Option **Bedingung** und die Bedingung mit der folgenden Syntax aus:

         operator LIKE|<|>|= *wert*
      c) Klicken Sie **Hinzufügen** an, um den Knoten hinzuzufügen.

g. Fahren Sie mit dem Editieren der RDB-Zuordnung fort, oder klicken Sie **Weiter** an, um das Fenster **Eine DAD angeben** zu öffnen.

12. **So entfernen Sie einen Knoten:**
    a. Klicken Sie einen Knoten in dem Feld auf der linken Seite an.
    b. Klicken Sie **Entfernen** an.
    c. Fahren Sie mit dem Editieren der RDB_node-Zuordnung fort, oder klicken Sie **Weiter** an, um das Fenster **Eine DAD angeben** zu öffnen.
13. Geben Sie einen Ausgabedateinamen für die geänderte DAD im Feld **Dateiname** des Fensters **Eine DAD angeben** ein.
14. Klicken Sie **Beenden** an, um den Knoten zu entfernen und zum LaunchPad-Fenster zurückzukehren.

Von der DB2-Befehls-Shell aus: Die DAD-Datei ist eine XML-Datei, die Sie mit einem beliebigen Texteditor erstellen können. Die folgenden Schritte zeigen Teile des Muster-Anhangs, „Dokumentzugriffsdefinitionsdateien" auf Seite 286. In diesen Beispielen finden Sie ausführliche Informationen und den jeweiligen Kontext.

1. Öffnen Sie einen Texteditor.
2. Erstellen Sie die DAD-Kopfzeilen:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "pfad\dad.dtd"> --> Pfad- und
      Dateiname der DTD für die DAD
```

3. Fügen Sie die Befehle <DAD></DAD> ein.
4. Geben Sie nach dem Befehl <DAD> die DTD-ID an, die die DAD-Datei mit der DTD des XML-Dokuments verbindet.

```
<dtdid>pfad\dtd_name.dtd> --> Pfad- und Dateiname der DTD
             für Ihre Anwendung
```

5. Geben Sie an, ob eine Gültigkeitsprüfung durchgeführt werden soll (ob über eine DTD sichergestellt werden soll, daß das XML-Dokument ein gültiges XML-Dokument ist). Beispiel:

```
<validation>NO</validation>  --> Geben Sie YES oder NO an
```

6. Verwenden Sie das Element <Xcollection> zum Definieren der Zugriffs- und Speichermethode als XML-Objektgruppe. Die Zugriffs- und Speichermethode geben an, daß die XML-Daten in einer Objektgruppe von DB2-Tabellen gespeichert sind.

```
<Xcollection>
</Xcollection>
```

7. Fügen Sie die folgenden Prologinformationen hinzu:

```
<prolog>?xml version="1.0"?</prolog>
```

   Dieser Text ist in genau dieser Form erforderlich.
8. Fügen Sie die Befehle <doctype></doctype> hinzu.

Beispiel:

```
<doctype>! DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. Definieren Sie den Root-Knoten über den <root_node>. Innerhalb des Root-Knotens geben Sie die Elemente und Attribute an, aus denen das XML-Dokument besteht.
10. Ordnen Sie die Elemente und Attribute in dem XML-Dokument den Element- und Attributknoten zu, die den DB2-Daten entsprechen. Diese Knoten bieten einen Pfad von den XML-Daten zu den DB2-Daten.
    a. Definieren Sie einen Root-Knoten element_node. Dieser Elementknoten enthält:
       - Einen RDB_node, der Tabellenknoten angibt mit einer Verknüpfungsbedingung zur Angabe der Objektgruppe
       - Untergeordnete Elemente
       - Attribute

       So geben Sie die Tabellenknoten und die Bedingung an:

       1) Erstellen Sie ein RDB_node-Element. Beispiel:

          ```
          <RDB_node>
          </RDB_node>
          ```

       2) Definieren Sie einen <table_node> für jede Tabelle mit Daten, die in das XML-Dokument einbezogen werden sollen. Wenn Sie beispielsweise drei Tabellen ORDER_TAB, PART_TAB und SHIP_TAB haben, die Spaltendaten für das Dokument enthalten, erstellen Sie für jede der Tabellen einen Tabellenknoten. Beispiel:

          ```
          <RDB_node>
          <table name="ORDER_TAB">
          <table name="PART_TAB">
          <table name="SHIP_TAB"></RDB_node>
          ```

       3) Geben Sie wahlweise eine Schlüsselspalte für jede Tabelle an, wenn Sie vorhaben, diese Objektgruppe zu aktivieren. Das Schlüsselattribut ist normalerweise zum Zusammensetzen nicht erforderlich; wenn Sie jedoch eine Objektgruppe aktivieren, muß die verwendete DAD-Datei sowohl das Zusammensetzen wie auch das Zerlegen unterstützen. Beispiel:

          ```
          <RDB_node>
          <table name="ORDER_TAB" key="order_key">
          <table name="PART_TAB" key="part_key">
          <table name="SHIP_TAB" key="date mode">
          </RDB_node>
          ```

       4) Definieren Sie eine Verknüpfungsbedingung für die Tabellenknoten in der Objektgruppe.

          Die Syntax lautet:

          ```
          ausdruck = ausdruck AND
          ausdruck = ausdruck
          ```

Beispiel:

```
<RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
<condition>
   order_tab.order_key = part_tab.order_key AND
   part_tab.part_key = ship_tab.part_key
</condition>
</RDB_node>
```

b. Definieren Sie einen Befehl <element_node> für jedes Element in Ihrem XML-Dokument, das einer Spalte in einer DB2-Tabelle zugeordnet ist. Beispiel:

```
<element_node name="name">
</element_node>
```

Ein Elementknoten kann einen der folgenden Elementtypen enthalten:

- <text_node>: zur Angabe, daß das Element einen Inhalt für eine DB2-Tabelle hat; das Element enthält keine untergeordneten Elemente
- <attribute_node>: zur Angabe eines Attributs. Attributknoten werden im nächsten Schritt definiert.

Der text_node enthält einen <RDB_node> für die Zuordnung von Inhalt zu einer DB2-Tabelle und einem Spaltennamen.

RDB_nodes werden für Elemente auf der untersten Ebene verwendet, die einen Inhalt haben, der einer DB2-Tabelle zugeordnet werden soll. Ein RDB_node hat die folgenden untergeordneten Elemente.

- <table>: definiert die Tabelle, die dem Element entspricht
- <column>: definiert die Spalte, die das entsprechende Element enthält und gibt den Spaltentyp mit dem Typattribut an.
- <condition>: definiert wahlweise eine Bedingung zu der Spalte

Sie können beispielsweise ein XML-Element <Tax> haben, das einer Spalte mit dem Namen TAX zugeordnet ist:

XML-Dokument:

```
<Tax>0.02</Tax>
```

In diesem Fall soll der Wert `0.02` ein Wert in der Spalte TAX sein.

```
<element_node name="Tax">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
```

```
        <column name="tax"/>
      </RDB_node>
   </text_node>
</element_node>
```

In diesem Beispiel gibt <RDB_node> an, daß der Wert des Elements <Tax> ein Textwert ist und daß die Daten in der Tabelle PART_TAB in der Spalte TAX gespeichert werden. Informationen zur RDB_node-Zuordnung finden Sie in den Beispiel-DAD-Dateien in „Dokumentzugriffsdefinitionsdateien" auf Seite 286 sowie in der DTD für die DAD-Datei in „Anhang A. DTD für die DAD-Datei" auf Seite 277; hier ist auch die vollständige Syntax für die DAD-Datei beschrieben.

c. Fügen Sie wahlweise jedem Element <column> ein Typattribut hinzu, wenn Sie vorhaben, diese Objektgruppe zu aktivieren. Das Typattribut ist normalerweise zum Zusammensetzen nicht erforderlich; wenn Sie jedoch eine Objektgruppe aktivieren, muß die verwendete DAD-Datei sowohl das Zusammensetzen wie auch das Zerlegen unterstützen. Beispiel:

```
<column name="tax" type="real"/>
```

d. Definieren Sie einen <attribute_node> für jedes Attribut in Ihrem XML-Dokument, das einer Spalte in einer DB2-Tabelle zugeordnet ist. Beispiel:

```
<attribute_node name="key">
</attribute_node>
```

Der attribute_node enthält einen <RDB_node> für die Zuordnung des Attributwerts zu einer DB2-Tabelle und Spalte. Ein <RDB_node> hat die folgenden untergeordneten Elemente.

- <table>: definiert die Tabelle, die dem Element entspricht
- <column>: definiert die Spalte, die das entsprechende Element enthält.
- <condition>: definiert wahlweise eine Bedingung zu der Spalte

Sie wollen beispielsweise einen Attribut`schlüssel` für ein Element <Order> haben. Der Wert des Schlüssels muß in einer Spalte PART_KEY gespeichert werden. Erstellen Sie in der DAD-Datei einen <Attributknoten> für den Schlüssel, und geben Sie die Tabelle an, in der der Wert gespeichert werden soll.

DAD-Datei

```
<attribute_node name="key">
  <RDB_node>
    <table name="part_tab">
    <column name="part_key"/>
  <RDB_node>
</attribute_node>
```

Zusammengesetztes XML-Dokument:

```
<Order key="1">
```

11. Stellen Sie sicher, daß Sie einen Endbefehl </root_node> nach dem letzten Befehl </element_node> verwenden.
12. Stellen Sie sicher, daß Sie einen Endbefehl </Xcollection> nach dem Befehl </root_node> verwenden.
13. Stellen Sie sicher, daß Sie einen Endbefehl </DAD> nach dem Befehl </Xcollection> verwenden.

Angeben einer Formatvorlage für das XML-Dokument

Beim Erstellen von Dokumenten unterstützt der XML Extender auch Verarbeitungsanweisungen für Formatvorlagen über das Element <stylesheet>. Die Verarbeitungsanweisungen müssen sich innerhalb des Root-Elements <Xcollection> befinden, das mit <doctype> und <prolog> für die XML-Dokumentstruktur definiert ist. Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
<SQL_stmt>
...
</SQL_stmt>
<Xcollection>...
<prolog>...</prolog>
<doctype>...</doctype>
<stylesheet>?xml-stylesheet type="text/css" href="order.css"?</stylesheet>
<root_node>...</root_node>
...
</Xcollection>
...
</DAD>
```

XML-Dokumente mit RDB_node-Zuordnung zerlegen

Verwenden Sie die RDB_node-Zuordnung zum Zerlegen von XML-Dokumenten. Diese Methode verwendet den <RDB_node> zur Angabe von DB2-Tabellen, Spalten und Bedingungen für einen Element- oder Attributknoten. Der <RDB_node> verwendet die folgenden Elemente:

- <table>: definiert die Tabelle, die dem Element entspricht
- <column>: definiert die Spalte, die das entsprechende Element enthält.
- <condition>: definiert wahlweise eine Bedingung zu der Spalte

Die im <RDB_node> verwendeten untergeordneten Elemente hängen vom Kontext des Knotens ab; es gelten dabei folgende Regeln:

Für die Knotenart:	RDB untergeordnetes Element wird verwendet:		
	Tabelle	**Spalte**	**Bedingung[1]**
Stammelement	J	N	J
Attribut	J	J	wahlfrei
Text	J	J	wahlfrei
(1) Erforderlich bei mehreren Tabellen			

Verwaltungsassistent verwenden: So erstellen Sie eine DAD zum Zerlegen:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit DAD-Dateien arbeiten** im LaunchPad-Fenster an. Das Fenster **Eine DAD angeben** wird angezeigt.
3. Wählen Sie aus, ob Sie eine bestehende DAD-Datei editieren oder eine neue DAD erstellen wollen.

   So editieren Sie eine vorhandene DAD:

   a. Geben Sie den DAD-Dateinamen im Feld **Dateiname** ein, oder klicken Sie **...** an, um nach einer vorhandenen DAD zu suchen.

   b. Vergewissern Sie sich, daß der Assistent die angegebene DAD-Datei erkennt.

      - Wenn der Assistent die angegebene DAD-Datei erkennt, können Sie **Weiter** auswählen, und die RDB_node-Zuordnung "XML-Objektgruppe" wird im Feld **Typ** angezeigt.
      - Wenn der Assistent die angegebene DAD-Datei nicht erkennt, kann **Weiter** nicht ausgewählt werden. Geben Sie entweder den DAD-Dateinamen im Feld **Dateiname** erneut ein, oder klicken Sie **...** an, um erneut nach vorhandenen DAD-Dateien zu suchen. Fahren Sie mit diesen Schritten fort, bis **Weiter** ausgewählt werden kann.

   c. Klicken Sie **Weiter** an, um das Fenster **Gültigkeitsprüfung auswählen** zu öffnen.

   So erstellen Sie eine neue DAD:

   a. Lassen Sie das Feld **Dateiname** leer.

   b. Wählen Sie die RDB_node-Zuordnung "XML-Objektgruppe" im Menü**Typ** aus.

   c. Klicken Sie **Weiter** an, um das Fenster **Gültigkeitsprüfung auswählen** zu öffnen.

4. Wählen Sie im Fenster **Gültigkeitsprüfung auswählen** aus, ob Ihre XML-Dokumente mit einer DTD geprüft werden sollen.
   - So führen Sie eine Überprüfung durch:
     a. Klicken Sie **Prüfen Sie XML-Dokumente mit der DTD** an.
     b. Wählen Sie die für die Gültigkeitsprüfung zu verwendende DTD im Menü **DTD-ID** aus.

     Wenn Sie keine DTDs in das DTD-Repository für Ihre Datenbank importiert haben, können Sie keine Gültigkeitsprüfung für Ihre XML-Dokumente durchführen.
   - Klicken Sie **Nicht prüfen** an, um fortzufahren, ohne Ihre XML-Dokumente zu überprüfen.
5. Klicken Sie **Weiter** an, um das Fenster **Text angeben** zu öffnen.
6. Wenn Sie nur ein XML-Dokument zerlegen, ignorieren Sie das Feld **Prolog**. Wenn Sie die DAD-Datei zum Zusammensetzen und Zerlegen verwenden, geben Sie den Prologna men im Feld **Prolog** des Fensters **Text angeben** ein. Der Prolog ist nicht erforderlich, wenn Sie XML-Dokumente in DB2-Daten zerlegen.

   ```
   <?xml version="1.0"?>
   ```

   Wenn Sie eine vorhandene DAD editieren, wird der Prolog automatisch im Feld **Prolog** angezeigt.
7. Wenn Sie nur ein XML-Dokument zerlegen, ignorieren Sie das Feld **Doctype**. Wenn Sie die DAD-Datei zum Zusammensetzen und Zerlegen verwenden, geben Sie den Dokumenttyp des XML-Dokuments im Feld **Doctype** ein.

   Wenn Sie eine vorhandene DAD editieren, wird der Dokumenttyp automatisch im Feld **Doctype** angezeigt.
8. Klicken Sie **Weiter** an, um das Fenster **RDB-Zuordnung** zu öffnen.
9. Wählen Sie einen Element- oder Attributknoten für die Zuordnung aus, indem Sie in dem Feld links vom Fenster **RDB-Zuordnung** darauf klicken.

   Ordnen Sie die Elemente und Attribute in dem XML-Dokument den Element- und Attributknoten zu, die den DB2-Daten entsprechen. Diese Knoten bieten einen Pfad von den XML-Daten zu den DB2-Daten.
10. **So fügen Sie den Root-Knoten hinzu:**
    a. Wählen Sie das Symbol **Root** aus.
    b. Klicken Sie **Neues Element** an, um einen neuen Knoten zu definieren.
    c. Geben Sie im Bereich **Details** für **Knotenart** die Auswahl **Element** an.
    d. Geben Sie den Namen des Knotens der höchsten Ebene im Feld **Knotenname** ein.
    e. Klicken Sie **Hinzufügen** an, um den neuen Knoten zu erstellen.

Sie haben den Root-Knoten oder das Element erstellt, das allen anderen Element- und Attributknoten in der Hierarchie übergeordnet ist. Der Root-Knoten hat untergeordnete Tabellenelemente und eine Verknüpfungsbedingung.

f. Fügen Sie Tabellenknoten für jede Tabelle hinzu, die Teil der Objektgruppe ist.
   1) Heben Sie den Root-Knotennamen hervor, und wählen Sie **Neues Element** aus.
   2) Geben Sie im Bereich **Details** für **Knotenart** die Auswahl **Tabelle** an.
   3) Wählen Sie den Namen der Tabelle unter **Tabellenname** aus. Die Tabelle muß bereits vorhanden sein.
   4) Geben Sie eine Schlüsselspalte für die Tabelle im Feld **Tabellenschlüssel** ein.
   5) Klicken Sie **Hinzufügen** an, um den Tabellenknoten hinzuzufügen.
   6) Wiederholen Sie diese Schritte für jede Tabelle.

g. Fügen Sie eine Verknüpfungsbedingung für die Tabellenknoten hinzu.
   1) Heben Sie den Root-Knotennamen hervor, und wählen Sie **Neues Element** aus.
   2) Geben Sie im Bereich **Details** für **Knotenart** die Auswahl **Bedingung** an.
   3) Geben Sie im Feld **Bedingung** die Verknüpfungsbedingung mit der folgenden Syntax ein:

      ```
      table_name.table_column = table_name.table_column AND
      table_name.table_column = table_name.table_column ...
      ```

   4) Klicken Sie **Hinzufügen** an, um die Bedingung hinzuzufügen.

Sie können diesem Knoten jetzt neue untergeordnete Elemente und Attribute hinzufügen.

11. **So fügen Sie einen Element- oder Attributknoten hinzu:**
    a. Klicken Sie einen übergeordneten Knoten in dem Feld links an, um ein untergeordnetes Element oder Attribut hinzuzufügen.

       Wenn Sie keinen übergeordneten Knoten ausgewählt haben, kann **New** nicht ausgewählt werden.

    b. Klicken Sie **Neues Element** an.
    c. Wählen Sie eine Knotenart im Menü **Knotenart** im Bereich **Details** aus.

       Das Menü **Knotenart** zeigt nur die Knotenarten an, die an dieser Stelle in der Zuordnung gültig sind. **Element** oder **Attribut**.

    d. Geben Sie einen Knotennamen im Feld **Knotenname** an.
    e. Klicken Sie **Hinzufügen** an, um den neuen Knoten hinzuzufügen.

f. **So ordnen Sie den Inhalt eines Element- oder Attributknotens einer relationalen Tabelle zu:**

1) Geben Sie einen Textknoten an.
   a) Klicken Sie den übergeordneten Knoten an.
   b) Klicken Sie **Neues Element** an.
   c) Wählen Sie im Feld **Knotenart** die Option **Text** aus.
   d) Wählen Sie **Hinzufügen** aus, um den Knoten auszuwählen.
2) Fügen Sie einen Tabellenknoten hinzu.
   a) Wählen Sie den eben erstellten Textknoten aus, und klicken Sie **Neues Element** an.
   b) Wählen Sie im Feld **Knotenart** die Option **Tabelle** aus, und geben Sie einen Tabellennamen für das Element ein.
   c) Klicken Sie **Hinzufügen** an, um den Knoten hinzuzufügen.
3) Fügen Sie einen Spaltenknoten hinzu.
   a) Wählen Sie den Textknoten erneut aus, und klicken Sie **Neues Element** an.
   b) Wählen Sie im Feld **Knotenart** die Option **Spalte** aus, und geben Sie einen Spaltennamen für das Element ein.
   c) Geben Sie einen Basisdatentyp für die Spalte im Feld **Typ** ein, um anzugeben, welchen Typ sie Spalte haben muß, um die nicht markierten Daten speichern zu können.
   d) Klicken Sie **Hinzufügen** an, um den Knoten hinzuzufügen.

   Einschränkung: Mit dem Verwaltungsassistenten können keine neuen Spalten erstellt werden. Wenn Sie **Spalte** als Knotenart auswählen, können Sie nur eine Spalte auswählen, die in Ihrer DB2-Datenbank bereits vorhanden ist.
4) Fügen Sie wahlweise eine Bedingung für die Spalte hinzu.
   a) Wählen Sie den Textknoten erneut aus, und klicken Sie **Neues Element** an.
   b) Wählen Sie im Feld **Knotenart** die Option **Bedingung** und die Bedingung mit der folgenden Syntax aus:

      operator LIKE|<|>|= *wert*
   c) Klicken Sie **Hinzufügen** an, um den Knoten hinzuzufügen.

Sie können diese Knoten ändern, indem Sie den Knoten auswählen, die Felder unter **Details** ändern und **Ändern** anklicken.

g. Fahren Sie mit dem Editieren der RDB-Zuordnung fort, oder klicken Sie **Weiter** an, um das Fenster **Eine DAD angeben** zu öffnen.

12. **So entfernen Sie einen Knoten:**

a. Klicken Sie einen Knoten in dem Feld auf der linken Seite an.

b. Klicken Sie **Entfernen** an.

c. Fahren Sie mit dem Editieren der RDB_node-Zuordnung fort, oder klicken Sie **Weiter** an, um das Fenster **Eine DAD angeben** zu öffnen.

13. Geben Sie einen Ausgabedateinamen für die geänderte DAD im Feld **Dateiname** des Fensters **Eine DAD angeben** ein.
14. Klicken Sie **Beenden** an, um den Knoten zu entfernen und zum LaunchPad-Fenster zurückzukehren.

Von der DB2-Befehls-Shell aus: Die DAD-Datei ist eine XML-Datei, die Sie mit einem beliebigen Texteditor erstellen können. Die folgenden Schritte zeigen Teile des Muster-Anhangs „Dokumentzugriffsdefinitionsdateien" auf Seite 286. In diesen Beispielen finden Sie ausführliche Informationen und den jeweiligen Kontext.

1. Öffnen Sie einen Texteditor.
2. Erstellen Sie die DAD-Kopfzeilen:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "pfad\dad.dtd"> --> Pfad- und
       Dateiname der DTD für die DAD
```

3. Fügen Sie die Befehle <DAD></DAD> ein.
4. Geben Sie nach dem Befehl <DAD> die DTD-ID an, die die DAD-Datei mit der DTD des XML-Dokuments verbindet.

```
<dtdid>pfad\dtd_name.dtd> --> Pfad- und Dateiname der DTD
       für Ihre Anwendung
```

5. Geben Sie an, ob eine Gültigkeitsprüfung durchgeführt werden soll (ob über eine DTD sichergestellt werden soll, daß das XML-Dokument ein gültiges XML-Dokument ist). Beispiel:

```
<validation>NO</validation>  --> Geben Sie YES oder NO an
```

6. Verwenden Sie das Element <Xcollection> zum Definieren der Zugriffs- und Speichermethode als XML-Objektgruppe. Die Zugriffs- und Speichermethode geben an, daß die XML-Daten in einer Objektgruppe von DB2-Tabellen gespeichert sind.

```
<Xcollection>
</Xcollection>
```

7. Fügen Sie die folgenden Prologinformationen hinzu:

```
<prolog>?xml version="1.0"?</prolog>
```

   Dieser Text ist in genau dieser Form erforderlich.

8. Fügen Sie die Befehle <doctype></doctype> hinzu. Beispiel:

```
<doctype>! DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. Definieren Sie den Root-Knoten mit den Befehlen <root_node></root_node>. Innerhalb des Root-Knotens geben Sie die Elemente und Attribute an, aus denen das XML-Dokument besteht.

10. Ordnen Sie nach dem Befehl <root_node> die Elemente und Attribute in dem XML-Dokument den Element- und Attributknoten zu, die den DB2-Daten entsprechen. Diese Knoten bieten einen Pfad von den XML-Daten zu den DB2-Daten.
    a. Definieren Sie einen Anfangs-Root-element_node. Dieser Elementknoten enthält:
       - Tabellenknoten mit einer Verknüpfungsbedingung zur Angabe der Objektgruppe.
       - Untergeordnete Elemente
       - Attribute

       So geben Sie die Tabellenknoten und die Bedingung an:

       1) Erstellen Sie ein RDB_node-Element. Beispiel:

          ```
          <RDB_node>
          </RDB_node>
          ```

       2) Definieren Sie einen <table_node> für jede Tabelle mit Daten, die in das XML-Dokument einbezogen werden sollen. Wenn Sie beispielsweise drei Tabellen ORDER_TAB, PART_TAB und SHIP_TAB haben, die Spaltendaten für das Dokument enthalten, erstellen Sie für jede der Tabellen einen Tabellenknoten. Beispiel:

          ```
          <RDB_node>
          <table name="ORDER_TAB">
          <table name="PART_TAB">
          <table name="SHIP_TAB"></RDB_node>
          ```

       3) Definieren Sie eine Verknüpfungsbedingung für die Tabellenknoten in der Objektgruppe. Die Syntax lautet:

          ```
          ausdruck = ausdruck AND
          ausdruck = ausdruck ...
          ```

          Beispiel:

          ```
          <RDB_node>
          <table name="ORDER_TAB">
          <table name="PART_TAB">
          <table name="SHIP_TAB">
          <condition>
             order_tab.order_key = part_tab.order_key AND
             part_tab.part_key = ship_tab.part_key
          </condition>
          </RDB_node>
          ```

       4) Geben Sie für jede Tabelle einen Primärschlüssel an. Der Primärschlüssel besteht aus einer einzelnen Spalte oder mehreren Spalten, dem zusammengesetzten Schlüssel. Zum Angeben des Primärschlüssels fügen Sie einen Attributschlüssel zum Tabellenelement des RDB_node hinzu. Das folgende Beispiel definiert einen Primärschlüssel für jede der Tabellen im RDB_node des Root-Elementknotens Order:

```
<element_node name="Order">
    <RDB_node>
         <table name="order_tab" key="order_key"/>
         <table name="part_tab" key="part_key price"/>
         <table name="ship_tab" key="date mode"/>
         <condition>
             order_tab.order_key = part_tab.order_key AND
             part_tab.part_key = ship_tab.part_key
         </condition>
    <RDB_node>
```

Die für das Zerlegen angegebenen Informationen werden beim Zusammensetzen eines XML-Dokuments ignoriert.

Das Schlüsselattribut ist zum Zusammensetzen erforderlich; wenn Sie eine Objektgruppe aktivieren, muß die verwendete DAD-Datei sowohl das Zusammensetzen wie auch das Zerlegen unterstützen.

b. Definieren Sie einen Befehl <element_node> für jedes Element in Ihrem XML-Dokument, das einer Spalte in einer DB2-Tabelle zugeordnet ist. Beispiel:

```
<element_node name="name">
</element_node>
```

Ein Elementknoten kann einen der folgenden Elementtypen enthalten:

- <text_node>: zur Angabe, daß das Element einen Inhalt für eine DB2-Tabelle hat; in diesem Fall enthält das Element keine untergeordneten Elemente.
- <attribute_node>: zur Angabe eines Attributs; Attributknoten werden im nächsten Schritt definiert
- Untergeordnete Elemente

Der text_node enthält einen RDB_node für die Zuordnung von Inhalt zu einer DB2-Tabelle und einem Spaltennamen.

RDB_nodes werden für Elemente auf der untersten Ebene verwendet, die einen Inhalt haben, der einer DB2-Tabelle zugeordnet werden soll. Ein RDB_node hat die folgenden untergeordneten Elemente.

- <table>: definiert die Tabelle, die dem Element entspricht
- <column>: definiert die Spalte, die das entsprechende Element enthält.
- <condition>: definiert wahlweise eine Bedingung zu der Spalte

Sie können beispielsweise ein XML-Element <Tax> haben, für das Sie den nicht markierten Inhalt in der Spalte ″TAX″ speichern wollen:

XML-Dokument:

```
<Tax>0.02</Tax>
```

In diesem Fall soll der Wert `0.02` in der Spalte TAX gespeichert werden.

In der DAD-Datei geben Sie einen <RDB_node> ein, um das XML-Element der DB2-Tabelle und der Spalte zuzuordnen.

DAD-Datei:

```
<element_node name="Tax">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
         <column name="tax"/>
      </RDB_node>
   </text_node>
</element_node>
```

Der <RDB_node> gibt an, daß der Wert des Elements <Tax> ein Textwert ist und daß die Daten in der Tabelle PART_TAB in der Spalte TAX gespeichert werden.

c. Definieren Sie einen <attribute_node> für jedes Attribut in Ihrem XML-Dokument, das einer Spalte in einer DB2-Tabelle zugeordnet ist. Beispiel:

```
<attribute_node name="key">
</attribute_node>
```

Der attribute_node enthält einen RDB_node für die Zuordnung des Attributwerts zu einer DB2-Tabelle und Spalte. Ein RDB_node hat die folgenden untergeordneten Elemente.

- <table>: definiert die Tabelle, die dem Element entspricht
- <column>: definiert die Spalte, die das entsprechende Element enthält.
- <condition>: definiert wahlweise eine Bedingung zu der Spalte

Sie können beispielsweise einen Attributschlüssel für ein Element <Order> haben. Der Wert des Schlüssels muß in einer Spalte PART_KEY gespeichert werden.

XML-Dokument:

```
<Order key="1">
```

Erstellen Sie in der DAD-Datei einen Attributknoten für den Schlüssel, und geben Sie die Tabelle an, in der der Wert 1 gespeichert werden soll.

DAD-Datei:

```
<attribute_node name="key">
  <RDB_node>
    <table name="part_tab">
    <column name="part_key"/>
  <RDB_node>
</attribute_node>
```

11. Geben Sie den Spaltentyp für den RDB_node für jeden attribute_node und text_node an. Auf diese Weise wird der richtige Datentyp für jede Spalte sichergestellt, in denen nicht markierte Daten gespeichert werden. Fügen Sie zur Angabe der Spaltentypen dem Spaltenelement den Attributtyp hinzu. Das folgende Beispiel definiert einen Spaltentyp als INTEGER:

```
<attribute_node name="key">
    <RDB_node>
        <table name="order_tab"/>
        <column name="order_key" type="integer"/>
    </RDB_node>
</attribute_node>
```

12. Stellen Sie sicher, daß Sie einen Endbefehl </root_node> nach dem letzten Befehl </element_node> verwenden.
13. Stellen Sie sicher, daß Sie einen Endbefehl </Xcollection> nach dem Befehl </root_node> verwenden.
14. Stellen Sie sicher, daß Sie einen Endbefehl </DAD> nach dem Befehl </Xcollection> verwenden.

XML-Objektgruppen aktivieren

Beim Aktivieren einer XML-Objektgruppe wird die DAD-Datei syntaktisch analysiert, um die Tabellen und Spalten zu dem XML-Dokument zu identifizieren, und in der Tabelle XML_USAGE werden entsprechende Steuerinformationen aufgezeichnet. Das Aktivieren einer XML-Objektgruppe ist für folgende Aktionen wahlfrei:

- Zerlegen eines XML-Dokuments und Speichern der Daten in neuen DB2-Tabellen
- Zusammensetzen eines XML-Dokuments aus vorhandenen Daten in verschiedenen DB2-Tabellen

Wenn dieselbe DAD-Datei zum Zusammensetzen und Zerlegen verwendet wird, können Sie die Objektgruppe für das Zusammensetzen und das Zerlegen aktivieren.

Sie können eine XML-Objektgruppe über den XML Extender-Verwaltungsassistenten, über den Befehl **dxxadm** mit der Option enable_collection oder über die gespeicherte Prozedur dxxEnableCollection() des XML Extender aktivieren.

Verwaltungsassistent verwenden

Führen Sie zum Aktivieren einer XML-Objektgruppe die folgenden Schritte aus:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit XML-Objektgruppen arbeiten** im LaunchPad-Fenster an. Das Fenster **Eine Task auswählen** wird angezeigt.
3. Klicken Sie **Eine Objektgruppe aktivieren** und anschließend **Weiter** an. Das Fenster **Aktivieren einer Objektgruppe** wird angezeigt.
4. Wählen Sie den Namen der zu aktivierenden Objektgruppe im Feld **Name der Objektgruppe** im Kontextmenü aus.
5. Geben Sie den DAD-Dateinamen im Feld **DAD-Dateiname** ein, oder klicken Sie **...** an, um nach einer vorhandenen DAD-Datei zu suchen.
6. Wahlweise können Sie den Namen eines zuvor erstellten Tabellenbereichs im Feld **Tabellenbereich** eingeben.

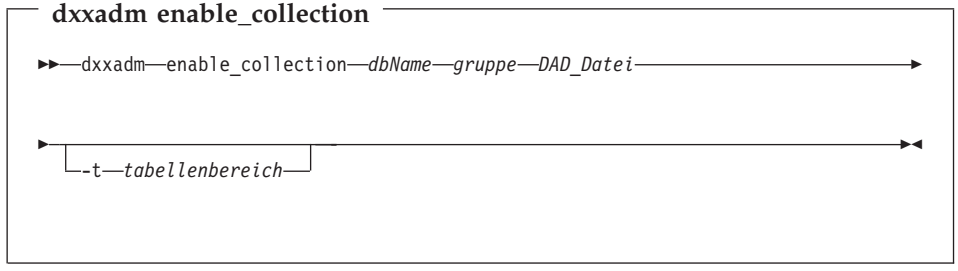
   Der Tabellenbereich enthält neue, zum Zerlegen generierte DB2-Tabellen.
7. Klicken Sie **Beenden** an, um die Objektgruppe zu aktivieren und zum LaunchPad-Fenster zurückzukehren.
   - Wenn die Objektgruppe erfolgreich aktiviert wurde, erscheint die Nachricht `Objektgruppe erfolgreich aktiviert`.

- Wurde die Objektgruppe nicht erfolgreich aktiviert, erscheint eine Fehlernachricht. Fahren Sie mit diesen Schritten fort, bis die Objektgruppe erfolgreich aktiviert wurde.

Von der DB2-Befehls-Shell aus

Geben Sie zum Aktivieren einer XML-Objektgruppe den Befehl **dxxadm** ein:

Syntax:



Parameter:

dbName
Der Name der Datenbank.

gruppe Der Name der XML-Objektgruppe. Dieser Wert wird als Parameter für die gespeicherten Prozeduren der XML-Objektgruppe verwendet.

DAD_Datei
Der Name der Datei, die die Dokumentzugriffsdefinition (DAD) enthält.

tabellenbereich
Ein vorhandener Tabellenbereich, der neue DB2-Tabellen enthält, die zum Zerlegen generiert wurden. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.

Beispiel: Das folgende Beispiel aktiviert eine Objektgruppe "sales_ord" in der Datenbank SALES_DB über die DB2-Befehls-Shell. Die DAD-Datei verwendet eine SQL-Zuordnung, sie ist in „DAD-Datei: XML-Objektgruppe - SQL-Zuordnung" auf Seite 288 beschrieben.

```
dxxadm enable_collection SALES_DB sales_ord getstart.dad
```

Nachdem Sie die XML-Objektgruppe aktiviert haben, können Sie XML-Dokumente mit den gespeicherten Prozeduren des XML Extender zusammensetzen oder zerlegen.

XML-Objektgruppen inaktivieren

Durch das Inaktivieren einer XML-Objektgruppe werden die Datensätze, die Tabellen und Spalten als Teil einer Objektgruppe kennzeichnen, aus der Tabelle XML_USAGE entfernt. Es werden keine Datentabellen freigegeben. Sie inaktivieren eine Objektgruppe, wenn Sie die DAD aktualisieren und eine Objektgruppe wieder aktivieren oder eine Objektgruppe freigeben wollen.

Sie können eine XML-Objektgruppe über den XML Extender-Verwaltungsassistenten, über den Befehl **dxxadm** mit der Option disable_collection oder über die gespeicherte Prozedur dxxDisableCollection() des XML Extender inaktivieren.

Verwaltungsassistent verwenden

Führen Sie zum Inaktivieren einer XML-Objektgruppe die folgenden Schritte aus:

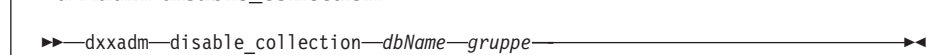
1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Mit XML-Objektgruppen arbeiten** im LaunchPad-Fenster an, um die Tasks zu XML Extender-Objektgruppen anzuzeigen. Das Fenster **Eine Task auswählen** wird angezeigt.
3. Klicken Sie **Eine XML-Objektgruppe inaktivieren** und anschließend **Weiter** an, um eine XML-Objektgruppe zu inaktivieren. Das Fenster **Inaktivieren einer Objektgruppe** wird angezeigt.
4. Geben Sie den Namen der zu inaktivierenden Objektgruppe im Feld **Name der Objektgruppe** ein.
5. Klicken Sie **Beenden** an, um die Objektgruppe zu inaktivieren und zum LaunchPad-Fenster zurückzukehren.
   - Wenn die Objektgruppe erfolgreich inaktiviert wurde, erscheint die Nachricht `Objektgruppe erfolgreich aktiviert`.
   - Wurde die Objektgruppe nicht erfolgreich inaktiviert, erscheint eine Fehlernachricht. Fahren Sie mit diesen Schritten fort, bis die Objektgruppe erfolgreich inaktiviert wurde.

Von der DB2-Befehls-Shell aus

Geben Sie zum Inaktivieren einer XML-Objektgruppe den Befehl **dxxadm** ein:

Syntax:

dxxadm disable_collection



Parameter:

dbName
Der Name der Datenbank.

gruppe Der Name der XML-Objektgruppe. Dieser Wert wird als Parameter für die gespeicherten Prozeduren der XML-Objektgruppe verwendet.

Beispiel:

```
dxxadm disable_collection SALES_DB sales_ord
```

Datenbank für XML inaktivieren

Sie inaktivieren die Datenbank, wenn Sie Ihre XML Extender-Umgebung bereinigen und die XML Extender-UDTs, UDFs, gespeicherte Prozeduren und Tabellen zur Verwaltungsunterstützung freigeben wollen. XML Extender inaktiviert die Datenbank, zu der Sie eine Verbindung haben, mit dem aktuellen Exemplar.

Wenn Sie eine Datenbank für XML inaktivieren, führt der XML Extender die folgenden Aktionen für die Datenbank aus:

- Löschen aller benutzerdefinierten Typen (UDTs) und Funktionen (UDFs)
- Löschen der Steuertabellen mit den Metadaten für den XML Extender
- Löschen des Schemas db2xml

Vorbereitungen

Inaktivieren Sie alle XML-Spalten oder Objektgruppe in der zu inaktivierenden Datenbank.

Verwaltungsassistent verwenden

Führen Sie zum Inaktivieren einer Datenbank für XML-Daten die folgenden Schritte aus:

1. Stellen Sie den Verwaltungsassistenten ein und starten Sie ihn. Siehe hierzu den Abschnitt „Verwaltungsassistent starten" auf Seite 73.
2. Klicken Sie **Datenbank inaktivieren** im LaunchPad-Fenster an, um die aktuelle Datenbank zu inaktivieren.

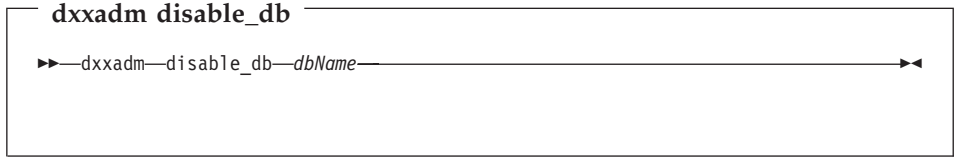
   Wenn eine Datenbank momentan nicht aktiviert ist, kann nur **Aktivieren einer Datenbank** ausgewählt werden.

   Wenn die Datenbank inaktiviert ist, kehren Sie zurück zum LaunchPad-Fenster.

Von der DB2-Befehls-Shell aus

Geben Sie **dxxadm** in der Befehlszeile ein, und geben Sie dabei die zu inaktivierende Datenbank an.

Syntax:



Parameter:

dbName
Der Name der zu inaktivierenden Datenbank.

Beispiel: Inaktiviert eine vorhandene Datenbank mit dem Namen SALES_DB.

```
dxxadm disable_db SALES_DB
```

Teil 3. Programmierung

Dieser Teil des Handbuchs beschreibt Programmiertechniken zur Verwaltung Ihrer XML-Daten.

Kapitel 5. XML-Spaltendaten verwalten

Bei der Verwendung von XML-Spalten speichern Sie ein vollständiges XML-Dokument in Form von Spaltendaten. Mit dieser Zugriffs- und Speichermethode bleibt das XML-Dokument intakt, und Sie haben dennoch die Möglichkeit, das Dokument zu indexieren und zu durchsuchen, Daten aus dem Dokument abzurufen und das Dokument zu aktualisieren. Eine XML-Spalte enthält XML-Dokumente in ihrem nativen DB2-Format als Spaltendaten. Nach dem Aktivieren einer Datenbank für XML stehen die folgenden benutzerdefinierten Typen (UDTs) zur Verfügung:

XMLCLOB
: XML-Dokumentinhalte, die als großes Zeichenobjekt (″Character Large Object, CLOB) in DB2 gespeichert sind

XMLVARCHAR
: XML-Dokumentinhalte, die als VARCHAR in DB2 gespeichert sind

XMLFile
: XML-Dokumente, die in einer Datei auf einem lokalen Dateisystem gespeichert sind

Sie können mit XML-UDTs als Datentyp für Spalten Anwendungstabellen erstellen oder ändern. Diese Tabellen werden als XML-Tabellen bezeichnet. Informationen zum Erstellen und Ändern von Tabellen für XML finden Sie im Abschnitt „XML-Tabelle erstellen oder ändern” auf Seite 86.

Nach dem Aktivieren einer Spalte für XML können Sie mit der Verwaltung des Inhalts der XML-Spalte beginnen. Sobald die XML-Spalte erstellt ist, können Sie die folgenden Verwaltungs-Tasks ausführen:

- Speichern von XML-Dokumenten in DB2
- Abrufen von XML-Daten oder -Dokumenten aus DB2
- Aktualisieren von XML-Dokumenten
- Löschen von XML-Daten oder -Dokumenten

Sie können diese Tasks über zwei verschiedene Methoden ausführen:

- *Standardumsetzungsfunktionen*, die den SQL-Basistyp in den XML-UDT umsetzen
- Von XML Extender bereitgestellte benutzerdefinierte Funktionen (UDFs)

In diesem Handbuch werden beide Methoden für jede der Tasks beschrieben.

UDT- und UDF-Namen

Der vollständige Name einer DB2-Funktion lautet *schemaname.funktionsname*, wobei *schemaname* eine Kennung ist, die eine logische Gruppierung für SQL-Objekte bietet. Der Schemaname für XML Extender-UDFs lautet db2xml. Der Schemaname db2xml ist außerdem das Qualifikationsmerkmal für die XML Extender-UDTs. In diesem Handbuch werden nur Verweise auf den Funktionsnamen verwendet.

Der Funktionspfad ist eine geordnete Liste von Schemanamen. DB2 verwendet die Reihenfolge von Schemanamen in der Liste, um die Verweise auf Funktionen und UDTs aufzulösen. Sie können den Funktionspfad durch Angabe der SQL-Anweisung SET CURRENT FUNCTION PATH angeben. Dadurch wird der Funktionspfad im speziellen Register CURRENT FUNCTION PATH festgelegt.

Für den XML Extender empfiehlt es sich, das Schema db2xml dem Funktionspfad hinzuzufügen. Auf diese Weise können Sie XML Extender-UDF- und -UDT-Namen eingeben, ohne das Präfix `db2xml` voranzustellen. Das folgende Beispiel zeigt, wie Sie das Schema db2xml dem Funktionspfad hinzufügen:

```
SET CURRENT FUNCTION PATH = db2xml, CURRENT FUNCTION PATH
```

Wichtig: Fügen Sie `db2xml` nicht als erstes Schema im Funktionspfad hinzu, wenn Sie sich als db2xml anmelden; db2xml wird automatisch als erstes Schema festgelegt, wenn Sie sich als db2xml anmelden. Dies führt zu einer Fehlerbedingung, da Ihr Funktionspfad mit zwei Schemata db2xml beginnt.

Daten speichern

Mit dem XML Extender können Sie intakte XML-Dokumente in eine XML-Spalte einfügen. Wenn Sie Seitentabellen definieren, aktualisiert der XML Extender diese Tabellen automatisch. Wenn Sie ein XML-Dokument direkt speichern, speichert der XML Extender den Basistyp als XML-Typ.

Task-Übersicht:

1. Stellen Sie sicher, daß Sie die DAD-Datei erstellt bzw. aktualisiert haben.
2. Stellen Sie fest, welcher Datentyp beim Speichern des Dokuments verwendet werden soll.
3. Wählen Sie eine Methode zum Speichern der Daten in der DB2-Tabelle aus (Umsetzungsfunktionen oder UDFs).
4. Geben Sie eine SQL-Anweisung INSERT an, die die XML-Tabelle und die die Spalte angibt, die das XML-Dokument enthalten soll.

Der XML Extender bietet zwei Methoden zum Speichern von XML-Dokumenten: Standardumsetzungsmethoden und Speicher-UDFs. Tabelle 8 zeigt, in welchen Fällen diese Methoden verwendet werden sollten.

Tabelle 8. Die XML Extender-Speicherfunktionen

Basistyp	**Speichern in DB2 als...**		
	XMLVARCHAR	**XMLCLOB**	**XMLFILE**
VARCHAR	XMLVARCHAR()	N/A	XMLFileFromVarchar()
CLOB	N/A	XMLCLOB()	XMLFileFromCLOB()
FILE	XMLVarcharFromFile()	XMLCLOBFromFile()	XMLFILE

Standardumsetzungsfunktion verwenden

Für jeden UDT gibt es eine *Standardumsetzungsfunktion* zur Umsetzung des SQL-Basistyps in den UDT. Sie können die vom XML Extender bereitgestellten Umsetzungsfunktionen in Ihren VALUES-Klauseln verwenden, um Daten einzufügen. Tabelle 9 zeigt die bereitgestellten Umsetzungsfunktionen:

Tabelle 9. Die XML Extender-Standardumsetzungsfunktionen

In SELECT-Klausel verwendete Umsetzung	**Rückgabetyp**	**Beschreibung**
XMLVARCHAR(VARCHAR)	XMLVARCHAR	Eingabe aus dem Speicherpuffer als VARCHAR
XMLCLOB(CLOB)	XMLCLOB	Eingabe aus dem Speicherpuffer als CLOB oder CLOB-Zeigereinheit
XMLFILE(VARCHAR)	XMLFILE	Nur Speicherung des Dateinamens

Beispiel: Die folgende Anweisung fügt einen umgesetzten Typ VARCHAR in den Typ XMLVARCHAR ein:

```
INSERT INTO sales_tab
VALUES('123456', 'Sriram Srinivasan', db2xml.XMLVarchar(:xml_buff))
```

Als Speicher-UDF:

Für jeden XML Extender-UDT gibt es eine Speicher-UDF zum Importieren von Daten in DB2 aus einer anderen Ressource als ihrem Basistyp. Wenn Sie beispielsweise eine XML-Datei als XMLCLOB in DB2 importieren wollen, können Sie die Funktion XMLCLOBFromFile() verwenden.

Tabelle 10 zeigt die vom XML Extender bereitgestellten Speicherfunktionen.

Tabelle 10. Die Speicher-UFDs des XML Extender

Benutzerdefinierte Speicherfunktion	Rückgabetyp	Beschreibung
XMLVarcharFromFile()	XMLVARCHAR	Liest ein XML-Dokument aus einer Datei auf dem Server ein und gibt den Wert des Typs XMLVARCHAR zurück.
XMLCLOBFromFile()	XMLCLOB	Liest ein XML-Dokument aus einer Datei auf dem Server ein und gibt den Wert des Typs XMLCLOB zurück.
XMLFileFromVarchar()	XMLFILE	Liest ein XML-Dokument aus dem Speicher als VARCHAR ein, schreibt es in eine externe Datei und gibt den Wert des Typs XMLFILE zurück; dieser Wert ist der Dateiname.
XMLFileFromCLOB()	XMLFILE	Liest ein XML-Dokument aus dem Speicher als CLOB oder als CLOB-Zeigereinheit ein, schreibt es in eine externe Datei und gibt den Wert des Typs XMLFILE zurück; dieser Wert ist der Dateiname.

Beispiel: Die folgende Anweisung speichert einen Datensatz mit der Funktion XMLCLOBFromFile() in einer XML-Tabelle als XMLCLOB.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
    VALUES( '1234', 'Sriram Srinivasan,
      XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

In diesem Beispiel wird das XML-Objekt aus der Datei `c:\dxx\samples\cmd\getstart.xml` in die Spalte ORDER in der Tabelle SALES_TAB importiert.

Daten abrufen

Mit dem XML Extender können Sie entweder ein vollständiges Dokument oder den Inhalt der Elemente und Attribute abrufen. Wenn Sie eine XML-Spalte direkt abrufen, gibt der XML Extender den UDT als Spaltentyp zurück. Ausführliche Informationen zum Abrufen von Daten finden Sie in den folgenden Abschnitten:

- „Ein vollständiges Dokument abrufen"
- „Elementinhalte und Attributwerte abrufen" auf Seite 133

Der XML Extender bietet zwei Methoden zum Abrufen von Daten: Standardumsetzungsfunktionen und die überladene UDF Content(). Tabelle 11 zeigt, in welchen Fällen die beiden Methoden verwendet werden sollten.

Tabelle 11. Die XML Extender-Funktionen zum Abrufen

XML-Typ	**Aus DB2 abrufen als...**		
	VARCHAR	**CLOB**	**FILE**
XMLVARCHAR	VARCHAR	N/A	Content()
XMLCLOB	N/A	XMLCLOB	Content()
XMLFILE	N/A	Content()	FILE

Ein vollständiges Dokument abrufen

Task-Übersicht:

1. Stellen Sie sicher, daß Sie das XML-Dokument in einer XML-Tabelle gespeichert haben, und stellen Sie fest, welche Daten abgerufen werden sollen.
2. Wählen Sie eine Methode zum Abrufen der Daten aus der DB2-Tabelle aus (Umsetzungsfunktionen oder UDFs).
3. Bei Verwendung der überladenen UDF Content() muß ermittelt werden, welcher Datentyp den abzurufenden Daten zugeordnet ist und welcher Datentyp exportiert werden soll.
4. Geben Sie eine SQL-Abfrage an, die die XML-Tabelle und -Spalte angibt, aus der das XML-Dokument abgerufen werden soll.

Der XML Extender bietet zwei Methoden zum Abrufen von Daten:

Standardumsetzungsfunktion verwenden
Verwenden Sie die mit DB2 bereitgestellte Standardumsetzungsfunktion für UDTs, um einen XML-UDT in einen SQL-Basistyp umzusetzen und anschließend damit zu arbeiten. Sie können die vom XML Extender bereitgestellten Umsetzungsfunktionen in Ihren SELECT-Anweisungen zum Abrufen von Daten verwenden. Tabelle 12 auf Seite 132 zeigt die bereitgestellten Umsetzungsfunktionen:

Tabelle 12. Die XML Extender-Standardumsetzungsfunktionen

In SELECT-Klausel verwendete Umsetzung	Rückgabetyp	Beschreibung
varchar(XMLVARCHAR)	VARCHAR	XML-Dokument in VARCHAR
clob(XMLCLOB)	CLOB	XML-Dokument in CLOB
varchar(XMLFile)	VARCHAR	XML-Dateiname in VARCHAR

Beispiel: Im folgenden Beispiel werden XMLVARCHAR-Daten abgerufen und im Speicher als Datentyp VARCHAR abgelegt:

```
EXEC SQL SELECT db2xml.varchar(order) from sales_tab
```

Überladene UDF Content() verwenden

Verwenden Sie die UDF Content() zum Abrufen des Dokumentinhalts aus dem Zusatzspeicher in den Hauptspeicher, oder exportieren Sie das Dokument aus dem internen Speicher in eine *externe Datei* auf dem DB2-Server.

Wenn Sie beispielsweise Ihr XML-Dokument als XMLFILE gespeichert haben und im Hauptspeicher damit arbeiten wollen, können Sie die UDF Content() verwenden, die den Datentyp XMLFILE als Eingabe verwenden und CLOB zurückgeben kann.

Die UDF Content() führt je nach dem angegebenen Datentyp zwei verschiedene Abruffunktionen aus:

Ruft ein Dokument aus dem Zusatzspeicher ab und legt es im Hauptspeicher ab

Sie können mit Content() das XML-Dokument in einen Speicherpuffer abrufen oder in eine CLOB-Zeigereinheit, wenn das Dokument als externe Datei gespeichert ist. Verwenden Sie die folgende Funktionssyntax, wobei *xmlobj* die abgefragte XML-Spalte ist:

XMLFILE to CLOB: Ruft Daten aus einer Datei ab und exportiert sie in eine CLOB-Zeigereinheit.

```
Content(xmlobj XMLFile)
```

Ruft ein Dokument aus dem internen Speicher ab und exportiert es in eine externe Datei

Sie können mit Content() auch ein in DB2 als Datentyp XML-CLOB gespeichertes XML-Dokument abrufen und es in eine Datei im Dateisystem des Datenbank-Servers exportieren. Der Name der Datei wird als Datentyp VARCHAR zurückgegeben. Verwenden Sie die folgende Syntax, wobei *xmlobj* die

abgefragte XML-Spalte und *Dateiname* die externe Datei ist. *XML-Typ* kann den Datentyp XMLVARCHAR oder XMLCLOB haben.

XML-Typ **to external file:** Ruft den als Datentyp XML gespeicherten XML-Inhalt ab und exportiert ihn in eine externe Datei.

```
Content(xmlobj
XML-Typ, dateiname varchar(512))
```

Hierbei gilt folgendes:

xmlobj Der Name der XML-Spalte, aus der der XML-Inhalt abgerufen werden soll; *xmlobj* kann den Datentyp XMLVARCHAR oder XMLCLOB haben.

dateiname
Der Name der Datei, in der die XML-Daten gespeichert werden sollen.

In dem folgenden Beispiel verdeutlicht ein kleines C-Programm- segment mit *eingebettetem SQL-Code*, wie ein XML-Dokument aus einer Datei in den Hauptspeicher abgerufen wird. Bei diesem Beispiel wird davon ausgegangen, daß die Spalte BOOK den Datentyp XMLFILE hat.

```
EXEC SQL BEGIN DECLARE SECTION;
          SQL TYPE IS CLOB_LOCATOR xml_buff;
          EXEC SQL END DECLARE SECTION;
          EXEC SQL CONNECT TO SALES_DB
          EXEC SQL DECLARE c1 CURSOR FOR
           SELECT Content(order) from sales_tab
                       EXEC SQL OPEN c1;
          do {
          EXEC SQL FETCH c1 INTO :xml_buff;
          if (SQLCODE != 0) {
           break;
           }
          else {
          /* beliebige Aktion mit dem XML-Dok im Puffer ausführen */}}
          EXEC SQL CLOSE c1;
          EXEC SQL CONNECT RESET;
```

Elementinhalte und Attributwerte abrufen

Sie können den Inhalt eines *Elements* oder eines *Attribut*werts aus einem oder mehreren XML-Dokumenten (Suche in einem einzelnen Dokument oder in einer Dokumentgruppe) abrufen (extrahieren). Der XML Extender bietet benutzerdefinierte Funktionen zum Extrahieren, die Sie in der SELECT-Klausel für jeden SQL-Datentyp angeben können.

Das Abrufen von Inhalt und Werten der Elemente und Attribute ist bei der Entwicklung von Anwendungen sehr hilfreich, weil Sie auf XML-Daten als relationale Daten zugreifen können. Sie könnten beispielsweise 1000 XML-Dokumente haben, die in der Spalte ORDER in der Tabelle SALES_TAB gespeichert sind. Sie können die Namen aller Kunden, die bereits Artikel bestellt haben, mit der folgenden SQL-Anweisung abrufen; die UDF zum Extrahieren ist in der SELECT-Klausel zum Abrufen dieser Informationen enthalten:

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

In diesem Beispiel ruft die Extraktions-UDF das Element <customer> aus der Spalte ORDER als Datentyp VARCHAR ab. Der Standortpfad lautet `/Order/Customer/Name` (die Syntax zum Standortpfad ist im Abschnitt „Standortpfad" auf Seite 56 beschrieben). Darüber hinaus wird die Anzahl der zurückgegebenen Werte mit Hilfe einer WHERE-Klausel verringert; diese Klausel gibt an, daß nur der Inhalt der Elemente <customer> abgerufen werden soll, deren Unterelement <ExtendedPrice> einen Wert größer als 2500.00 hat.

So extrahieren Sie Elementinhalte oder Attributwerte: Verwenden Sie die in Tabelle 13 auf Seite 135 aufgeführten Extraktions-UDFs; verwenden Sie hierbei die folgende Syntax als Tabellen- oder *Skalarfunktion*:

```
extract abgerufener_datentyp(xmlobj, pfad)
```

Hierbei gilt folgendes:

abgerufener_datentyp
: Der Datentyp, der von der Extraktionsfunktion zurückgegeben wird; dies kann einer der folgenden Typen sein:

  - INTEGER
  - SMALLINT
  - DOUBLE
  - REAL
  - CHAR
  - VARCHAR
  - CLOB
  - DATE
  - TIME
  - TIMESTAMP
  - FILE

xmlobj Der Name der XML-Spalte, aus der das Element oder Attribut extrahiert werden soll. Diese Spalte muß als einer der folgenden benutzerdefinierten XML-Typen definiert sein:

- XMLVARCHAR
- XMLCLOB as LOCATOR
- XMLFILE

pfad Der Standortpfad des Elements oder Attributs in dem XML-Dokument (z. B. `/Order/Customer/Name`). Die Syntax für den Standortpfad ist im Abschnitt „Standortpfad" auf Seite 56 beschrieben.

Wichtig: Die Extraktions-UDFs unterstützen Standortpfade, die Prädikate mit Attribute haben, nicht jedoch Elemente. Das folgende Prädikat wird beispielsweise unterstützt:

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

Das folgende Prädikat wird nicht unterstützt:

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

Tabelle 13 zeigt die Extraktionsfunktionen im skalaren und tabellarischen Format:

Tabelle 13. Die XML Extender-Extraktionsfunktionen

Skalarfunktion	Tabellenfunktion	Zurückgegebener Spaltenname (Tabellenfunktion)	Rückgabetyp
extractInteger()	extractIntegers()	returnedInteger	INTEGER
extractSmallint()	extractSmallints()	returnedSmallint	SMALLINT
extractDouble()	extractDoubles()	returnedDouble	DOUBLE
extractReal()	extractReals()	returnedReal	REAL
extractChar()	extractChars()	returnedChar	CHAR
extractVarchar()	extractVarchars()	returnedVarchar	VARCHAR
extractCLOB()	extractCLOBs()	returnedCLOB	CLOB
extractDate()	extractDates()	returnedDate	DATE
extractTime()	extractTimes()	returnedTime	TIME
extractTimestamp()	extractTimestamps()	returnedTimestamp	TIMESTAMP

Skalarfunktion, Beispiel:

Im folgenden Beispiel wird ein Wert zurückgegeben, wenn der Attributwert für key = ″1″ ist. Der Wert wird als INTEGER extrahiert und automatisch in einen Typ DECIMAL konvertiert.

```
CREATE TABLE t1(key decimal(3,2));
INSERT into t1 values
SELECT * from table(db2xml.extractInteger(db2xml.XMLFile
       ('c:\dxx\samples\xml\getstart.xml'), '/Order/[@key="1"]'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird jeder Schlüsselwert für die Bestellung als INTEGER extrahiert

```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
       ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

XML-Daten aktualisieren

Mit dem XML Extender können Sie das gesamte XML-Dokument aktualisieren, indem Sie die XML-Spaltendaten ersetzen, oder Sie können die Werte der angegebenen Elemente oder Attribute aktualisieren.

Task-Übersicht:

1. Stellen Sie sicher, daß Sie das XML-Dokument in einer XML-Tabelle gespeichert haben, und stellen Sie fest, welche Daten abgerufen werden sollen.
2. Wählen Sie eine Methode zum Aktualisieren der Daten aus der DB2-Tabelle aus (Umsetzungsfunktionen oder UDFs).
3. Geben Sie eine SQL-Abfrage an, die die zu aktualisierende XML-Tabelle und -Spalte angibt.

Wichtig: Beim Aktualisieren einer Spalte, die für XML aktiviert ist, aktualisiert der XML Extender automatisch die Seitentabellen, um diese Änderungen nachzuvollziehen. Aktualisieren Sie diese Tabellen nicht direkt, ohne das in der XML-Spalte gespeicherte Original-XML-Dokument durch Ändern des entsprechenden XML-Elements oder -Attributwerts zu aktualisieren. Solche Aktualisierungen können zu Problemen mit der Konsistenz der Daten führen.

So aktualisieren Sie ein XML-Dokument:

Verwenden Sie eine der folgenden Methoden:

Standardumsetzungsfunktion verwenden

Für jeden benutzerdefinierten Typ (UDT) gibt es eine Standardumsetzungsfunktion zur Umsetzung des SQL-Basistyps in den UDT. Sie können die vom XML Extender bereitgestellten Umsetzungsfunktionen zum Aktualisieren des XML-Dokuments verwenden. Tabelle 9 auf Seite 129 zeigt die bereitgestellten Umsetzungsfunktionen; es wird davon ausgegangen, daß die Spalte ORDER mit einer anderen vom XML Extender bereitgestellten Funktion erstellt wurde.

Beispiel: Aktualisiert den Typ XMLVARCHAR aus dem umgesetzten Typ VARCHAR; dabei wird davon ausgegangen, daß `xml_buf` eine als Typ VARCHAR definierte Host-Variable ist.

```
UPDATE sales_tab VALUES('123456', 'Sriram Srinivasan',
db2xml.XMLVarchar(:xml_buff))
```

Speicher-UDF verwenden

Für jeden XML Extender-UDT gibt es eine Speicher-UDF zum Importieren von Daten in DB2 aus einer anderen Ressource als ihrem Basistyp. Sie können mit einer Speicher-UDF das gesamte XML-Dokument aktualisieren, indem Sie es ersetzen.

Beispiel: Mit dem folgenden Beispiel wird ein XML-Dokument über die Funktion XMLVarcharFromFile() aktualisiert:

```
UPDATE sales_tab
     set order = XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml')
     WHERE sales_person = 'Sriram Srinivasan'
```

In diesem Beispiel wird das XML-Objekt aus der Datei `c:\dxx\samples\cmd\getstart.xml` in die Spalte ORDER in der Tabelle SALES_TAB aktualisiert.

Im Abschnitt Tabelle 10 auf Seite 130 finden Sie eine Liste der vom XML Extender bereitgestellten Speicherfunktionen.

So aktualisieren Sie spezifische Elemente und Attribute eines XML-Dokuments:

Verwenden Sie die UDF Update(), um die Werte nacheinander zu aktualisieren, statt das gesamte Dokument zu aktualisieren. Mit der UDF geben Sie einen Standortpfad und den Wert des Elements oder Attributs an, für das der zu ersetzende Standortpfad steht. (Die Syntax für den Standortpfad ist im Abschnitt „Standortpfad" auf Seite 56 beschrieben.) Sie brauchen das XML-Dokument nicht zu editieren: Der XML Extender nimmt die Änderungen für Sie vor.

Die UDF Update aktualisiert die gesamte XML-Datei und stellt die Datei entsprechend den Informationen aus dem Parser wieder her. Im Abschnitt „Wie die Aktualisierungsfunktion das XML-Dokument verarbeitet" auf Seite 215 finden Sie Informationen dazu, wie die UDF Update das Dokument verarbeitet; hier finden Sie auch Beispiele für Dokument vor und nach der Aktualisierung.

Syntax:

```
Update(xmlobj, pfad, wert)
```

Hierbei gilt folgendes:

xmlobj Der Name der XML-Spalte, für die der Wert des Elements bzw. des Attributs aktualisiert werden soll.

pfad Der Standortpfad des zu aktualisierenden Elements bzw. Attributs. Die Syntax für den Standortpfad ist im Abschnitt „Standortpfad" auf Seite 56 beschrieben. Im Abschnitt „Mehrfaches Vorkommen" auf Seite 218 finden Sie Informati0onen darüber, was bei einem mehrfachen Vorkommen zu beachten ist.

wert Der zu aktualisierende Wert.

Beispiel: Mit der folgenden Anweisung wird der Wert des Elements <Customer> mit der UDF Update() auf die Zeichenfolge IBM aktualisiert:

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
       WHERE sales_person = 'Sriram Srinivasan'
```

Mehrfaches Vorkommen:

Wenn in der UDF Update() ein Pfad angegeben ist, wird der Inhalt jedes Elements oder Attributs mit einem entsprechenden Pfad mit dem angegebenen Wert ersetzt. Dies bedeutet: wenn in einem Dokument mehrere Pfade vorkommen, ersetzt die Aktualisierungsfunktion die vorhandenen Werte durch den Wert in dem Parameter *wert*.

XML-Dokumente durchsuchen

Das Durchsuchen von XML-Daten ähnelt dem Abrufen von XML-Daten: mit beiden Techniken werden Daten zur weiteren Bearbeitung abgerufen; in beiden Fällen werden mit der WHERE-Klausel Prädikate als Abrufkriterien definiert.

Der XML Extender bietet verschiedene Methoden zum Durchsuchen der XML-Dokumente in einer XML-Spalte, je nach den Anforderungen Ihrer Anwendung. Er bietet die Möglichkeit, die Dokumentstruktur zu durchsuchen und Ergebnisse entsprechend dem Elementinhalt und den Attributwerten zurückzumelden. Sie können eine Sicht der XML-Spalte und ihrer Seitentabellen durchsuchen, für einen höheren Durchsatz die Seitentabellen direkt durchsuchen oder die Extraktions-UDFs mit WHERE-Klauseln verwenden. Darüber hinaus können Sie den DB2 Text Extender verwenden und Spaltendaten im strukturellen Inhalt nach einer Zeichenfolge durchsuchen.

Mit dem XML Extender können Sie Indizes zu Spalten von Seitentabellen mit XML-Elementinhalte oder Attibutwerte verwenden, die aus XML-Dokumenten extrahiert wurden; auf diese Weise können Sie die Suche beschleunigen. Durch die Angabe des Datentyps eines Elements oder Attributs können Sie nach allgemeinen SQL-Datentypen suchen oder Bereiche durchsuchen. In unserem Beispiel zu den Bestellungen könnten Sie etwa nach allen Bestellungen suchen, deren Preis höher ist als 2500.00.

Darüber hinaus können Sie mit dem DB2 UDB Text Extender eine strukturelle Textsuche oder eine Volltextsuche durchführen. Sie können beispielsweise eine Spalte RESUME haben, die Bewerbungen zur Wiedervorlage im XML-Format enthält. Sie wollen die Namen aller Bewerber mit Java-Kenntnissen ermitteln. Hierzu durchsuchen Sie mit dem DB2 Text Extender die XML-Dokumente nach allen Wiedervorlagen, in denen das Element <skill> die Zeichenfolge JAVA enthält.

In den folgenden Abschnitten werden die Suchmethoden beschrieben:

- „XML-Dokument nach Struktur durchsuchen" auf Seite 140
- „Text Extender für eine strukturelle Textsuche verwenden" auf Seite 142

XML-Dokument nach Struktur durchsuchen

Mit den Suchfunktionen des XML Extender können Sie XML-Daten in einer Spalte entsprechend der Dokumentstruktur durchsuchen, also nach Elementen und Attributen. Zum Durchsuchen der Spaltendaten verwenden Sie eine SELECT-Anweisung auf unterschiedliche Arten; zurückgegeben wird ein *Ergebnissatz* entsprechend den Übereinstimmungen mit den Dokumentelementen und -attributen. Sie können mit folgenden Methoden nach Spaltendaten suchen:

- Suchen mit Direktabfrage in Seitentabellen
- Suchen über eine *verknüpfte Sicht*
- Suchen mit Extraktions-UDFs
- Suchen nach Elementen oder Attributen mit mehrfachem Vorkommen

Diese Methoden werden in den folgenden Abschnitten beschrieben; sie verwenden Beispiele aus dem folgenden Szenario. Die Anwendungstabelle SALES_TAB enthält eine XML-Spalte mit dem Namen ORDER. Diese Spalte enthält drei Seitentabellen ORDER_SIDE_TAB, PART_SIDE_TAB und SHIP_SIDE_TAB. Eine Standardsicht, sales_order_view, wurde beim Aktivieren der Spalte ORDER aktiviert; sie verknüpft diese Tabellen mit der folgenden Anweisung CREATE VIEW:

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                             order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_side_tab.order_key, order_side_tab.customer,
            part_side_tab.part_key, ship_side_tab.date
     FROM sales_tab, order_side_tab, part_side_tab, ship_side_tab
     WHERE sales_tab.invoice_num = order_side_tab.invoice_num
               AND sales_tab.invoice_num = part_side_tab.invoice_num
               AND sales_tab.invoice_num = ship_side_tab.invoice_num
```

Mit Direktabfrage in Seitentabellen suchen

Eine Direktabfrage mit einer Suche "Unterabfragen" bietet den besten Durchsatz für eine strukturelle Suche, wenn die Seitentabellen indexiert sind. Für ein korrektes Durchsuchen der Seitentabellen können Sie eine Abfrage oder Unterabfrage verwenden.

Beispiel: Die folgende Anweisung verwendet eine Abfrage und eine Unterabfrage, um eine Seitentabelle direkt zu durchsuchen:

```
SELECT sales_person from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

In diesem Beispiel ist invoice_num der Primärschlüssel in der Tabelle SALES_TAB.

Über eine verknüpfte Sicht suchen

Sie können mit dem XML Extender eine Standardsicht erstellen, die die Anwendungstabelle und die Seitentabellen mit einer eindeutigen ID verknüpft. Mit dieser Standardsicht oder jeder anderen Sicht, die Anwendungstabelle und Seitentabellen verknüpft, können Sie Spaltendaten durchsuchen und die Seitentabellen abfragen. Diese Methode bietet eine einzelne virtuelle Sicht der Anwendungstabelle und ihrer Seitentabellen. Je mehr Seitentabellen erstellt werden, desto aufwendiger ist allerdings die Abfrage.

Tip: Sie können die *root_id* oder DXXROOT_ID (vom XML Extender erstellt) verwenden, um die Tabellen beim Erstellen Ihrer eigenen Sicht zu verknüpfen.

Beispiel: Mit der folgenden Anweisung wird eine Sicht gesucht.

```
SELECT sales_person from sales_order_view
   WHERE price > 2500.00
```

Die SQL-Anweisung gibt die Werte von sales_person aus der verknüpften Sicht sales_order_view zurück, die Bestellungen mit einem Preis größer als 2500.00 aufweisen.

Mit Extraktions-UDFs suchen

Sie können auch die Extraktions-UDFs des XML Extender zum Suchen nach Elementen und Attributen verwenden, wenn Sie keine Indizes oder Seitentabellen für die Anwendungstabelle erstellt haben. Die Verwendung der Extraktions-UDFs zum Durchsuchen der XML-Daten ist sehr aufwendig und sollte nur in Verbindung mit WHERE-Klauseln verwendet werden, die die Anzahl der bei der Suche berücksichtigten XML-Dokumente begrenzen.

Beispiel: Die folgende Anweisung sucht mit einer Extraktions-UDF des XML Extender:

```
SELECT sales_person from sales_tab
     WHERE extractVarchar(order, '/Order/Customer/Name')
    like '%IBM%'
AND invoice_num > 100
```

In diesem Beispiel extrahiert die UDF die Elemente </Order/Customer/Name> mit dem Wert IBM.

Nach Elementen oder Attributen mit mehrfachem Vorkommen suchen

Verwenden Sie beim Suchen nach Elementen oder Attributen mit mehrfachem Vorkommen die DISTINCT-Klausel, um doppelte Werte zu verhindern.

Beispiel: Die folgende Anweisung sucht mit einer DISTINCT-Klausel:

```
SELECT sales_person from sales_tab
            WHERE invoice_num in
              (SELECT DISTINCT invoice_num from part_side_tab
               WHERE price > 2500.00 )
```

In diesem Beispiel gibt die DAD-Datei an, daß `/Order/Part/Price` mehrfach vorkommt, und erstellt dafür eine Seitentabelle PART_SIDE_TAB. Die Tabelle PART_SIDE_TAB kann mehrere Zeilen mit derselben invoice_num enthalten. Durch die Verwendung von DISTINCT werden nur eindeutige Werte zurückgegeben.

Text Extender für eine strukturelle Textsuche verwenden

Beim Durchsuchen der XML-Dokumentstruktur sucht der XML Extender Element- und Attributwerte, die in allgemeine Datentypen umgesetzt werden; er sucht jedoch nicht nach Text. Sie können den DB2 UDB Text Extender für eine strukturelle Suche oder eine Volltextsuche nach einer für XML aktivierten Spalte verwenden. Der Text Extender unterstützt die XML-Dokumentsuche in DB2 UDB Version 6.1 und höheren Versionen. Text Extender ist für Windows Betriebssysteme, AIX und Sun Solaris verfügbar.

Strukturelle Textsuche

Sucht nach Textzeichenfolgen, die auf der Baumstruktur des XML-Dokuments aufbauen. Wenn Sie beispielsweise die Dokumentstruktur `/Order/Customer/Name` haben und nach der Zeichenfolge „IBM" innerhalb des Unterelements <Customer> suchen wollen, können Sie eine strukturelle Textsuche verwenden. Das Dokument kann auch die Zeichenfolge IBM in einem Unterelement <Comment> oder als Name eines Produktteils enthalten. Bei der strukturellen Suche wird nur in den angegebenen Elementen nach der Zeichenfolge gesucht. In diesem Beispiel werden nur die Dokumente gefunden, die IBM im Unterelement </Order/Customer/Name> enthalten; Dokumente, die die Zeichenfolge IBM in anderen Elementen, nicht jedoch in </Order/Customer/Name> enthalten, werden nicht zurückgegeben.

Volltextsuche

Sucht nach Textzeichenfolgen überall in der Dokumentstruktur, ohne Rücksicht auf Elemente oder Attribute. Im vorigen Beispiel werden alle Dokumente zurückgegeben, die die Zeichenfolge IBM enthalten, unabhängig davon, wo diese Zeichenfolge vorkommt.

Zur Verwendung der Suche mit dem Text Extender müssen Sie den DB2 Text Extender installieren und Ihre Datenbank und Ihre Tabellen wie nachfolgend beschrieben aktivieren. Eine Beschreibung der Vorgehensweise bei der Suche mit dem Text Extender finden Sie in dem Kapitel zum Suchen mit den UDFs des Text Extender im Handbuch *DB2 Universal Database Text Extender Verwaltung und Programmierung*.

Eine XML-Spalte für den Text Extender aktivieren

Angenommen, Sie haben eine für XML aktivierte Datenbank, führen Sie die folgenden Schritte aus, um den Text Extender für das Durchsuchen des Inhalts einer für XML aktivierten Spalte zu aktivieren. Als Beispiel dienen eine Datenbank mit dem Namen SALES_DB, die Tabelle mit dem Namen ORDER und die XML-Spalten XVARCHAR und XCLOB:

1. Schlagen Sie in der Datei `install.txt` auf der Extenders-CD nach, wie der Text Extender installiert werden muß.
2. Geben Sie den Befehl **txstart** an einer der folgenden Stellen ein:
   - Bei UNIX-Betriebssystemen geben Sie den Befehl über die Eingabeaufforderung des Exemplareigners ein.
   - Bei Windows NT geben Sie den Befehl in dem Befehlsfenster ein, in dem DB2INSTANCE angegeben wurde.
3. Öffnen Sie das Befehlszeilenfenster des Text Extender. Bei diesem Schritt wird davon ausgegangen, daß Sie eine Datenbank mit dem Namen SALES_DB und eine Tabelle mit dem Namen ORDER haben, die die beiden XML-Spalten XVARCHAR und XCLOB enthält. Eventuell müssen Sie die Beispielprogramme unter `dxx\samples\c` ausführen.
4. Stellen Sie die Verbindung zur Datenbank her. Geben Sie bei der Eingabeaufforderung **db2tx** folgendes ein:

   ```
   'connect to SALES_DB'
   ```

5. Aktivieren Sie die Datenbank für den Text Extender.

   Geben Sie bei der Eingabeaufforderung **db2tx** folgendes ein:

   ```
   'enable database'
   ```

6. Aktivieren Sie die Spalten in der XML-Tabelle für den Text Extender. Definieren Sie die Datentypen des XML-Dokuments, der Sprache, der Codepages und anderer Informationen zu der Spalte.
   - Geben Sie für die VARCHAR-Spalte XVARCHAR folgendes ein:

     ```
     'enable text column order xvarchar function db2xml.varchartovarchar handle
     varcharhandle ccsid 850 language us_english format xml indextype precise
     indexproperty sections_enabled
     documentmodel (Order) updateindex update'
     ```

- Geben Sie für die CLOB-Spalte XCLOB folgendes ein:

  ```
  'enable text column order xclob function db2xml.clob handle clobhandle
   ccsid 850 language us_english indextype precise updateindex update'
  ```

7. Überprüfen Sie den Status des Index.
   - Geben Sie für die Spalte XVARCHAR folgendes ein: `get index status order handle varcharhandle`
   - Geben Sie für die Spalte XCLOB folgendes ein: `get index status order handle clobhandle`
8. Definieren Sie das XML-Dokumentmodell in einer Dokumentmodell-INI-Datei mit dem Namen `desmodel.ini`. Diese Datei befindet sich für UNIX in `/db2tx/txins000` und für Windows NT in `\instance\\db2tx\txins000`; sie ist in einzelne Abschnitte einer Initialisierungsdatei untergliedert. Beispiel der Datei `textmodel.ini`:

   ```
   ;list of document models
   [MODELS]
   modelname=Order

   ; an 'Order' document model definition
   ; left side = section name identifier
   ; right side = section name tag

   [Order]
   Order = /Order
   Order/Customer/Name = /Order/Customer/Name
   Order/Customer/Email = /Order/Customer/Email
   Order/Part/@color = /Order/Part/@color
   Order/Part/Shipment/ShipMode = /Order/Part/Shipment/ShipMode
   ```

Mit dem Text Extender nach Text suchen

Die Suchfunktion des Text Extender funktioniert gut mit der strukturellen Dokumentsuche des XML Extender. Die empfohlene Methode ist das Erstellen einer Abfrage, die nach dem Dokumentelement oder den Attributen sucht und mit dem Text Extender den Elementinhalt oder die Attributwerte sucht.

Beispiel: Die folgenden Anweisungen suchen einen XML-Dokumenttext mit dem Text Extender. Geben Sie im DB2-Befehlsfenster folgendes ein:

```
'connect to SALES_DB'
'select xvarchar from order where db2tx.contains(varcharhandle,
 'model Order section(Order/Customer/Name) "Motors"')=1'
'select xclob from order where db2tx.contains(clobhandle,
 'model Order section(Order/Customer/Name) "Motors"')=1'
```

Die UDF Contains() des Text Extender führt eine Suche durch.

Dieses Beispiel enthält nicht alle Schritte, die zum Durchsuchen von Spaltendaten mit dem Text Extender erforderlich sind.

Eine Beschreibung der Vorgehensweise bei der Suche mit dem Text Extender finden Sie in dem Kapitel zum Suchen mit den UDFs des Text Extender im Handbuch *DB2 Universal Database Text Extender Verwaltung und Programmierung*.

XML-Dokumente löschen

Verwenden Sie die SQL-Anweisung SQL DELETE zum Löschen einer XML-Dokumentzeile aus einer XML-Spalte. Sie können WHERE-Klauseln angeben, um die Angabe der zu löschenden Dokumente einzugrenzen.

Beispiel: Die folgenden Anweisungen löschen alle Dokumente, die für <ExtendedPrice> einen Wert größer als 2500.00 haben.

```
DELETE from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

Einschränkungen beim Aufruf von Funktionen von JDBC aus

Beim Verwenden von Parametermarken in Funktionen erfordert eine JDBC-Einschränkung, daß die Parametermarke für die Funktion in den Datentyp der Spalte umgesetzt wird, in die die zurückgegebenen Daten eingefügt werden sollen. Die Logik der Funktionsauswahl weiß nicht, welchen Datentyp das Argument letztendlich hat und kann die Referenz nicht auflösen.

Als Ergebnis kann JDBC den folgenden Code nicht auflösen:

```
db2xml.XMLstandardumsetzungsfunktion(länge)
```

Sie können die CAST-Spezifikation verwenden, um einen Typ wie beispielsweise VARCHAR für die Parametermarke bereitzustellen; anschließend kann die Funktionslogik fortfahren:

```
db2xml.XMLstandardumsetzungsfunktion(CAST(? AS umsetzungstyp(länge))
```

Beispiel 1: Im folgenden Beispiel wird die Parametermarke als VARCHAR umgesetzt. Der übergebene Parameter ist ein XML-Dokument, der als VARCHAR(1000) übergeben und in die Spalte ORDER eingefügt wird.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarchar(cast (? as varchar(1000))))";
```

Beispiel 2: Im folgenden Beispiel wird die Parametermarke als VARCHAR umgesetzt. Der übergebene Parameter ist ein Dateiname; der Inhalt wird in VARCHAR umgesetzt und in die Spalte ORDER eingefügt.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarcharfromFILE(cast (? as varchar(1000))))";
```

Kapitel 6. XML-Objektgruppendaten verwalten

Eine XML-Objektgruppe ist eine Gruppe relationaler Tabellen mit Daten, die XML-Dokumenten zugeordnet sind. Mit dieser Zugriffs- und Speichermethode können Sie aus vorhandenen Daten ein XML-Dokument zusammensetzen, ein XML-Dokument zerlegen und XML als Austauschmethode verwenden.

Die relationalen Tabellen können neue Tabellen sein, die der XML Extender beim Zerlegen von XML-Dokumenten generiert, oder vorhandene Tabellen mit Daten, die mit dem XML Extender zum Generieren von XML-Dokumenten für Ihre Anwendungen verwendet werden können. Spaltendaten in diesen Tabellen enthalten keine XML-Befehle, sondern nur den Inhalt und die Werte, die den Elementen und Attributen zugeordnet sind. Gespeicherte Prozeduren agieren als Zugriffs- und Speichermethode zum Speichern, Abrufen, Aktualisieren, Suchen und Löschen von XML-Objektgruppendaten.

Die Parameterbegrenzungen für die gespeicherten Prozeduren der XML-Objektgruppe sind in „Anhang D. Die XML Extender-Begrenzungen" auf Seite 305 beschrieben.

Sie können die CLOB-Größen für die Ergebnisse der gespeicherten Prozeduren vergrößern; dies wird im Abschnitt „Die CLOB-Begrenzung vergrößern" auf Seite 222 beschrieben.

In den folgenden Abschnitten finden Sie Informationen zum Verwalten Ihrer XML-Objektgruppen:

- „XML-Dokumente aus DB2-Daten zusammensetzen"
- „XML-Dokumente in DB2-Daten zerlegen" auf Seite 156

XML-Dokumente aus DB2-Daten zusammensetzen

"Zusammensetzen" heißt das Generieren einer Gruppe von XML-Dokumenten aus relationalen Daten in einer XML-Objektgruppe. Sie können XML-Dokumente mit Hilfe gespeicherter Prozeduren zusammensetzen. Zur Verwendung dieser gespeicherten Prozeduren müssen Sie eine DAD-Datei erstellen, die die Zuordnung zwischen dem XML-Dokument und der DB2-Tabellenstruktur angibt. Die gespeicherten Prozeduren verwenden die DAD-Datei zum Zusammensetzen des XML-Dokuments. Im Abschnitt „XML-Objektgruppen planen" auf Seite 59 finden Sie Informationen zum Erstellen einer DAD-Datei.

Vorbereitungen

- Ordnen Sie die Struktur des XML-Dokuments den relationalen Tabellen zu, die den Inhalt der Element- und Attributwerte enthalten.
- Wählen Sie eine Zuordnungsmethode aus: SQL-Zuordnung oder RDB_node-Zuordnung.
- Bereiten Sie die DAD-Datei vor. Ausführliche Informationen hierzu finden Sie im Abschnitt „XML-Objektgruppen planen" auf Seite 59.
- Aktivieren Sie wahlweise die XML-Objektgruppe.

XML-Dokument zusammensetzen

Der XML Extender bietet zwei gespeicherte Prozeduren, dxxGenXML() und dxxRetrieveXML(), zum Zusammensetzen von XML-Dokumenten.

dxxGenXML()

Diese gespeicherte Prozedur wird für Anwendungen verwendet, die gelegentliche Aktualisierungen vornehmen, oder für Anwendungen, bei denen der Systemaufwand für die Verwaltung der XML-Daten eingespart werden soll. Die gespeicherte Prozedur dxxGenXML() erfordert keine aktivierte Objektgruppe und verwendet statt dessen eine DAD-Datei.

Die gespeicherte Prozedur dxxGenXML() konstruiert XML-Dokumente mit Hilfe von Daten in XML-Objektgruppentabellen, die über das Element <Xcollection> in der DAD-Datei angegeben sind. Diese gespeicherte Prozedur fügt jedes XML-Dokument als Zeile in eine *Ergebnistabelle* ein. Sie können auch einen Cursor in der Ergebnistabelle öffnen und die Ergebnisgruppe abrufen. Die Ergebnistabelle muß von der Anwendung erstellt werden; sie hat immer eine Spalte des Typs VARCHAR, CLOB, XMLVARCHAR oder XMLCLOB.

Wenn Sie das Element für die Prüfung in der DAD-Datei auf YES setzen, fügt der XML Extender außerdem die Spalte DXX_VALID des Typs INTEGER hinzu und fügt den Wert 1 für ein gültiges XML-Dokument und 0 für ein ungültiges Dokument hinzu.

Die gespeicherte Prozedur dxxGenXML() ermöglicht außerdem die Angabe der maximalen Anzahl von Zeilen, die in der Ergebnistabelle generiert werden sollen. Auf diese Weise wird die Verarbeitungszeit verkürzt. Die gespeicherte Prozedur gibt die tatsächliche Anzahl von Zeilen in der Tabelle zusammen mit allen eventuellen Rückkehrcodes und Nachrichten zurück.

Die entsprechende gespeicherte Prozedur für das Zerlegen ist dxxShredXML(); sie verwendet ebenfalls die DAD-Datei als Eingabeparameter. Für diese Prozedur braucht die XML-Objektgruppe nicht aktiviert zu sein.

So erstellen Sie eine XML-Objektgruppe: dxxGenXML()

Integrieren Sie den Aufruf einer gespeicherten Prozedur in Ihrer Anwendung mit der folgenden Deklaration einer gespeicherten Prozedur:

```
dxxGenXML(CLOB(100K)     DAD,             /* Eingabe */
          char(resultTabName) resultTabName,/* Eingabe */
          integer        overrideType,    /* Eingabe */
          varchar(1024)  override,        /* Eingabe */
          integer        maxRows,         /* Eingabe */
          integer        numRows,         /* Ausgabe */
          long           returnCode,      /* Ausgabe */
          varchar(1024)  returnMsg)       /* Ausgabe */
```

Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „dxxGenXML()" auf Seite 232.

Beispiel: Mit dem folgenden Beispiel wird ein XML-Dokument zusammengesetzt:

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB(100K) dad;        /* DAD */
     SQL TYPE is CLOB_FILE  dadFile;    / * DAD-Datei */
     char             result_tab[32];   /* Name der Ergebnistabelle */
     char             override[2];      /* Überschr., auf NULL gesetzt*/
     short            overrideType;     /* definiert in dxx.h */
     short            max_row;          /* maximale Anzahl Zeilen */
     short            num_row;          /* tatsächl. Anzahl Zeilen */
     long             returnCode;       /* Rückkehr-Fehlercode */
     char             returnMsg[1024];  /* Fehlernachrichtentext */
     short            dad_ind;
     short            rtab_ind;
     short            ovtype_ind;
     short            ov_inde;
     short            maxrow_ind;
     short            numrow_ind;
     short            returnCode_ind;
     short            returnMsg_ind;

EXEC SQL END DECLARE SECTION;

     /* Tabelle erstellen */
     EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
     /* Daten aus einer Datei in ein CLOB einlesen */
     strcpy(dadfile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
     dadfile.name_length = strlen("c:\dxx\samples\dad\getstart_
     xcollection.dad");
     dadfile.file_options = SQL_FILE_READ;
     EXEC SQL VALUES (:dadfile) INTO :dad;
     strcpy(result_tab,"xml_order_tab");
```

```
override[0] = '\0';
overrideType = NO_OVERRIDE;
max_row = 500;
num_row = 0;
returnCode = 0;
msg_txt[0] = '\0';
collection_ind = 0;
dad_ind = 0;
rtab_ind = 0;
ov_ind = -1;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxGenXML(:dad:dad_ind;
     :result_tab:rtab_ind,
     :overrideType:ovtype_ind,:override:ov_ind,
     :max_row:maxrow_ind,:num_row:numrow_ind,
     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Die Ergebnistabelle enthält nach dem Aufrufen der gespeicherten Prozedur 250 Zeilen, da die in der DAD-Datei angegebene SQL-Abfrage 250 XML-Dokumente generiert hatte.

dxxRetrieveXML()

Diese gespeicherte Prozedur wird für Anwendungen verwendet, die regelmäßige Aktualisierungen vornehmen. Da dieselben Tasks wiederholt ausgeführt werden, ist die Verbesserung der Leistung von großer Bedeutung. Das Aktivieren einer XML-Objektgruppe und die Verwendung des Gruppennamens in der gespeicherten Prozedur verbessert die Leistung. Die gespeicherte Prozedur dxxRetrieveXML() funktioniert wie die gespeicherte Prozedur dxxGenXML() mit dem Unterschied, daß sie den Namen einer aktivierten XML-Objektgruppe statt einer DAD-Datei verwendet. Wenn eine XML-Objektgruppe aktiviert ist, wird eine DAD-Datei in der Tabelle XML_USAGE gespeichert. Der XML Extender ruft daher die DAD-Datei ab; von dieser Stelle ab ist dxxRetrieveXML() identisch mit dxxGenXML().

dxxRetrieveXML() ermöglicht die Verwendung derselben DAD-Datei beim Zusammensetzen und Zerlegen. Die gespeicherte Prozedur kann auch zum Abrufen zerlegter XML-Dokumente verwendet werden. Die entsprechende gespeicherte Prozedur für das Zerlegen ist dxxInsertXML(); sie verwendet ebenfalls den Namen einer aktivierten XML-Objektgruppe.

So erstellen Sie eine XML-Objektgruppe: dxxRetrieveXML()

Integrieren Sie den Aufruf einer gespeicherten Prozedur in Ihrer Anwendung mit der folgenden Deklaration einer gespeicherten Prozedur:

```
dxxRetrieveXML(char(collectionName) collectionName, /* Eingabe */
               char(resultTabName) resultTabName,   /* Eingabe */
               integer        overrideType,    /* Eingabe */
               varchar(1024)  override,        /* Eingabe */
               integer        maxRows,         /* Eingabe */
               integer        numRows,         /* Ausgabe */
               long           returnCode,      /* Ausgabe */
               varchar(1024)  returnMsg)       /* Ausgabe */
```

Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „dxxRetrieveXML()" auf Seite 236.

Beispiel: Das folgende Beispiel zeigt einen Aufruf von dxxRetrieveXML(). Dabei wird davon ausgegangen, daß die Ergebnistabelle mit dem Namen XML_ORDER_TAB erstellt wird und eine Spalte des Typs XMLVARCHAR enthält.

```
#include "dxx.h"
#include "dxxrc.h"

            EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
             char   collection[32]; /* DAD-Puffer */
             char   result_tab[32]; /* Name der Ergebnistabelle */
             char   override[2];    /* Überschreiben, auf NULL gesetzt*/
             short  overrideType;   /* definiert in dxx.h */
             short  max_row;        /* maximale Anzahl Zeilen */
             short  num_row;        /* tatsächliche Anzahl Zeilen */
             long   returnCode;     /* Rückkehr-Fehlercode */
             char   returnMsg[1024];/* Fehlernachrichtentext */
             short  dadbuf_ind;
             short  rtab_ind;
             short  ovtype_ind;
             short  ov_inde;
             short  maxrow_ind;
             short  numrow_ind;
             short  returnCode_ind;
             short  returnMsg_ind;

             EXEC SQL END DECLARE SECTION;

             /* Tabelle erstellen */
             EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

             /* Host-Variable und Anzeiger initialisieren  */
             strcpy(collection,"sales_ord");
             strcpy(result_tab,"xml_order_tab");
             override[0] = '\0';
```

```
overrideType = NO_OVERRIDE;
max_row = 500;
num_row = 0;
returnCode = 0;
msg_txt[0] = '\0';
collection_ind = 0;
rtab_ind = 0;
ov_ind = -1;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
       :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
       :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Werte in der DAD-Datei dynamisch überschreiben

Für dynamische Abfragen können Sie zwei wahlfreie Parameter zum Überschreiben der Bedingungen in der DAD-Datei verwenden: *override* und *overrideType*. Entsprechend der Eingabe von *overrideType* kann die Anwendung die Befehlswerte <SQL_stmt> für die SQL-Zuordnung überschreiben oder die Bedingungen in RDB_nodes für die RDB_node-Zuordnung in der DAD-Datei.

Diese Parameter haben folgende Werte und Regeln:

overrideType
Dieser Parameter ist ein erforderlicher Eingabeparameter (IN), der den Typ des Parameters *override* angibt. *overrideType* kann die folgenden Werte haben:

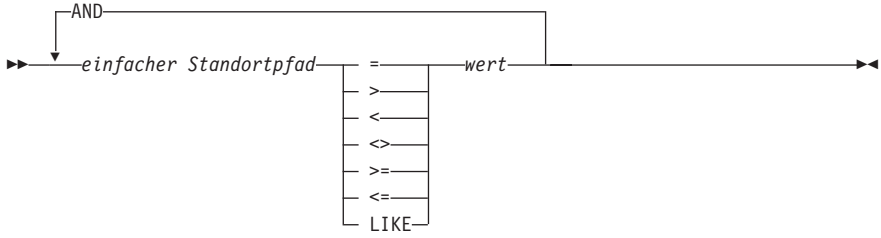
NO_OVERRIDE
Gibt an, daß eine Bedingung in der DAD-Datei nicht überschrieben werden soll.

SQL_OVERRIDE
Gibt an, daß eine Bedingung in der DAD-Datei durch eine SQL-Anweisung überschrieben werden soll.

XML_OVERRIDE
Gibt an, daß eine Bedingung in der DAD-Datei durch eine XPath-Bedingung überschrieben werden soll.

override
Dieser Parameter ist ein wahlfreier Eingabeparameter (IN), der die override-Bedingung für die DAD-Datei angibt. Die Syntax für den Eingabewert entspricht dem für *overrideType* angegebenen Wert.

- Wenn Sie NO_OVERRIDE angeben, ist der Eingabewert eine Nullzeichenfolge.
- Wenn Sie SQL_OVERRIDE angeben, ist der Eingabewert eine gültige SQL-Anweisung. ***Erforderlich:*** Wenn Sie SQL_OVERRIDE und eine SQL-Anweisung verwenden, müssen Sie in der DAD-Datei das SQL-Zuordnungsschema verwenden. Die SQL-Eingabeanweisung überschreibt die im Element <SQL_stmt> in der DAD-Datei angegebene SQL-Anweisung.
- Wenn Sie XML_OVERRIDE verwenden, ist der Eingabewert eine Zeichenfolge, die ein oder mehrere Ausdrücke enthält. ***Erforderlich:*** Wenn Sie XML_OVERRIDE und einen Ausdruck verwenden, müssen Sie in der DAD-Datei das RDB_node-Zuordnungsschema verwenden. Der XML-Eingabeausdruck überschreibt die in der DAD-Datei angegebene RDB_node-Bedingung. Der Ausdruck verwendet die folgende Syntax:



Hierbei gilt folgendes:

einfacher Standortpfad
Ein einfacher Standortpfad mit der von XPATH definierten Syntax; die Syntax ist in Tabelle 5 auf Seite 58 beschrieben.

Operatoren
Kann ein Leerzeichen enthalten, um den Operator von den anderen Teilen des Ausdrucks zu trennen.

wert
Ein numerischer Wert oder ein Zeichenfolge in einfachen Anführungszeichen.

Sie können um die Operationen herum Leerzeichen eingeben; um den Operator LIKE sind Leerzeichen erforderlich.

Bei Angabe des Werts XML_OVERRIDE wird die Bedingung für den RDB_node im text_node oder attribute_node, die dem einfachen Standortpfad entspricht, von dem angegebenen Ausdruck überschrieben.

XML_OVERRIDE ist nicht vollständig XPath-kompatibel. Der einfache Standortpfad wird nur zum Kennzeichnen des Elements oder Attributs, das einer Spalte zugeordnet ist, verwendet.

Beispiele:

Die folgenden Beispiele zeigen ein dynamisches Überschreiben mit SQL_OVERRIDE und XML_OVERRIDE. Die meisten gespeicherten Prozeduren in diesem Handbuch verwenden NO_OVERRIDE.

Beispiel: Eine gespeicherte Prozedur, die SQL_OVERRIDE verwendet.

```
include "dxx.h"
include "dxxrc.h"

       EXEC SQL INCLUDE SQLCA;
       EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];    /* DAD-Puffer */
         char    result_tab[32];    /* Name der Ergebnistabelle */
         char    override[256];     /* überschreiben, SQL_stmt */
         short   overrideType;      /* definiert in dxx.h */
         short   max_row;           /* maximale Anzahl Zeilen */
         short   num_row;           /* tatsächliche Anzahl Zeilen */
         long    returnCode;        /* Rückkehr-Fehlercode */
         char    returnMsg[1024];   /* Fehlernachrichtentext */
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short   returnCode_ind;
         short   returnMsg_ind;

         EXEC SQL END DECLARE SECTION;

         /* Tabelle erstellen */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

         /* Host-Variable und Anzeiger initialisieren  */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
         sprintf(override,"%s %s %s %s %s %s %s",
                   "SELECT o.order_key, customer, p.part_key, quantity, price,",
                   "tax, ship_id, date, mode ",
                   "FROM order_tab o, part_tab p,",
                   "table(select substr(char(timestamp(generate_unique())),16",
                   "as ship_id,date,mode from ship_tab)as s",
                   "WHERE p.price > 50.00 and s.date >'1998-12-01' AND",
                   "p.order_key = o.order_key and s.part_key = p.part_key");
        overrideType = SQL_OVERRIDE;
         max_row = 500;
         num_row = 0;
         returnCode = 0;
         msg_txt[0] = '\0';
```

```
collection_ind = 0;
rtab_ind = 0;
ov_ind = 0;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
       :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
       :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

In diesem Beispiel muß das Element <xcollection> in der DAD-Datei ein Element <SQL_stmt> enthalten. Der Parameter *override* überschreibt den Wert von <SQL_stmt> durch Ändern des Preises auf einen Wert größer als 50.00, und das Datum wird auf Wert größer als 1998-12-01 geändert.

Beispiel: Eine gespeicherte Prozedur, die XML_OVERRIDE verwendet.

```
include "dxx.h"
include "dxxrc.h"

       EXEC SQL INCLUDE SQLCA;
       EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* DAD-Puffer */
         char    result_tab[32];  /* Name der Ergebnistabelle */
         char    override[256];   /* überschreiben, SQL_stmt */
         short   overrideType;    /* definiert in dxx.h */
         short   max_row;         /* maximale Anzahl Zeilen */
         short   num_row;         /* tatsächliche Anzahl Zeilen */
         long    returnCode;      /* Rückkehr-Fehlercode */
         char    returnMsg[1024]; /* Fehlernachrichtentext */
         short   dadbuf_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short   returnCode_ind;
         short   returnMsg_ind;

         EXEC SQL END DECLARE SECTION;

         /* Tabelle erstellen */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

         /* Host-Variable und Anzeiger initialisieren  */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
         sprintf(override,"%s %s",
```

```
            "/Order/Part/Price > 50.00 AND ",
            "Order/Part/Shipment/ShipDate > '1998-12-01'");
overrideType = XML_OVERRIDE;
 max_row = 500;
 num_row = 0;
 returnCode = 0;
 msg_txt[0] = '\0';
 collection_ind = 0;
 rtab_ind = 0;
 ov_ind = 0;
 ovtype_ind = 0;
 maxrow_ind = 0;
 numrow_ind = -1;
 returnCode_ind = -1;
 returnMsg_ind = -1;

 /* Gespeicherte Prozedur aufrufen */
 EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
        :result_tab:rtab_ind,
        :overrideType:ovtype_ind,:override:ov_ind,
        :max_row:maxrow_ind,:num_row:numrow_ind,
        :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

In diesem Beispiel hat das Element <collection> in der DAD-Datei einen RDB_node für den Root-element_node. Der Wert für *override* hängt vom XML-Wert ab. Der XML Extender wandelt den einfachen Standortpfad um in die zugeordnete DB2-Spalte um.

XML-Dokumente in DB2-Daten zerlegen

Das Zerlegen eines XML-Dokuments bedeutet, daß die Daten in einem XML-Dokument heruntergebrochen und in relationalen Tabellen gespeichert werden. Der XML Extender bietet gespeicherte Prozeduren zum Zerlegen von XML-Daten aus XML-Quellendokumenten in relationale Tabellen. Zur Verwendung dieser gespeicherten Prozeduren müssen Sie eine DAD-Datei erstellen, die die Zuordnung zwischen dem XML-Dokument und der DB2-Tabellenstruktur angibt. Die gespeicherten Prozeduren verwenden die DAD-Datei zum Zerlegen des XML-Dokuments. Im Abschnitt „XML-Objektgruppen planen” auf Seite 59 finden Sie Informationen zum Erstellen einer DAD-Datei.

XML-Objektgruppe zum Zerlegen aktivieren

In den meisten Fällen müssen Sie eine XML-Objektgruppe aktivieren, bevor Sie die gespeicherte Prozedur verwenden können. In den folgenden Fällen müssen Sie eine XML-Objektgruppe aktivieren:

- Beim Zerlegen von XML-Dokumenten in neue Tabellen muß eine XML-Objektgruppe aktiviert sein, da alle Tabellen in der XML-Objektgruppe vom XML Extender beim Aktivieren der Objektgruppe erstellt werden.
- Wenn die Beibehaltung der Reihenfolge von Elementen und Attributen wichtig ist. Der XML Extender behält die Reihenfolge von Elementen und

Attributen mit mehrfachem Vorkommen nur für Tabellen bei, die beim Aktivieren einer Objektgruppe erstellt werden. Beim Zerlegen von XML-Dokumenten in vorhandene relationale Tabellen kann die Beibehaltung der Reihenfolge nicht garantiert werden.

Wenn Sie die DAD-Datei spontan übergeben wollen, wenn die Tabellen bereits in der Datenbank vorhanden sind, brauchen Sie keine XML-Objektgruppe zu aktivieren.

Einschränkungen für die Tabellengröße beim Zerlegen

Beim Zerlegen wird die RDB_node-Zuordnung verwendet zur Angabe, wie ein XML-Dokument in DB2-Tabellen zerlegt werden soll. Hierbei werden die Element- und Attributwerte in Tabellenzeilen extrahiert. Die Werte aus den einzelnen XML-Dokumenten werden in einer oder mehreren DB2-Tabellen gespeichert. Jede Tabelle kann maximal 1024 Zeilen aus jedem Dokument enthalten.

Wenn ein XML-Dokument beispielsweise in fünf Tabellen zerlegt wird, kann jede dieser fünf Tabellen bis zu 1024 Zeilen für dieses bestimmte Dokument haben. Wenn die Tabelle Zeilen für mehrere Dokumente enthält, kann sie bis zu 1024 Zeilen für jedes Dokument enthalten. Wenn die Tabelle 20 Dokumente enthält, kann sie also 20.480 Zeilen enthalten, 1024 pro Dokument.

Die Verwendung mehrfach vorkommender Elemente (Elemente mit Standortpfaden, die mehr als einmal in der XML-Struktur vorkommen) wirkt sich auf die Anzahl der Zeilen aus. Ein Dokument, das beispielsweise ein Element <Part> 20 mal enthält, kann als 20 Zeilen in eine Tabelle zerlegt werden. Beachten Sie bei Verwendung mehrfach vorkommender Elemente diese Einschränkung der Tabellengröße.

Vorbereitungen

- Ordnen Sie die Struktur des XML-Dokuments den relationalen Tabellen zu, die den Inhalt der Element- und Attributwerte enthalten.
- Bereiten Sie die DAD-Datei mit der RDB_node-Zuordnung vor. Siehe hierzu den Abschnitt „XML-Objektgruppen planen" auf Seite 59.
- Aktivieren Sie wahlweise die XML-Objektgruppe.

XML-Dokument zerlegen

Der XML Extender bietet zwei gespeicherte Prozeduren, dxxShredXML() und dxxInsertXML(), zum Zerlegen von XML-Dokumenten.

dxxShredXML()

Diese gespeicherte Prozedur wird für Anwendungen verwendet, die gelegentliche Aktualisierungen vornehmen, oder für Anwendungen, bei denen der Systemaufwand für die Verwaltung der XML-Daten ein-

gespart werden soll. Die gespeicherte Prozedur dxxShredXML() erfordert keine aktivierte Objektgruppe und verwendet statt dessen eine DAD-Datei.

Die gespeicherte Prozedur dxxShredXML() verwendet zwei Eingabeparameter: eine DAD-Datei und das zu zerlegende XML-Dokument; sie gibt zwei Ausgabeparameter aus: einen Rückkehrcode und eine Rückkehrnachricht.

Die gespeicherte Prozedur dxxShredXML() fügt entsprechend der Angabe <Xcollection> in der DAD-Datei ein XML-Dokument in eine XML-Objektgruppe ein. Es wird davon ausgegangen, daß die im Element <Xcollection> der DAD-Datei verwendeten Tabellen vorhanden sind und daß die Spalten den in der DAD-Zuordnung angegebenen Datentypen entsprechen. Ist dies nicht der Fall, wird eine Fehlernachricht zurückgegeben. Die gespeicherte Prozedur dxxShredXML() zerlegt anschließend das XML-Dokument und fügt unmarkierte XML-Daten in die in der DAD-Datei angegebenen Tabellen ein.

Die entsprechende gespeicherte Prozedur für das Zusammensetzen ist dxxGenXML(); sie verwendet ebenfalls die DAD-Datei als Eingabeparameter. Für diese Prozedur braucht die XML-Objektgruppe nicht aktiviert zu sein.

So zerlegen Sie eine XML-Objektgruppe: dxxShredXML()

Integrieren Sie den Aufruf einer gespeicherten Prozedur in Ihrer Anwendung mit der folgenden Deklaration einer gespeicherten Prozedur:

```
dxxShredXML(CLOB(100K)    DAD,           /* Eingabe */
            CLOB(1M)      xmlobj,        /* Eingabe */
            long          returnCode,    /* Ausgabe */
            varchar(1024) returnMsg)     /* Ausgabe */
```

Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „dxxShredXML()” auf Seite 241.

Beispiel: Das folgende Beispiel zeigt einen Aufruf von dxxShredXML():

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
     EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB    dad;             /* DAD*/
     SQL TYPE is CLOB_FILE dadFile;       /* DAD-Datei */
     SQL TYPE is CLOB    xmlDoc;          /* XML-Eingabedokument */
     SQL TYPE is CLOB_FILE xmlFile;       /* Eingabe-XML-Datei */
     long                returnCode;      /* Fehlercode */
     char                returnMsg[1024]; /* Fehlernachrichtentext */
```

```
short               dad_ind;
short               xmlDoc_ind;
short               returnCode_ind;
short               returnMsg_ind;
EXEC SQL END DECLARE SECTION;

/* Host-Variable und Anzeiger initialisieren  */
strcpy(dadFile.name,"c:\dxx\samples\dad\
getstart_xcollection.dad");
dadFile.name_length=strlen("c:\dxx\samples\dad\
getstart_xcollection.dad");
dadFile.file_option=SQL_FILE_READ;
strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart_
xcollection.xml");
xmlFile.name_length=strlen("c:\dxx\samples\cmd\
getstart_xcollection.xml");
xmlFile.file_option=SQL_FILE_READ;
SQL EXEC VALUES (:dadFile) INTO :dad;
SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
returnCode = 0;
returnMsg[0] = '\0';
dad_ind = 0;
xmlDoc_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
           :xmlDoc:xmlDoc_ind,
           :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

Diese gespeicherte Prozedur wird für Anwendungen verwendet, die regelmäßige Aktualisierungen vornehmen. Da dieselben Tasks wiederholt ausgeführt werden, ist die Verbesserung der Leistung von großer Bedeutung. Das Aktivieren einer XML-Objektgruppe und die Verwendung des Gruppennamens in der gespeicherten Prozedur verbessert die Leistung. Die gespeicherte Prozedur dxxInsertXML() funktioniert genau wie dxxShredXML() mit dem Unterschied, daß dxxInsertXML() eine aktivierte XML-Objektgruppe als ersten Eingabeparameter verwendet.

Die gespeicherte Prozedur dxxInsertXML() fügt ein XML-Dokument in eine aktivierte XML-Objektgruppe, die einer DAD-Datei zugeordnet ist, ein. Die DAD-Datei enthält Spezifikationen für die Gruppentabellen und die Zuordnung. Die Gruppentabellen werden entsprechend den Spezifikationen in der <Xcollection> geprüft oder erstellt. Die gespeicherte Prozedur dxxInsertXML() zerlegt anschließend das XML-Dokument entsprechend der Zuordnung und fügt unmarkierte XML-Daten in die Tabellen der benannten XML-Objektgruppe ein.

Die entsprechende gespeicherte Prozedur für das Zusammensetzen ist dxxRetrieveXML(); sie verwendet ebenfalls den Namen einer aktivierten XML-Objektgruppe.

So zerlegen Sie eine XML-Objektgruppe: dxxInsertXML()

Integrieren Sie den Aufruf einer gespeicherten Prozedur in Ihrer Anwendung mit der folgenden Deklaration einer gespeicherten Prozedur:

```
dxxInsertXML(char(collectionName) collectionName,  /* Eingabe */
             CLOB(1M)       xmlobj,                /* Eingabe */
             long           returnCode,            /* Ausgabe */
             varchar(1024)  returnMsg)             /* Ausgabe */
```

Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „dxxInsertXML()" auf Seite 243.

Beispiel: Das folgende Beispiel zeigt einen Aufruf von dxxInsertXML():

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
     EXEC SQL BEGIN DECLARE SECTION;
      char           collection[64];  /* Name einer XML-Objektgruppe */
      SQL TYPE is CLOB_FILE xmlFile;  /* Eingabe-XML-Datei */
      SQL TYPE is CLOB  xmlDoc;       /* XML-Eingabedokument */
      long           returnCode;      /* Fehlercode */
      char           returnMsg[1024]; /* Fehlernachrichtentext */
      short          collection_ind;
      short          xmlDoc_ind;
      short          returnCode_ind;
      short          returnMsg_ind;
      EXEC SQL END DECLARE SECTION;

         /* Host-Variable und Anzeiger initialisieren  */
         strcpy(collection,"sales_ord")
         strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart_
         xcollection.xml");
         xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart_
         xcollection.xml");
         xmlobj.file_option=SQL_FILE_READ;
         SQL EXEC VALUES (:xmlFile) INTO (:xmlDoc);
         returnCode = 0;
         returnMsg[0] = '\0';
         collection_ind = 0;
         xmlobj_ind = 0;
         returnCode_ind = -1;
         returnMsg_ind = -1;

         /* Gespeicherte Prozedur aufrufen */
         EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
                     :xmlDoc:xmlDoc_ind,
                     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Eine XML-Objektgruppe aufrufen

Sie können XML-Objektgruppen aktualisieren, löschen, durchsuchen und abrufen. Denken Sie daran, daß der Zweck der Verwendung einer XML-Objektgruppe das Speichern oder Abrufen unmarkierter, reiner Daten in Datenbanktabellen ist. Die Daten in den vorhandenen Datenbanktabellen haben nichts mit den ankommenden XML-Dokumenten zu tun; die Aktionen zum Aktualisieren, Löschen und Durchsuchen bestehen lediglich aus normalen SQL-Zugriffen auf diese Tabellen. Wenn die Daten aus eingehenden XML-Dokumenten zerlegt werden, sind die ursprünglichen XML-Dokumente nicht mehr vorhanden.

Der XML Extender bietet die Möglichkeit, von der XML-Objektgruppensicht aus Operationen mit den Daten vorzunehmen. Über die SQL-Anweisungen UPDATE und DELETE können Sie die für das Zusammensetzen von XML-Dokumenten verwendeten Daten ändern und somit die XML-Objektgruppe aktualisieren.

Überlegungen:

- Löschen Sie zum Aktualisieren eines Dokuments nicht die Zeile mit dem Primärschlüssel der Tabelle; dieser Schlüssel entspricht für die anderen Gruppentabellen der Zeile mit dem Fremdschlüssel. Wenn der Primärschlüssel und der Fremdschlüssel gelöscht werden, wird auch das Dokument gelöscht.
- Zum Ersetzen oder Löschen von Elementen und Attributwerten können Sie Zeilen in Tabellen einer niedrigeren Ebene löschen und einfügen, ohne das Dokument zu löschen.
- Zum Löschen eines Dokuments löschen Sie die Zeile, die den in der DAD-Datei angegebenen Anfangs-element_node bildet.

XML-Daten in einer XML-Objektgruppe aktualisieren

Der XML Extender ermöglicht das Aktualisieren nicht markierter Daten, die in XML-Objektgruppentabellen gespeichert sind. Durch das Aktualisieren von Tabellenwerten in XML-Objektgruppen aktualisieren Sie den Text eines XML-Elements oder den Wert eines XML-Attributs. Darüber hinaus kann bei mehrfachem Vorkommen der Elemente oder Attribute durch ein Aktualisieren ein Exemplar der Daten gelöscht werden.

Aus der Sicht von SQL ist das Ändern des Werts von Elementen oder Attributen ein Aktualisierungsvorgang; das Löschen eines Exemplars eines Elements oder Attributs ist dagegen ein Löschvorgang. Von XML aus betrachtet, ist das XML-Dokument weiterhin vorhanden, so lange der Elementtext bzw. der Attributwert existiert; somit handelt es sich aus dieser Sicht um ein Aktualisieren.

Voraussetzungen: Beachten Sie beim Aktualisieren von Daten in einer XML-Objektgruppe die folgenden Regeln.

- Geben Sie die Beziehung Primär-Fremdschlüssel zwischen den Gruppentabellen an, sofern die vorhandenen Tabellen eine solche Beziehung aufweisen. Liegt keine solche Beziehung vor, stellen Sie sicher, daß Spalten vorhanden sind, die verknüpft werden können.
- Geben Sie die in der DAD-Datei angegebene Verknüpfungsbedingung an:
  - Für SQL-Zuordnung in dem Element <SQL_stmt>
  - Für RDB_node-Zuordnung im RDB_node der Stammelementknotens

Elemente und Attributknoten aktualisieren

In einer XML-Objektgruppe sind alle Elementtexte und Attributwerte Spalten in Datenbanktabellen zugeordnet. Unabhängig davon, ob die Spaltendaten bereits vorher vorhanden sind oder aus ankommenden XML-Dokumenten zerlegt werden, ersetzen Sie die Daten über die übliche SQL-Aktualisierungstechnik.

Zum Aktualisieren eines Elements oder Attributwerts geben Sie eine WHERE-Klausel in der SQL-Anweisung UPDATE an, die die in der DAD-Datei angegebene Verknüpfungsbedingung enthält.

Beispiel:

```
UPDATE SHIP_TAB
    set MODE = 'BOAT'
  WHERE MODE='AIR' AND PART_KEY in
    (SELECT PART_KEY from PART_TAB WHERE ORDER_KEY=68)
```

Der Elementwert für <ShipMode> wird in der Tabelle SHIP_TAB von AIR in BOAT aktualisiert; der Schlüssel ist 68.

Exemplare von Elementen und Attributen löschen

Zum Aktualisieren zusammengesetzter XML-Dokumente durch Eliminieren mehrfach auftretender Elemente oder Attribute löschen Sie mit Hilfe der WHERE-Klausel eine Zeile mit dem Feldwert, der dem Element oder Attributwert entspricht. So lange Sie nicht die Zeile löschen, die die Werte für den Anfangs-element_node enthält, wird das Löschen von Werten als eine Aktualisierung des XML-Dokuments betrachtet.

In der folgenden DELETE-Anweisung löschen Sie beispielsweise ein Element <shipment> durch Angabe eines eindeutigen Werts eines seiner Unterelemente.

```
DELETE from SHIP_TAB
   WHERE DATE='1999-04-12'
```

Mit der Angabe eines DATE-Werts wird die Zeile, die diesem Wert entspricht, gelöscht. Das zusammengesetzte Dokument enthielt ursprünglich zwei Elemente <shipment>; jetzt enthält es nur noch ein solches Element.

XML-Dokument aus einer XML-Objektgruppe löschen

Sie können ein aus einer Objektgruppe zusammengesetztes XML-Dokument löschen. Dies bedeutet: Wenn Sie eine XML-Objektgruppe haben, die aus mehreren XML-Dokumenten besteht, können Sie eines dieser zusammengesetzten Dokumente löschen.

Zum Löschen des Dokuments löschen Sie eine Zeile in der Tabelle, die den DAD-Datei angegebenen Anfangs-element_node bildet. Diese Tabelle enthält den Primärschlüssel für die Gruppentabelle der Ausgangsebene und den Fremdschlüssel für die Tabellen der niedrigeren Ebenen.

Die folgende DELETE-Anweisung gibt den Wert der Primärschlüsselspalte an.

```
DELETE from order_tab
   WHERE order_key=1
```

ORDER_KEY ist der Primärschlüssel in der Tabelle ORDER_TAB und gleichzeitig der Anfangs-element_node beim Zusammensetzen des XML-Dokuments. Durch das Löschen dieser Zeile wird ein beim Zusammensetzen generiertes Dokument gelöscht. Aus der Sicht von XML wird das Dokument somit aus der XML-Objektgruppe gelöscht.

XML-Dokumente aus einer XML-Objektgruppe abrufen

Das Abrufen von XML-Dokumenten aus einer XML-Objektgruppe ähnelt dem Zusammensetzen von Dokumenten aus der Objektgruppe.

Verwenden Sie zum Abrufen von XML-Dokumenten die gespeicherte Prozedur dxxRetrieveXML(). Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „dxxRetrieveXML()" auf Seite 236.

Überlegungen zur DAD-Datei: Wenn Sie XML-Dokumente in eine XML-Objektgruppe zerlegen, kann die Reihenfolge mehrfach vorkommender Elemente und Attributwerte verlorengehen, es sei denn, Sie geben diese Reihenfolge in der DAD-Datei ein. Zur Beibehaltung dieser Reihenfolge müssen Sie das RDB_node-Zuordnungsschema verwenden. Dieses Zuordnungsschema ermöglicht die Angabe des Attributs orderBy für die Tabelle mit dem Stammelement in seinem RDB_node.

XML-Objektgruppe durchsuchen

In diesem Abschnitt wird das Durchsuchen einer XML-Objektgruppe mit folgenden Zielen beschrieben:

- **XML-Dokumente mit Suchkriterien generieren:**

  Diese Aufgabe entspricht dem Zusammensetzen mit einer Bedingung. Sie können die Suchkriterien wie folgt angeben:

  – Angeben der Bedingung im text_node und attribute_node der DAD-Datei

  – Angaben des Parameters *overwrite* bei der Verwendung der gespeicherten Prozeduren dxxGenXML() und dxxRetrieveXML().

  Wenn Sie beispielsweise eine XML-Objektgruppe sales_ord mit der DAD-Datei `order.dad` aktiviert haben und jetzt den Preis mit Formatdaten überschreiben wollen, die Sie aus dem Web abgerufen haben, können Sie den Wert des DAD-Elements <SQL_stmt> wie folgt überschreiben:

```
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
       ...

EXEC SQL END DECLARE SECTION;

         float    price_value;

         /* Tabelle erstellen */
         EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

         /* Host-Variable und Anzeiger initialisieren  */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
         overrideType = SQL_OVERRIDE;
         max_row = 20;
         num_row = 0;
         returnCode = 0;
         msg_txt[0] = '\0';
         override_ind = 0;
         overrideType_ind = 0;
         rtab_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
         returnCode_ind = -1;
         returnMsg_ind = -1;
```

```
/* price_value von irgendwoher abrufen, z. B. Formatdaten  */
price_value = 1000.00         /* Beispiel */

/* Überschreiben angeben */
sprintf(overwrite,
     "SELECT o.order_key, customer, p.part_key, quantity,
         price, tax, ship_id, date, mode
     FROM order_tab o, part_tab p,
         table(select substr(char(timestamp(generate_unique())),16)
         as ship_id, date, mode from ship_tab)as s
     WHERE p.price > %d and s.date >'1996-06-01' AND
         p.order_key = o.order_key and s.part_key = p.part_key",
         price_value);

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL db2xml.dxxRetrieve(:collection:collection_ind,
       :result_tab:rtab_ind,
       :overrideType:overrideType_ind,:overwrite:overwrite_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
       :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Die Bedingung ″price > 2500.00″ in `order.dad` wird durch ″price > ?″ überschrieben, wobei ″?″ von der Eingabevariablen *price_value* abhängt.

- **Zerlegte XML-Daten suchen:**

  Sie können zum Durchsuchen von Gruppentabellen die üblichen SQL-Abfrageoperationen verwenden. Sie können Gruppentabellen verknüpfen, Unterabfragen verwenden und anschließend eine strukturelle Textsuche in den Textspalten durchführen. Mit dem Ergebnis der strukturellen Suche können Sie die abzurufenden Daten anwenden oder das angegebene XML-Dokument generieren.

Teil 4. Referenz

Dieser Teil des Handbuchs enthält Syntaxinformationen für den XML Extender-Verwaltungsbefehl, die benutzerdefinierten Datentypen (UDTs), die benutzerdefinierten Funktionen (UDFs) und die gespeicherten Prozeduren. Außerdem finden Sie hier Nachrichtentexte zur Fehlerbestimmung.

Kapitel 7. XML Extender-Verwaltungsbefehl: dxxadm

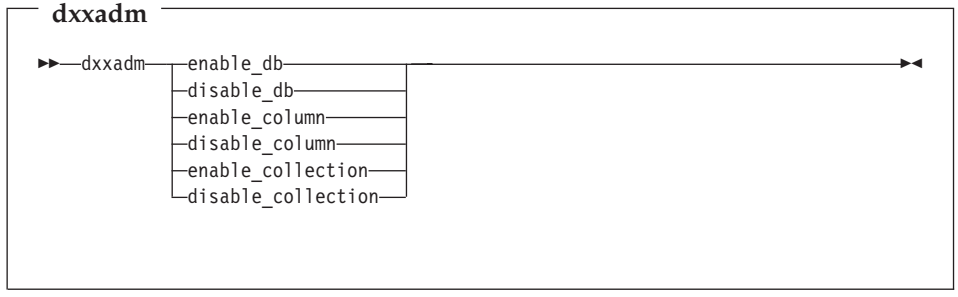
Der XML Extender bietet einen Verwaltungsbefehl **dxxadm** zur Ausführung der folgenden Verwaltungs-Tasks:Sie führen die folgenden XML Extender Verwaltungs-Tasks über den Aufruf (CALL) von **dxxadm** mit verschiedenen Befehlsoptionen aus:

- Aktivieren oder Inaktivieren einer Datenbank für den XML Extender
- Aktivieren oder Inaktivieren einer XML-Spalte
- Aktivieren oder Inaktivieren einer XML-Objektgruppe

Sie können zur Ausführung der einzelnen Verwaltungs-Tasks auch den XML Extender-Verwaltungsassistenten oder gespeicherte Prozeduren verwenden.

Höhere Syntax

Das folgende Syntaxdiagramm zeigt die höhere Syntax des Befehls **dxxadm**. Beschreibungen der einzelnen Optionen finden Sie in den folgenden Abschnitten.



Verwaltungsbefehlsoptionen

Die folgenden Abschnitte beschreiben die verschiedenen Befehlsoptionen des Befehls **dxxadm**, die dem Systemprogrammierer zur Verfügung stehen:

- „enable_db" auf Seite 170
- „disable_db" auf Seite 172
- „enable_column" auf Seite 174
- „disable_column" auf Seite 176
- „enable_collection" auf Seite 178
- „disable_collection" auf Seite 180

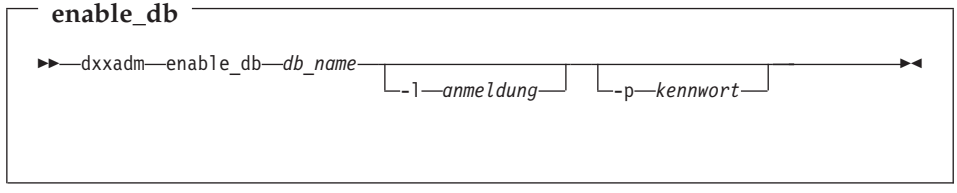
enable_db

Zweck

Stellt eine Verbindung zur Datenbank her und aktiviert sie, so daß sie mit dem XML Extender verwendet werden kann. Wenn die Datenbank aktiviert ist, erstellt der XML Extender die folgenden Objekte:

- Die benutzerdefinierten Typen (UDTs) des XML Extender.
- Die benutzerdefinierten Funktionen (UDFs) des XML Extender.
- Die XML Extender-DTD-Referenztabelle, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden. Eine vollständige Beschreibung der Tabelle DTD_REF finden Sie im Abschnitt „DTD-Referenztabelle" auf Seite 245.
- Die XML Extender-Verwendungstabelle, XML_USAGE, in der allgemeine Informationen für alle für XML aktivierten Spalten und Objektgruppen gespeichert werden. Eine vollständige Beschreibung der Tabelle XML_USAGE finden Sie im Abschnitt „XML-Verwendungstabelle" auf Seite 246.

Format



Parameter

Tabelle 14. Parameter für enable_db

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der sich die XML-Daten befinden.
-l *anmeldung*	Die wahlfreie Benutzer-ID; wenn sie angegeben ist, wird über diese ID die Verbindung zur Datenbank hergestellt. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das wahlfreie Kennwort; wenn es angegeben ist, wird es zum Herstellen der Verbindung zu der Datenbank verwendet. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiele

Mit dem folgenden Beispiel wird die Datenbank SALES_DB aktiviert.

```
dxxadm enable_db SALES_DB
```

disable_db

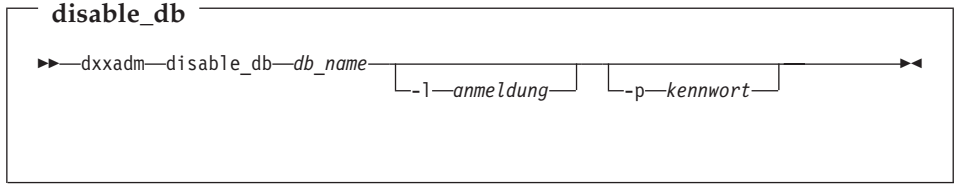
Zweck

Stellt die Verbindung zu der für XML aktivierten Datenbank her und inaktiviert sie. Wenn die Datenbank inaktiviert ist, kann sie vom XML Extender nicht mehr verwendet werden. Wenn der XML Extender die Datenbank inaktiviert, werden die folgenden Objekte freigegeben:

- Die benutzerdefinierten Typen (UDTs) des XML Extender.
- Die benutzerdefinierten Funktionen (UDFs) des XML Extender.
- Die XML Extender-DTD-Referenztabelle, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden. Eine vollständige Beschreibung der Tabelle DTD_REF finden Sie im Abschnitt „DTD-Referenztabelle" auf Seite 245.
- Die XML Extender-Verwendungstabelle, XML_USAGE, in der allgemeine Informationen für alle für XML aktivierten Spalten und Objektgruppen gespeichert werden. Eine vollständige Beschreibung der Tabelle XML_USAGE finden Sie im Abschnitt „XML-Verwendungstabelle" auf Seite 246.

Wichtig: Sie müssen alle XML-Spalten inaktivieren, bevor Sie versuchen, eine Datenbank zu inaktivieren. Der XML Extender kann eine Datenbank nicht inaktivieren, wenn diese Spalten oder Objektgruppen enthält, die für XML aktiviert wurden.

Format



Parameter

Tabelle 15. Parameter für disable_db

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der sich die XML-Daten befinden.
-l *anmeldung*	Die wahlfreie Benutzer-ID; wenn sie angegeben ist, wird über diese ID die Verbindung zur Datenbank hergestellt. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das wahlfreie Kennwort; wenn es angegeben ist, wird es zum Herstellen der Verbindung zu der Datenbank verwendet. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiele

Mit dem folgenden Beispiel wird die Datenbank SALES_DB inaktiviert.

```
dxxadm disable_db SALES_DB
```

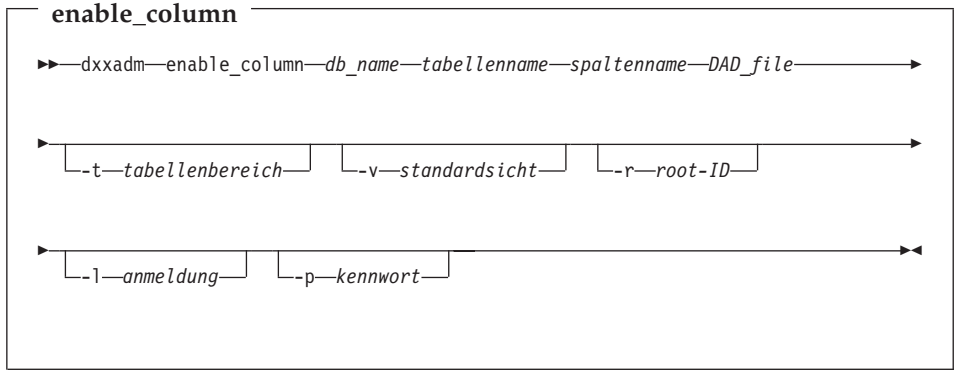
enable_column

Zweck

Stellt eine Verbindung zu einer Datenbank her und aktiviert eine XML-Spalte, so daß sie die UDTs des XML Extender enthalten kann. Beim Aktivieren einer Spalte führt der XML Extender die folgenden Tasks aus:

- Festlegen, ob die XML-Tabelle einen Primärschlüssel hat; falls nicht, ändert der XML Extender die XML-Tabelle und fügt die Spalte DXXROOT_ID hinzu.
- Erstellen der in der DAD-Datei angegebenen Seitentabellen mit einer Spalte für die eindeutige Kennung jeder Zeile in der XML-Tabelle. Diese Spalte ist entweder die vom Benutzer angegebene *root_id* oder die DXXROOT_ID, die vom XML Extender benannt wurde.
- Erstellen einer Standardsicht für die XML-Tabelle und ihre Seitentabellen; wahlweise wird dabei ein von Ihnen angegebener Name verwendet.

Format



Parameter

Tabelle 16. Parameter für enable_column

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der sich die XML-Daten befinden.
tabellenname	Der Name der Tabelle, in der sich die XML-Spalte befindet.
spaltenname	Der Name der XML-Spalte.
DAD-Datei	Der Name der DAD-Datei, die das XML-Dokument der XML-Spalte und den Seitentabellen zuordnet.
-t *tabellenbereich*	Der wahlfreie Tabellenbereich; enthält die der XML-Spalte zugeordneten Seitentabellen. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.
-v *standardsicht*	Der wahlfreie Name der Standardsicht, die die XML-Spalte und die Seitentabellen verknüpft.
-r *Root-ID*	Der Name des Primärschlüssels in der XML-Spaltentabelle, der als Root-ID für Seitentabellen verwendet werden soll. Die Root-ID ist wahlfrei.
-l *anmeldung*	Die wahlfreie Benutzer-ID; wenn sie angegeben ist, wird über diese ID die Verbindung zur Datenbank hergestellt. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das wahlfreie Kennwort; wenn es angegeben ist, wird es zum Herstellen der Verbindung zu der Datenbank verwendet. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiele

Mit dem folgenden Beispiel wird eine XML-Spalte aktiviert.

```
dxxadm enable_column SALES_DB SALES_TAB ORDER -v sales_order_view -r
INVOICE_NUMBER
```

disable_column

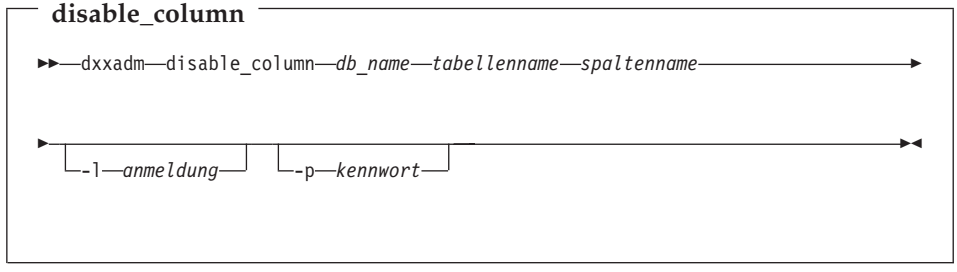
Zweck

Stellt die Verbindung zu einer Datenbank her und inaktiviert die für XML aktivierte Spalte. Wenn die Spalte inaktiviert ist, kann sie keine XML-Datentypen mehr enthalten. Beim Inaktivieren einer für XML aktivierten Spalte werden die folgenden Aktionen ausgeführt:

- Der Eintrag "XML-Spaltenverwendung" wird aus der Tabelle XML_USAGE entfernt.
- Der USAGE_COUNT wird in der Tabelle DTD_REF reduziert.
- Alle dieser Spalte zugeordneten Auslöser werden freigegeben.
- Alle dieser Spalte zugeordneten Seitentabellen werden gelöscht.

Wichtig: Sie müssen eine XML-Spalte inaktivieren, bevor Sie eine XML-Tabelle freigeben. Wenn eine XML-Tabelle freigegeben wird, ohne daß ihre XML-Spalte inaktiviert wird, behält der XML Extender beide erstellten Seitentabellen sowie den XML-Spalteneintrag in der Tabelle XML_USAGE bei.

Format



Parameter

Tabelle 17. Parameter für disable_column

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der sich die Daten befinden.
tabellenname	Der Name der Tabelle, in der sich die XML-Spalte befindet.
spaltenname	Der Name der XML-Spalte.
-l *anmeldung*	Die wahlfreie Benutzer-ID; wenn sie angegeben ist, wird über diese ID die Verbindung zur Datenbank hergestellt. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das wahlfreie Kennwort; wenn es angegeben ist, wird es zum Herstellen der Verbindung zu der Datenbank verwendet. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiele

Mit dem folgenden Beispiel wird eine für XML aktivierte Spalte inaktiviert.

```
dxxadm disable_column SALES_DB SALES_TAB ORDER
```

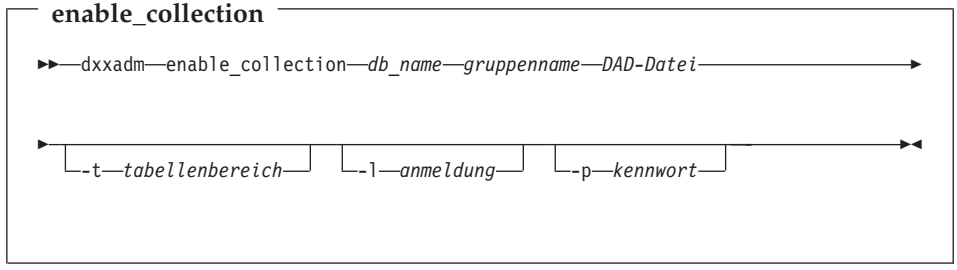
enable_collection

Zweck

Stellt die Verbindung zu einer Datenbank her und aktiviert eine XML-Objektgruppe entsprechend der angegebenen DAD. Beim Aktivieren einer Objektgruppe führt der XML Extender die folgenden Tasks aus:

- Erstellt einen Eintrag zur XML-Objektgruppenverwendung in der Tabelle XML_USAGE.
- Bei der RDB_node-Zuordnung werden die in der DAD-Datei angegebenen Objektgruppentabellen erstellt, wenn diese Tabellen nicht in der Datenbank vorhanden sind.

Format



Parameter

Tabelle 18. Parameter für enable_collection

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der sich die Daten befinden.
gruppenname	Der Name der XML-Objektgruppe.
DAD-Datei	Der Name der DAD-Datei, die das XML-Dokument den relationalen Tabellen in der Objektgruppe zuordnet.
-t *tabellenbereich*	Der wahlfreie Tabellenbereich, der der Objektgruppe zugeordnet ist. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.
-l *anmeldung*	Die wahlfreie Benutzer-ID; wenn sie angegeben ist, wird über diese ID die Verbindung zur Datenbank hergestellt. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.

Tabelle 18. Parameter für enable_collection (Forts.)

Parameter	Beschreibung
-p *kennwort*	Das wahlfreie Kennwort; wenn es angegeben ist, wird es zum Herstellen der Verbindung zu der Datenbank verwendet. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiele

Mit dem folgenden Beispiel wird eine XML-Objektgruppe aktiviert.

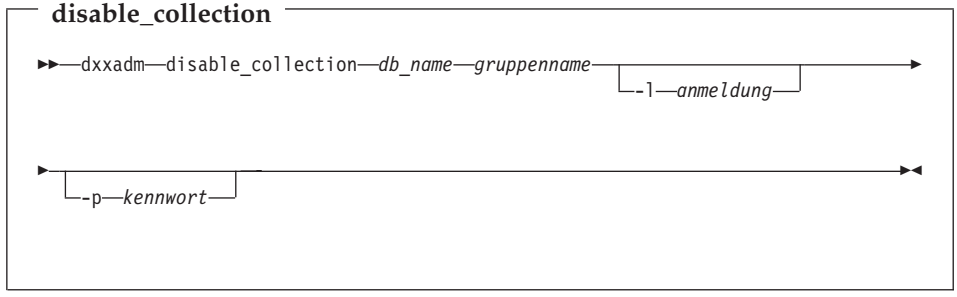
```
dxxadm enable_collection SALES_DB sales_ord getstart_xcollection.dad
-t orderspace
```

disable_collection

Zweck

Stellt die Verbindung zu einer Datenbank her und inaktiviert eine für XML aktivierte Objektgruppe. Der Objektgruppenname kann in den gespeicherten Prozeduren dxxRetrieveXML (Zusammenstellen) und dxxInsertXML (Zerlegen) nicht mehr verwendet werden. Wenn eine XML-Objektgruppe inaktiviert ist, wird der entsprechende Gruppeneintrag aus der Tabelle XML_USAGE gelöscht. Beachten Sie, daß durch das Inaktivieren der Objektgruppe die im Schritt enable_collection erstellten Gruppentabellen nicht gelöscht werden.

Format



Parameter

Tabelle 19. Parameter für disable_collection

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der sich die Daten befinden.
gruppenname	Der Name der XML-Objektgruppe.
-l *anmeldung*	Die wahlfreie Benutzer-ID; wenn sie angegeben ist, wird über diese ID die Verbindung zur Datenbank hergestellt. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das wahlfreie Kennwort; wenn es angegeben ist, wird es zum Herstellen der Verbindung zu der Datenbank verwendet. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiele

Mit dem folgenden Beispiel wird eine XML-Objektgruppe inaktiviert.

```
dxxadm disable_collection SALES_DB sales_ord
```

Kapitel 8. Benutzerdefinierte Typen des XML Extender

Die benutzerdefinierten Typen ("User-Defined Types", UDTs) des XML Extender sind Datentypen, die für XML-Spalten und XML-Objektgruppen verwendet werden. Alle UDTs haben den Schemanamen db2xml. Der XML Extender erstellt UDTs zum Speichern und Abrufen von XML-Dokumenten. Tabelle 20 enthält eine Übersicht über die UDTs.

Tabelle 20. Die XML Extender-UDTs

Spalte benutzerdefinierter Typ	Quellendatentyp	Beschreibung der Verwendung
XMLVARCHAR	VARCHAR(*varchar_len*)	Speichert ein vollständiges XML-Dokument als VARCHAR in DB2.
XMLCLOB	CLOB(*clob_len*)	Speichert ein vollständiges XML-Dokument als großes Zeichenobjekt ("Character Large Object, CLOB) in DB2.
XMLFILE	VARCHAR(512)	Gibt den Dateinamen des lokalen Datei-Servers an. Wenn XMLFILE für die XML-Spalte angegeben ist, speichert der XML Extender das XML-Dokument in einer externen Server-Datei. Der Text Extender kann nicht mit XMLFILE aktiviert werden. Sie müssen selbst die Integrität zwischen dem Dateiinhalt und DB2 sowie den für die Indexierung erstellten Seitentabellen sicherstellen.

Hierbei hängen *varchar_len* und *clob_len* von dem verwendeten Betriebssystem ab.

Für DB2 UDB ist *varchar_len* = 3 KB und *clob_len* = 2 GB.

Diese UDTs werden nur zur Angabe der Typen von Anwendungsspalten verwendet; sie gelten nicht für die vom XML Extender erstellten Seitentabellen.

Kapitel 9. Benutzerdefinierte Funktionen des XML Extender

Der XML Extender bietet Funktionen zum Speichern, Abrufen, Durchsuchen und Aktualisieren von XML-Dokumenten sowie zum Extrahieren von XML-Elementen oder Attributen. Verwenden Sie die benutzerdefinierten XML-Funktionen (UDFs) für XML-Spalten, nicht jedoch für XML-Objektgruppen. Alle UDFs haben den Schemanamen db2xml, der vor UDFs weggelassen werden kann.

Die vier Arten von XML Extender-Funktionen sind: Speicherfunktionen, Abruffunktionen, Extraktionsfunktionen und eine Aktualisierungsfunktion.

Speicherfunktionen
Speicherfunktionen fügen XML-Dokumente in eine DB2-Datenbank ein. Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „Speicherfunktionen" auf Seite 185.

Abruffunktionen
Abruffunktionen rufen XML-Dokumente aus XML-Spalten in einer DB2-Datenbank ab. Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „Abruffunktionen" auf Seite 190.

Extraktionsfunktionen
Extraktionsfunktionen extrahieren den Elementinhalt und den Attributwert aus einem XML-Dokument und setzen ihn in den Datentyp um, der im Funktionsnamen angegeben ist. Der XML Extender bietet eine Gruppe von Extraktionsfunktionen für verschiedene SQL-Datentypen. Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „Extraktionsfunktionen" auf Seite 197.

Aktualisierungsfunktion
Die Aktualisierungsfunktion Update() ändert den Elementinhalt bzw. Attributwert und gibt eine Kopie eines XML-Dokuments mit einem aktualisierten Wert zurück, das mit dem Standortpfad angegeben ist. Über die Funktion Update() kann der Anwendungsprogrammierer das zu aktualisierende Element bzw. Attribut angeben. Eine ausführliche Beschreibung der Syntax sowie Beispiele finden Sie im Abschnitt „Aktualisierungsfunktion" auf Seite 214.

Tabelle 21 auf Seite 184 zeigt eine Zusammenfassung der XML Extender-Funktionen.

Tabelle 21. Die benutzerdefinierten Funktionen des XML Extender

Typ	Funktion
Speicherfunktionen	XMLVarcharFromFile()
	XMLCLOBFromFile()
	XMLFileFromVarchar()
	XMLFileFromCLOB()
Abruffunktionen	Content(): Abrufen von XMLFile in ein CLOB
	Content(): Abrufen von XMLVarchar in eine externe Server-Datei
	Content(): Abrufen von XMLCLOB in eine externe Server-Datei
Extraktionsfunktionen	extractInteger() und extractIntegers()
	extractSmallint() and extractSmallints()
	extractDouble() und extractDoubles()
	extractReal() und extractReals()
	extractChar() und extractChars()
	extractVarchar() und extractVarchars()
	extractCLOB() und extractCLOBs()
	extractDate() und extractDates()
	extractTime() und extractTimes()
	extractTimestamp() und extractTimestamps()
Aktualisierungsfunktion	Update()

Beim Verwenden von Parametermarken in UDFs erfordert eine JDBC-Einschränkung, daß die Parametermarke für die UDF in den Datentyp der Spalte umgesetzt wird, in die die zurückgegebenen Daten eingefügt werden sollen. Im Abschnitt „Einschränkungen beim Aufruf von Funktionen von JDBC aus" auf Seite 145 finden Sie Informationen zum Umsetzen der Parametermarken.

Speicherfunktionen

Verwenden Sie die Speicherfunktionen zum Einfügen von XML-Dokumenten in eine DB2-Datenbank. Sie können die Standardumsetzungsfunktionen eines UDT direkt in den Anweisungen INSERT oder SELECT verwenden, wie im Abschnitt „Daten speichern" auf Seite 128 beschrieben. Darüber hinaus bietet der XML Extender UDFs, um auf XML-Dokumente aus anderen Datenquellen als dem UDT-Basisdatentyp zuzugreifen und ihn in den gewünschten UDT umzusetzen.

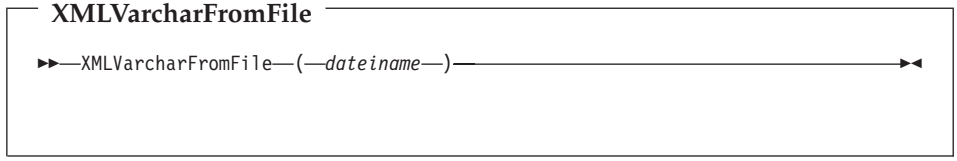
Der XML Extender umfaßt die folgenden Speicherfunktionen:

- „XMLVarcharFromFile()" auf Seite 186
- „XMLCLOBFromFile()" auf Seite 187
- „XMLFileFromVarchar()" auf Seite 188
- „XMLFileFromCLOB()" auf Seite 189

XMLVarcharFromFile()

Zweck
Liest ein XML-Dokument aus einer Server-Datei und gibt das Dokument als Typ XMLVARCHAR zurück.

Syntax



Parameter

Tabelle 22. Parameter für XMLVarcharFromFile

Parameter	Datentyp	Beschreibung
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Server-Datei.

Rückgabetyp
XMLVARCHAR

Beispiel
Mit dem folgenden Beispiel wird ein XML-Dokument aus einer Server-Datei gelesen und als XMLVARCHAR-Typ in eine XML-Spalte eingefügt.

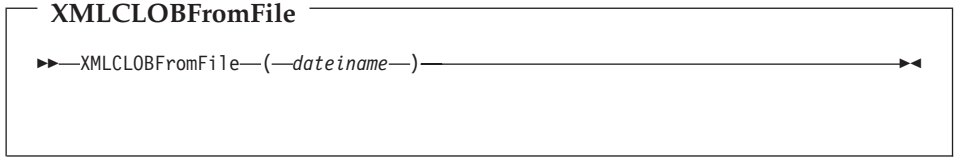
```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
   XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

In diesem Beispiel wird ein Datensatz in die Tabelle SALES_TAB eingefügt. Die Funktion XMLVarcharFromFile() importiert das XML-Dokument aus einer Datei in DB2 und speichert es als XMLVARCHAR.

XMLCLOBFromFile()

Zweck
Liest ein XML-Dokument aus einer Server-Datei und gibt das Dokument als Typ XMLCLOB zurück.

Syntax



Parameter

Tabelle 23. Parameter für XMLCLOBFromFile

Parameter	Datentyp	Beschreibung
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Server-Datei.

Rückgabetyp
XMLCLOB as LOCATOR

Beispiel
Mit dem folgenden Beispiel wird ein XML-Dokument aus einer Server-Datei gelesen und als XMLCLOB-Typ in eine XML-Spalte eingefügt.

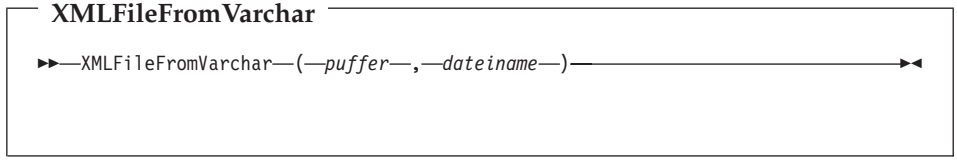
```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

Die Spalte ORDER in der Tabelle SALES_TAB ist als Typ XMLCLOB definiert. Das vorige Beispiel zeigt, wie die Spalte ORDER in die Tabelle SALES_TAB eingefügt wird.

XMLFileFromVarchar()

Zweck
Liest ein XML-Dokument aus dem Speicher als VARCHAR ein, schreibt es in eine externe Server-Datei und gibt den Dateinamen und -pfad als Typ XML-FILE zurück.

Syntax



Parameter

Tabelle 24. Parameter für XMLFileFromVarchar

Parameter	Datentyp	Beschreibung
puffer	VARCHAR(3K)	Der Hauptspeicherpuffer.
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Server-Datei.

Rückgabetyp
XMLFILE

Beispiel
Liest ein XML-Dokument aus dem Speicher als VARCHAR ein, schreibt es in eine externe Server-Datei und fügt den Dateinamen und -pfad als Typ XML-FILE in eine XML-Spalte ein.

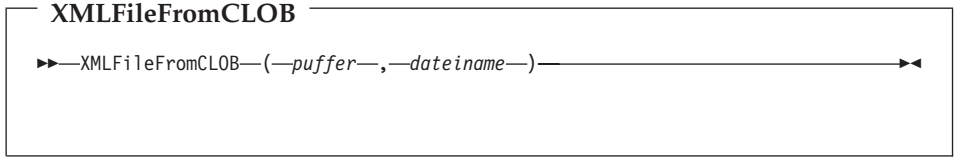
```
EXEC SQL BEGIN DECLARE SECTION;
      struct { short len; char data[3000]; } xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
  VALUES('1234', 'Sriram Srinivasan',
       XMLFileFromVarchar(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

Die Spalte ORDER in der Tabelle SALES_TAB ist als Typ XMLFILE definiert. Das vorige Beispiel zeigt, daß Sie ein in Ihrem Puffer befindliches Dokument in einer Server-Datei speichern können.

XMLFileFromCLOB()

Zweck
Liest ein XML-Dokument als CLOB-Zeigereinheit ein, schreibt es in eine externe Server-Datei und gibt den Dateinamen und -pfad als Typ XMLFILE zurück.

Syntax



Parameter

Tabelle 25. Parameter für XMLFileFromCLOB()

Parameter	Datentyp	Beschreibung
puffer	CLOB as LOCATOR	Der Puffer, der das XML-Dokument enthält.
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Server-Datei.

Rückgabetyp
XMLFILE

Beispiel
Liest ein XML-Dokument als CLOB-Zeigereinheit ein, schreibt es in eine externe Server-Datei und fügt den Dateinamen und -pfad als Typ XMLFILE in eine XML-Spalte ein.

```
EXEC SQL BEGIN DECLARE SECTION;
SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
   XMLFileFromCLOB(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

Die Spalte ORDER in der Tabelle SALES_TAB ist als Typ XMLFILE definiert. Wenn sich ein XML-Dokument in Ihrem Puffer befindet, können Sie es in einer Server-Datei speichern.

Abruffunktionen

Sie können die Standardumsetzungsfunktionen verwenden, um einen XML-UDT in den Basisdatentyp umzusetzen, wie im Abschnitt „Ein vollständiges Dokument abrufen" auf Seite 131 beschrieben. Der XML Extender bietet außerdem eine *überladene Funktion* Content(), die zum Abrufen verwendet wird.

Die Funktion Content() ermöglicht folgende Abrufarten:

- **Abrufen vom Zusatzspeicher auf dem Server in eine Host-Variable auf dem Client.**

  Sie können mit Content() ein XML-Dokument in einen Speicherpuffer abrufen, wenn es als externe Server-Datei gespeichert ist. Sie können „Content(): Abrufen von XMLFILE in ein CLOB" auf Seite 191 zu diesem Zweck verwenden.

- **Abrufen aus dem internen Speicher in eine externe Server-Datei**

  Sie können mit Content() auch ein in DB2 gespeichertes XML-Dokument abrufen und es in einer Server-Datei auf dem Dateisystem des DB2-Servers speichern. Die folgenden Content()-Funktionen werden zum Speichern von Informationen in externen Server-Dateien verwendet:

  – „Content(): Abrufen von XMLVARCHAR in eine externe Server-Datei" auf Seite 193

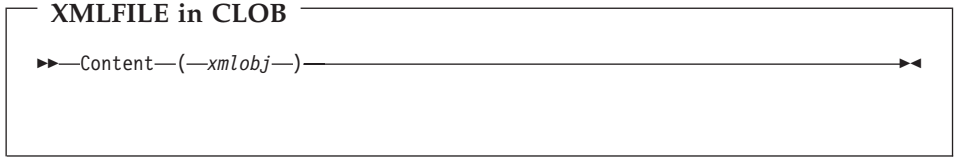
  – „Content(): Abrufen von XMLCLOB in eine externe Server-Datei" auf Seite 195

Bei den Beispielen im folgenden Abschnitt wird davon ausgegangen, daß Sie mit der DB2-Befehls-Shell arbeiten, in der Sie nicht „DB2" am Anfang jedes Befehls eingeben müssen.

Content(): Abrufen von XMLFILE in ein CLOB

Zweck
Ruft Daten aus einer Server-Datei ab und speichert sie in einer CLOB LOCATOR.

Syntax



Parameter

Tabelle 26. Parameter für XMLFILE in CLOB

Parameter	Datentyp	Beschreibung
xmlobj	XMLFILE	Das XML-Dokument.

Rückgabetyp
CLOB (*clob_len*) als LOCATOR

clob_len für DB2 UDB ist 2 GB.

Beispiel
Mit dem folgenden Beispiel werden Daten aus einer Server-Datei abgerufen und in einer CLOB-Zeigereinheit gespeichert.

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT TO SALES_DB

EXEC SQL DECLARE c1 CURSOR FOR

     SELECT Content(order) from sales_tab
     WHERE sales_person = 'Sriram Srinivasan'

EXEC SQL OPEN c1;

do {
   EXEC SQL FETCH c1 INTO :xml_buff;
   if (SQLCODE != 0) {
      break;
      }
   else {
     /* Aktion mit XML-Dokument im Puffer */
```

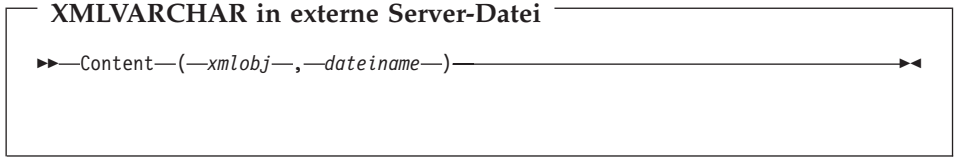
```
    }
  }

EXEC SQL CLOSE c1;

EXEC SQL CONNECT RESET;
```

Die Spalte ORDER in der Tabelle SALES_TAB hat den Typ XMLFILE, so daß die UDF Content() Daten aus einer Server-Datei abruft und in einer CLOB-Zeigereinheit speichert.

Content(): Abrufen von XMLVARCHAR in eine externe Server-Datei

Zweck
Ruft den als Typ XMLVARCHAR gespeicherten XML-Inhalt ab und speichert ihn in einer externen Server-Datei.

Syntax



Wichtig: Wenn bereits eine Datei mit dem angegebenen Namen vorhanden ist, überschreibt die Content-Funktion ihren Inhalt.

Anmerkung:

Parameter

Tabelle 27. Parameter für XMLVarchar in externe Server-Datei

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR	Das XML-Dokument.
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Server-Datei.

Rückgabetyp
VARCHAR(512)

Beispiel
Im folgenden Beispiel wird der als Typ XMLVARCHAR gespeicherte XML-Inhalt abgerufen und in einer externen Server-Datei gespeichert.

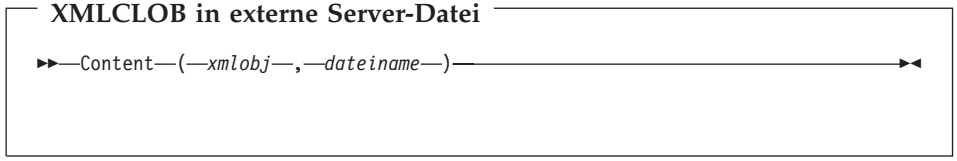
```
CREATE table app1 (id int NOT NULL, order db2xml.XMLVarchar);
INSERT into app1 values (1, '<?xml version="1.0"?>
   <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
     <Order key="1">
        <Customer>
          <Name>American Motors</Name>
          <Email>parts@am.com</Email>
        </Customer>
        <Part color="black">
          <key>68</key>
          <Quantity>36</Quantity>
          <ExtendedPrice>34850.16</ExtendedPrice>
          <Tax>6.000000e-02</Tax>
```

```
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>AIR </ShipMode>
            </Shipment>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>BOAT </ShipMode>
            </Shipment>
        </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart_.dad')
  from app1 where ID=1;
```

Content(): Abrufen von XMLCLOB in eine externe Server-Datei

Zweck
Ruft den als Typ XMLCLOB gespeicherten XML-Inhalt ab und speichert ihn in einer externen Server-Datei.

Syntax



Wichtig: Wenn bereits eine Datei mit dem angegebenen Namen vorhanden ist, überschreibt die Content-Funktion ihren Inhalt.

Parameter

Tabelle 28. Parameter für XMLCLOB in externe Server-Datei

Parameter	Datentyp	Beschreibung
xmlobj	XMLCLOB as LOCATOR	Das XML-Dokument.
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Server-Datei.

Rückgabetyp
VARCHAR(512)

Beispiel
Im folgenden Beispiel wird der als Typ XMLCLO gespeicherte XML-Inhalt abgerufen und in einer externen Server-Datei gespeichert.

```
CREATE table app1 (id int NOT NULL, order db2xml.XMLCLOB not logged);

INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
      <Customer>
          <Name>American Motors</Name>
          <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
          <key>68</key>
          <Quantity>36</Quantity>
          <ExtendedPrice>34850.16</ExtendedPrice>
          <Tax>6.000000e-02</Tax>
          <Shipment>
              <ShipDate>1998-08-19</ShipDate>
```

```
            <ShipMode>AIR </ShipMode>
        </Shipment>
        <Shipment>
            <ShipDate>1998-08-19</ShipDate>
            <ShipMode>BOAT </ShipMode>
        </Shipment>
    </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart.xml')
from app1 where ID=1;
```

Extraktionsfunktionen

Die Extraktionsfunktionen extrahieren den Elementinhalt oder den Attributwert aus einem XML-Dokument und geben die angeforderten SQL-Datentypen zurück. Der XML Extender bietet eine Gruppe von Extraktionsfunktionen für verschiedene SQL-Datentypen. Die Extraktionsfunktionen verwenden zwei Eingabeparameter. Der erste Parameter ist der XML Extender-UDT; dies kann einer der XML-UDTs sein. Der zweite Parameter ist der Standortpfad, der das XML-Element oder Attribut angibt. Jede Extraktionsfunktion gibt den Wert oder Inhalt zurück, der vom Standortpfad angegeben wird.

Da einige Element- oder Attributwerte mehrfach auftreten, geben die Extraktionsfunktionen entweder einen skalaren Wert oder einen Tabellenwert zurück; letzterer wird als Tabellenfunktion bezeichnet.

Der XML Extender umfaßt die folgenden Extraktionsfunktionen:

- „extractInteger() und extractIntegers()" auf Seite 198
- „extractSmallint() und extractSmallints()" auf Seite 200
- „extractDouble() und extractDoubles()" auf Seite 201
- „extractReal() und extractReals()" auf Seite 203
- „extractChar() und extractChars()" auf Seite 204
- „extractVarchar() und extractVarchars()" auf Seite 205
- „extractCLOB() und extractCLOBs()" auf Seite 207
- „extractDate() und extractDates()" auf Seite 209
- „extractTime() und extractTimes()" auf Seite 210
- „extractTimestamp() und extractTimestamps()" auf Seite 212

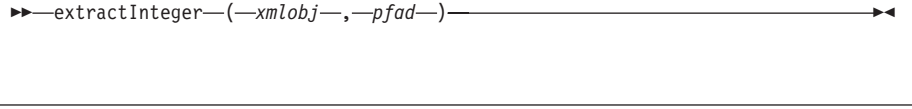
Bei den Beispielen im folgenden Abschnitt wird davon ausgegangen, daß Sie mit der DB2-Befehls-Shell arbeiten, in der Sie nicht „DB2" am Anfang jedes Befehls eingeben müssen.

extractInteger() und extractIntegers()

Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ INTEGER zurück.

Syntax

Skalarfunktion



Tabellenfunktion

Parameter

Tabelle 29. Funktionsparameter für extractInteger und extractIntegers

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
INTEGER

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedInteger

Beispiel

Skalarfunktion, Beispiel:

Im folgenden Beispiel wird ein Wert zurückgegeben, wenn der Attributwert für key = ″1″ ist. Der Wert wird als INTEGER extrahiert und automatisch in einen Typ DECIMAL konvertiert.

```
CREATE TABLE t1(key decimal(3,2));
INSERT into t1 values
SELECT * from table(db2xml.extractInteger(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/[@key="1"]'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird jeder Schlüsselwert für die Bestellung als INTEGER extrahiert

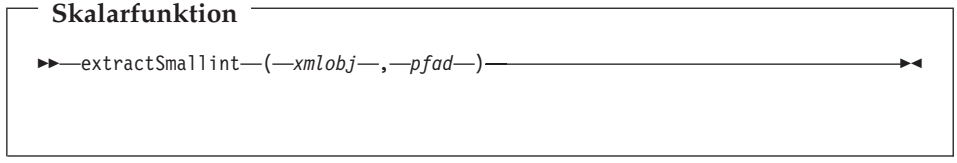
```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

extractSmallint() und extractSmallints()

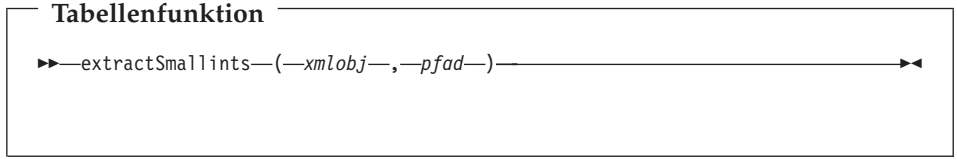
Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ SMALLINT zurück.

Syntax

Skalarfunktion



Tabellenfunktion



Parameter

Tabelle 30. Funktionsparameter für extractSmallint und extractSmallints

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
SMALLINT

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedSmallint

Beispiel
Im folgenden Beispiel wird der Wert von Quantity in allen Bestellungen als SMALLINT extrahiert

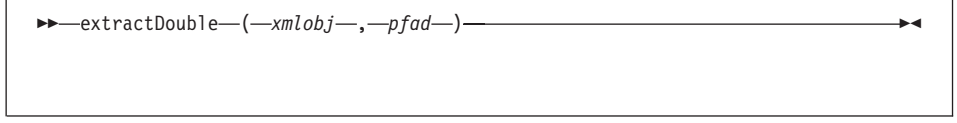
```
SELECT * from table(db2xml.extractSmallints(db2xml.File
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Quantity')) as x;
```

extractDouble() und extractDoubles()

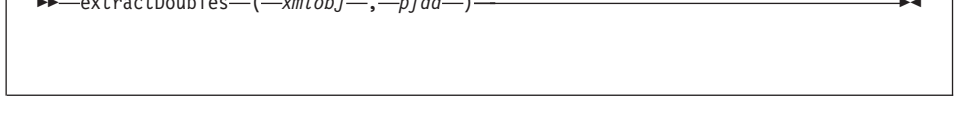
Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ DOUBLE zurück.

Syntax

Skalarfunktion



Tabellenfunktion



Parameter

Tabelle 31. Funktionsparameter für extractDouble und extractDoubles

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
DOUBLE

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedDouble

Beispiel

Skalarfunktion, Beispiel:

Mit dem folgenden Beispiel wird der Preis in einer Bestellung automatisch von einem Typ DOUBLE in den Typ DECIMAL umgesetzt.

```
CREATE TABLE t1(price, DECIMAL(5,2));
INSERT into t1 values (db2xml.extractDouble(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'),
    '/Order/Part[@color="black "]/ExtendedPrice'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von ExtendedPrice in jedem Teil der Bestellung als DOUBLE extrahiert.

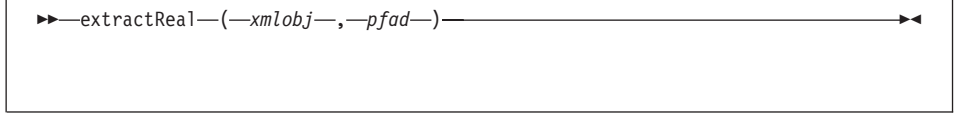
```
SELECT * from table(db2xml.extractDoubles(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/ExtendedPrice')) as x;
```

extractReal() und extractReals()

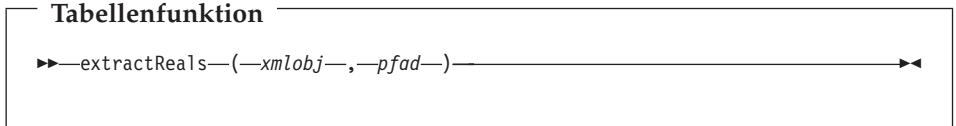
Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ REAL zurück.

Syntax

Skalarfunktion



Tabellenfunktion



Parameter

Tabelle 32. Funktionsparameter für extractReal und extractReals

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
REAL

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedReal

Beispiel
Im folgenden Beispiel wird jeder Wert von Tax als REAL extrahiert.

```
SELECT * from table(db2xml.extractReals(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Tax')) as x;
```

extractChar() und extractChars()

Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ CHAR zurück.

Syntax

Skalarfunktion

Tabellenfunktion

Parameter

Tabelle 33. Funktionsparameter für extractChar und extractChars

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
CHAR

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedChar

Beispiel
Im folgenden Beispiel wird der Wert von Color als CHAR extrahiert.

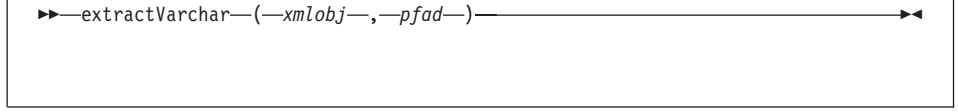
```
SELECT * from table(db2xml.extractChars(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/@Color')) as x;
```

extractVarchar() und extractVarchars()

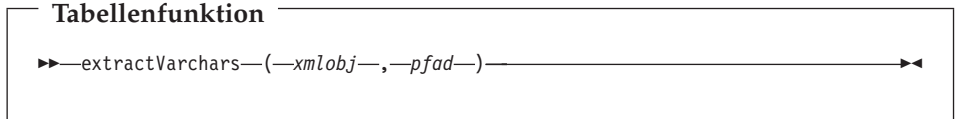
Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ VARCHAR zurück.

Syntax

Skalarfunktion



Tabellenfunktion



Parameter

Tabelle 34. Funktionsparameter für extractVarchar und extractVarchars

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
VARCHAR(4K)

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedVarchar

Beispiel
In einer Datenbank, in der mehr als 1000 XML-Dokumente in der Spalte ORDER der Tabelle SALES_TAB gespeichert sind, wollen Sie alle Kunden ermitteln, die Artikel bestellt haben, für die der ExtendedPrice größer als 2500.00 ist. Die folgende SQL-Anweisung verwendet die Extraktions-UDF in der SELECT-Klausel:

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

Die UDF extractVarchar() verwendet die Spalte ORDER als Eingabe und den Standortpfad `/Order/Customer/Name` als Auswahlkennung. Die UDF gibt die Namen der Kunden zurück. Mit der WHERE-Klausel wertet die Extraktionsfunktion nur die Bestellungen aus, die einen ExtendedPrice größer als 2500.00 aufweisen.

extractCLOB() und extractCLOBs()

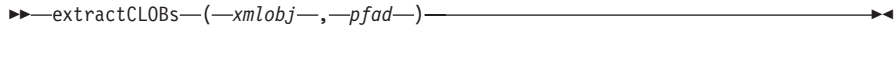
Zweck

Extrahiert einen Teil von XML-Dokumenten mit Element- und Attributformatierung sowie dem Inhalt von Elementen und Attributen einschließlich der Unterelemente. Diese Funktionen unterscheiden sich von den anderen Extraktionsfunktionen; sie geben nur den Inhalt der Elemente und Attribute zurück. Die Funktionen extractClob(s) sollten zum Extrahieren von Dokumentteilen verwendet werden, während extractVarchar(s) und extractChar(s) zum Extrahieren einfacher Werte verwendet werden sollten.

Syntax

Skalarfunktion

Tabellenfunktion



Parameter

Tabelle 35. Funktionsparameter für extractCLOB und extractCLOBs

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
CLOB(10K)

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedCLOB

Beispiel
In diesem Beispiel werden alle oder einige der Elemente aus einer Bestellung extrahiert.

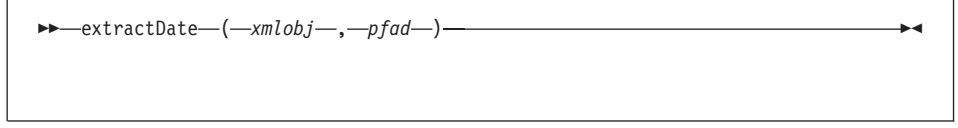
```
SELECT returnedCLOB as part
 from table(db2xml.extractCLOBs(db2xml.XMLFile('c:\dxx\samples\xml\getstart.xml'),
 '/Order/Part')) as x;
```

extractDate() und extractDates()

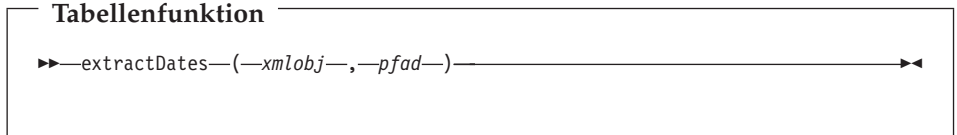
Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ DATE zurück.

Syntax

Skalarfunktion



Tabellenfunktion



Parameter

Tabelle 36. Funktionsparameter für extractDate und extractDates

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
DATE

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedDate

Beispiel
Im folgenden Beispiel wird der Wert von ShipDate als DATE extrahiert.

```
SELECT * from table(db2xml.extractDates(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Shipment/ShipDate')) as x;
```

extractTime() und extractTimes()

Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ TIME zurück.

Syntax

Skalarfunktion

Tabellenfunktion

Parameter

Tabelle 37. Funktionsparameter für extractTime und extractTimes

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
TIME

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedTime

Beispiel

Dieses Beispiel verwendet die Buch-Beispieldateien. Es sucht die XML-Datei book.xml für den Zeitpunkt, zu dem die Preise für Bücher festgesetzt wurden, und gibt die Werte als TIME zurück.

XML-Datei:

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
38.281
</price>
</book>
```

Beispiel für die Extraktion:

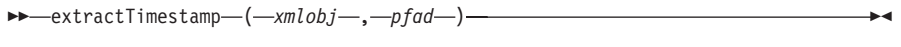
```
CREATE TABLE t1(SaleTime TIME);
INSERT INTO t1 values(db2xml.extractTime(doc, '/book/price/@time'));
SELECT * from t1;
```

extractTimestamp() und extractTimestamps()

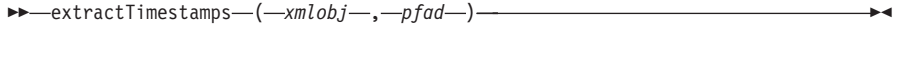
Zweck
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ TIMESTAMP zurück.

Syntax

Skalarfunktion



Tabellenfunktion



Parameter

Tabelle 38. Funktionsparameter für extractTimestamp und extractTimestamps

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Rückgabetyp
TIMESTAMP

Zurückgegebener Spaltenname (Tabellenfunktion)
returnedTimestamp

Beispiel

Dieses Beispiel verwendet die Buch-Beispieldateien. Es sucht die XML-Datei `book.xml` für den Zeitpunkt, zu dem die Preise für die Bücher festgesetzt wurden, und extrahiert den Wert als TIMESTAMP.

XML-Datei:

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
38.281
</price>
</book>
```

Beispiel für die Extraktion:

```
CREATE TABLE t1(SaleTime TIMESTAMP);
INSERT INTO t1 values(db2xml.extractTimestamp(doc, '/book/price/@timestamp'));
SELECT * from t1;
```

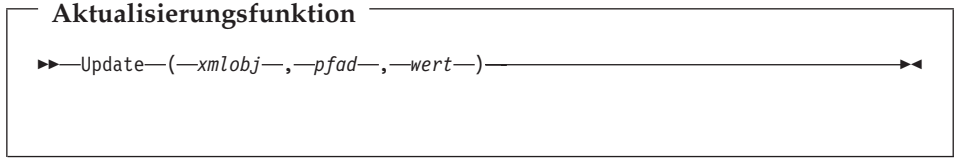
Aktualisierungsfunktion

Die Funktion Update() aktualisiert einen angegebenen Element- oder Attributwert in einem oder mehreren XML-Dokumenten in der XML-Spalte. Sie können auch die Standardumsetzungsfunktionen verwenden, um einen SQL-Basistyp in den XML-UDT umzusetzen, wie im Abschnitt „XML-Daten aktualisieren" auf Seite 136 beschrieben.

Zweck

Verwendet den Spaltennamen eines XML-UDT, einen Standortpfad und eine Zeichenfolge des Aktualisierungswerts und gibt einen XML-UDT zurück, der derselbe ist wie der erste Eingabeparameter. Mit der Funktion Update() können Sie das zu aktualisierende Element bzw. Attribut angeben.

Syntax



Parameter

Tabelle 39. Die Parameter für die UDF Update

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLCLOB as LOCATOR	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.
wert	VARCHAR	Die Aktualisierungs-Zeichenfolge.

Wichtig: Die Update-UDF unterstützt Standortpfade, die Prädikate mit Attribute haben, nicht jedoch Elemente. Das folgende Prädikat wird beispielsweise unterstützt:

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

Das folgende Prädikat wird nicht unterstützt:

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

Rückgabetyp

Datentyp	Rückgabetyp
XMLVARCHAR	XMLVARCHAR
XMLCLOB as LOCATOR	XMLCLOB

Beispiel

Das folgende Beispiel aktualisiert die Bestellung, die von dem Vertriebsmitarbeiter Sriram Srinivasan bearbeitet wird.

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
   WHERE sales_person = 'Sriram Srinivasan'
```

In diesem Beispiel wird der Inhalt von `/Order/Customer/Name` in `IBM` aktualisiert.

Verwendung

Wenn Sie mit der Aktualisierungsfunktion einen Wert in einem oder mehreren XML-Dokumenten ändern, werden tatsächlich die XML-Dokumente in der XML-Spalte ersetzt. Entsprechend der Ausgabe von dem XML-Parser werden einige Teile des Originaldokuments beibehalten, während andere Teile verlorengehen oder geändert werden. Die folgenden Abschnitte beschreiben, wie die Dokumente verarbeitet werden; außerdem finden Sie hier Beispiele, wie die Dokumente vor und nach der Aktualisierung aussehen.

Wie die Aktualisierungsfunktion das XML-Dokument verarbeitet

Wenn die Aktualisierungsfunktion XML-Dokumente ersetzt, muß sie das Dokument entsprechend der Ausgabe des XML-Parsers wiederherstellen. Tabelle 40 auf Seite 216 beschreibt mit Beispielen, wie die Teile des Dokuments verarbeitet werden. Beispiele zum Vergleich von XML-Dokumenten vor und nach einer Aktualisierung finden Sie im Abschnitt „Beispiele” auf Seite 219

Tabelle 40. Regeln der Aktualisierungsfunktion

Element oder Knotentyp	Codebeispiel zum XML-Dokument	Status nach der Aktualisierung
XML-Deklaration	`<?xml version='1.0' encoding='utf-8' standalone='yes' >`	Die XML-Deklaration wird beibehalten: • Versionsnummern werden beibehalten. • Codierungsdeklarationen werden beibehalten und werden angezeigt, wenn sie im Originaldokument angegeben sind. • Eigenständige Deklarationen werden beibehalten und angezeigt, wenn sie im Originaldokument angegeben sind. • Nach der Aktualisierung werden einfache Anführungszeichen zur Beschreibung der Werte verwendet.

Tabelle 40. Regeln der Aktualisierungsfunktion (Forts.)

Element oder Knotentyp	Codebeispiel zum XML-Dokument	Status nach der Aktualisierung

DOCTYPE-Deklaration	`<!DOCTYPE books SYSTEM "http://dtds.org/books.dtd" >`
	`<!DOCTYPE books PUBLIC "local.books.dtd" " http://dtds.org/books.dtd" >`
	`<!DOCTYPE books>`
	`-Alle`
	`<!DOCTYPE books ( S ExternalID ) ? [ internal-dtd-subset ] >`
	`-Zum Beispiel`
	`<!DOCTYPE books [ <!ENTITY mydog "Spot"> ] >? [ internal-dtd-subset ] >`
		Die Dokumentartdeklaration wird beibehalten:
		• Der Stammelementname wird unterstützt.
		• Allgemeine und System-`ExternalID`s werden beibehalten und angezeigt, wenn sie im Originaldokument angegeben sind.
		• Interne DTD-Untergruppen werden NICHT beibehalten. Entitäten werden ersetzt; Standardwerte für Attribute werden verarbeitet und erscheinen in den Ausgabe-dokumenten.
		• Nach der Aktualisierung werden doppelte Anführungszeichen zur Beschreibung von allgemeinen und System-URI-Werten verwendet.
		• Der aktuelle XML4c-Parser meldet keine XML-Deklaration, die keine `ExternalID` oder interne DTD-Untergruppe enthält. Nach einer Aktualisierung fehlt in diesem Fall die DOCTYPE-Deklaration.

Verarbeitungs-anweisungen	`<?xml-stylesheet title="compact" href="datatypes1.xsl" type="text/xsl"?>`	Verarbeitungsanweisungen werden beibehalten.

Tabelle 40. Regeln der Aktualisierungsfunktion (Forts.)

Element oder Knotentyp	Codebeispiel zum XML-Dokument	Status nach der Aktualisierung
Kommentare	`<!-- Kommentar -->`	Kommentar werden beibehalten, wenn sie innerhalb des Stammelements stehen. Kommentare außerhalb des Stammelements werden gelöscht.

Elemente	`<books>`
	`Inhalt`
	`</books>`
		Elemente werden beibehalten.

Attribute	`id='1' date='01/02/1997"`	Attribute von Elementen werden beibehalten. • Nach der Aktualisierung werden doppelte Anführungszeichen zur Beschreibung von Werten verwendet. • Daten in Attributen werden mit Escape-Zeichen maskiert. • Entitäten werden ersetzt.
Textknoten	`This chapter is about my dog &mydoc;.` `This chapter is about my dog Spot.`	Textknoten (Elementinhalt) werden beibehalten. • Daten innerhalb von Textknoten werden mit Escape-Zeichen maskiert. • Entitäten werden ersetzt.

Mehrfaches Vorkommen

Wenn in der UDF Update() ein Pfad angegeben ist, wird der Inhalt jedes Elements oder Attributs mit einem entsprechenden Pfad durch den angegebenen Wert ersetzt. Dies bedeutet: wenn in einem Dokument mehrere Pfade vorkommen, ersetzt die Aktualisierungsfunktion die vorhandenen Werte durch den Wert in dem Parameter *wert*.

Sie können ein Prädkat im Parameter *pfad* angeben, um eindeutige Standortpfade bereitzustellen und versehentliche Aktualisierungen zu verhindern. Die Aktualisierungs-UDF unterstützt Standortpfade, die Prädikate mit Attribute haben, nicht jedoch Elemente. Weitere Informationen hierzu finden Sie im Abschnitt „Parameter” auf Seite 214.

Beispiele

Die folgenden Beispiele zeigen Exemplare eines XML-Dokuments vor und nach einer Aktualisierung.

Beispiel 1:

Vorher:

```
<?xml version='1.0'   encoding='utf-8'    standalone="yes"?>
<!DOCTYPE book PUBLIC "public.dtd" "system.dtd">
<?pitarget option1='value1' option2='value2'?>
<!-- comment -->
<book>
  <chapter id="1" date='07/01/1997'>
    <!-- first section -->
<section>This is a section in Chapter One.</section>
</chapter>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
    <footnote>A footnote in Chapter Two is here.</footnote>
</chapter>
  <price date="12/22/1998" time="11.12.13"
       timestamp="1998-12-22-11.12.13.888888">38.281</price>
</book>
```

- Enthält Leerzeichen in der XML-Deklaration
- Gibt eine Verarbeitungsanweisung an
- Enthält einen Kommentar außerhalb des Stammknotens
- Gibt eine PUBLIC ExternalID an
- Enthält einen Kommentar innerhalb des Stammknotens

Nachher:

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<!DOCTYPE book PUBLIC "public.dtd" "system.dtd">
<?pitarget option1='value1' option2='value2'?><book>
  <chapter id="1" date="07/01/1997">
       <!-- first section -->
<section>This is a section in Chapter One.</section>
</chapter>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
       <footnote>A footnote in Chapter Two is here.</footnote>
</chapter>
  <price date="12/22/1998" time="11.12.13"
       timestamp="1998-12-22-11.12.13.888888">60.02</price>
</book>
```

- Leerzeichen innerhalb der Formatierung werden eliminiert
- Verarbeitungsanweisungen werden beibehalten
- Kommentar außerhalb des Stammknotens werden nicht beibehalten
- PUBLIC ExternalID wird beibehalten
- Kommentare innerhalb des Stammknotens werden beibehalten
- Der geänderte Wert ist der Wert des Elements <price>

Beispiel 2:

Vorher:

```
<?xml version='1.0'     ?>
<!DOCTYPE book>
<!-- comment -->
<book>
  ...
</book>
```

Enthält DOCTYPE-Deklaration ohne eine ExternalID oder eine interne DTD-Untergruppe. Nicht unterstützt.

Nachher:

```
<?xml version='1.0'?>
<book>
  ...
</book>
```

DOCTYPE-Deklaration wird vom XML-Parser nicht gemeldet und nicht beibehalten.

Beispiel 3:

Vorher:

```
<?xml version='1.0'     ?>
<!DOCTYPE book [ <!ENTITY myDog "Spot"> ]>
<!-- comment -->
<book>
  <chapter id="1" date='07/01/1997'>
       <!-- first section -->
       <section>This is a section in Chapter
         One about my dog &;myDog;.</section>
       ...
</chapter>
  ...
</book>
```

- Enthält Leerzeichen in der Formatierung
- Gibt die interne DTD-Untergruppe an
- Gibt die Entität im Textknoten an

Nachher:

```
<?xml version='1.0'?>
<!DOCTYPE book>
<book>
  <chapter id="1" date="07/01/1997">
       <!-- first section -->
       <section>This is a section in Chapter
         One about my dog Spot.</section>
          ...
</chapter>
  ...
</book>
```

- Leerzeichen in der Formatierung werden eliminiert
- Interne DTD-Untergruppe wird nicht beibehalten
- Entität im Textknoten wird aufgelöst und ersetzt

Kapitel 10. Gespeicherte Prozeduren des XML Extender

Der XML Extender bietet gespeicherte Prozeduren zur Verwaltung von XML-Spalten und -Objektgruppen. Diese gespeicherten Prozeduren können vom DB2-Client aus aufgerufen werden. Die Client-Schnittstelle kann in SQL, ODBC oder JDBC integriert werden. Ausführliche Informationen zum Aufrufen der gespeicherten Prozeduren finden Sie im Abschnitt zu gespeicherten Prozeduren im Handbuch *DB2 UDB Systemverwaltung*.

Die gespeicherten Prozeduren verwenden das Schema db2xml, das dem Schemanamen des XML Extender entspricht.

Der XML Extender bietet drei Arten von gespeicherten Prozeduren:

- „Gespeicherte Prozeduren zur Verwaltung" auf Seite 223; diese Prozeduren unterstützen Benutzer bei der Ausführung von Verwaltungs-Tasks
- „Gespeicherte Prozeduren zum Zusammensetzen" auf Seite 231; diese Prozeduren generieren XML-Dokumente mit Hilfe von Daten in vorhandenen Datenbanktabellen
- „Gespeicherte Prozeduren zum Zerlegen" auf Seite 240; diese Prozeduren brechen ankommende XML-Dokumente herunter und speichern Daten in neuen oder bereits vorhandenen Datenbanktabellen

Die Parameterbegrenzungen für die gespeicherten Prozeduren der XML-Objektgruppe sind in „Anhang D. Die XML Extender-Begrenzungen" auf Seite 305 beschrieben.

Kopfdateien angeben

Stellen Sie sicher, daß Sie die externen Kopfdateien für den XML Extender in dem Programm angeben, das die gespeicherten Prozeduren aufruft. Die Kopfdateien befinden sich in dem Verzeichnis `DXX_INSTALL/include`. `DXX_INSTALL` ist das Installationsverzeichnis für den XML Extender. Dieses Verzeichnis hängt von dem verwendeten Betriebssystem ab. Die Kopfdateien sind:

dxx.h Die vom XML Extender definierten Konstanten und Datentypen

dxxrc.h Der XML Extender-Rückkehrcode

Die Syntax zur Angabe dieser Kopfdateien lautet:

```
#include "dxx.h"
#include "dxxrc.h"
```

Stellen Sie sicher, daß der Pfad für die Kopfdateien in der Make-Datei mit der Kompilierungsoption angegeben ist.

Gespeicherte Prozeduren des XML Extender aufrufen

Im allgemeinen rufen Sie den XML Extender mit der folgenden Syntax auf:

```
CALL db2xml.funktionseingabepunkt
```

Hierbei gilt folgendes:

db2xml
Gibt die Bibliothek der gespeicherten Prozeduren des XML Extender an. Bei UNIX-Betriebssystemen ist diese Bibliothek im Verzeichnis `sqllib/function` gespeichert. Bei Windows-Betriebssystemen ist sie im Verzeichnis *DXX_INSTALL*`/bin` gespeichert.

funktionseingabepunkt
Gibt die an die gespeicherte Prozedur übergebenen Argumente an.

In der CALL-Anweisung müssen die an die gespeicherte Prozeduren übergebenen Argumente die Host-Variablen sein, nicht Konstanten oder Ausdrücke. Die Host-Variablen können Nullanzeiger enthalten. Siehe hierzu die Beispiele zum Aufrufen gespeicherter Prozeduren in den Verzeichnissen `DXX_INSTALL/samples/c` und `DXX_INSTALL/samples/cli` sowie in den folgenden Abschnitten dieses Handbuchs: „XML-Dokument zusammensetzen" auf Seite 42 und „Kapitel 6. XML-Objektgruppendaten verwalten" auf Seite 147.

Im Verzeichnis `DXX_INSTALL/samples/c` wurden SQC-Codedateien bereitgestellt für den Aufruf von gespeicherten Prozeduren von XML-Objektgruppen mit eingebettetem SQL-Code. Die Musterdateien im Verzeichnis `DXX_INSTALL/samples/cli` zeigen, wie gespeicherte Prozeduren über CLI (Call Level Interface) aufgerufen werden.

Die CLOB-Begrenzung vergrößern

Die Standardbegrenzung für CLOB-Parameter beim Übermitteln an eine gespeicherte Prozedur ist 1 MB. Sie können diese Begrenzung vergrößern, indem Sie die folgenden Schritte ausführen:

1. Löschen aller gepeicherten Prozeduren. Beispiel:

   ```
   db2   "drop procedure db2xml.dxxShredXML"
   ```

2. Erstellen einer neuen Prozedur mit der vergrößerten CLOB-Begrenzung. Beispiel:

```
db2   "create procedure db2xml.dxxShredXML(in     dadBuf      clob(100K),
                                           in     XMLObj      clob(10M),
                                           out    returnCode  integer,
                                           out    returnMsg   varchar(1024)
                                           )
       external name 'db2xml.dxxShredXML'
       language C
       parameter style DB2DARI
       not deterministic
       fenced
       null call;
```

Vorbereitungen

Binden Sie Ihre Datenbank mit der gespeicherten Prozedur des XML Extender und den DB2-CLI-Bindedateien. Sie können eine Muster-Befehlsdatei `getstart_prep.cmd` zum Binden der Dateien verwenden. Diese Befehlsdatei befindet sich im Verzeichnis `DXX_INSTALL/samples/cmd`.

1. Stellen Sie die Verbindung zur Datenbank her. Beispiel:

   ```
   db2 "connect to SALES_DB"
   ```

2. Wechseln Sie in das Verzeichnis `DXX_INSTALL/bnd`, und binden Sie den XML Extender an die Datenbank.

   ```
   db2 "bind @dxxbind.lst"
   ```

3. Wechseln Sie in das Verzeichnis `sqllib/bnd`, und binden Sie die CLI an die Datenbank.

   ```
   db2 "bind @db2cli.lst"
   ```

4. Beenden Sie die Verbindung.

   ```
   db2 "terminate"
   ```

Gespeicherte Prozeduren zur Verwaltung

Diese gespeicherten Prozeduren werden für Verwaltungsaufgaben verwendet, beispielsweise zum Aktivieren oder Inaktivieren einer XML-Spalte oder -Objektgruppe. Sie werden vom XML Extender-Verwaltungsassistenten und dem Verwaltungsbefehl **dxxadm** aufgerufen.

dxxEnableDB()

Zweck

Aktiviert die Datenbank. Wenn die Datenbank aktiviert ist, erstellt der XML Extender die folgenden Objekte:

- Die benutzerdefinierten Typen (UDTs) des XML Extender.
- Die benutzerdefinierten Funktionen (UDFs) des XML Extender.
- Die XML Extender-DTD-Referenztabelle, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden. Eine vollständige Beschreibung der Tabelle DTD_REF finden Sie im Abschnitt „DTD-Referenztabelle" auf Seite 245.
- Die XML Extender-Verwendungstabelle, XML_USAGE, in der allgemeine Informationen für alle für XML aktivierten Spalten und Objektgruppen gespeichert werden. Eine vollständige Beschreibung der Tabelle XML_USAGE finden Sie im Abschnitt „XML-Verwendungstabelle" auf Seite 246.

```
dxxEnableDB(char(  dbName) dbName,       /* Eingabe */
            long           returnCode,   /* Ausgabe */
            varchar(1024)  returnMsg)    /* Ausgabe */
```

Parameter

Tabelle 41. Parameter für dxxEnableDB()

Parameter	Beschreibung	IN/OUT-Parameter
dbName	Der Datenbankname.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

dxxDisableDB()

Zweck

Inaktiviert die Datenbank. Wenn der XML Extender die Datenbank inaktiviert, werden die folgenden Objekte freigegeben:

- Die benutzerdefinierten Typen (UDTs) des XML Extender.
- Die benutzerdefinierten Funktionen (UDFs) des XML Extender.
- Die XML Extender-DTD-Referenztabelle, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden. Eine vollständige Beschreibung der Tabelle DTD_REF finden Sie im Abschnitt „DTD-Referenztabelle" auf Seite 245.
- Die XML Extender-Verwendungstabelle, XML_USAGE, in der allgemeine Informationen für alle für XML aktivierten Spalten und Objektgruppen gespeichert werden. Eine vollständige Beschreibung der Tabelle XML_USAGE finden Sie im Abschnitt „XML-Verwendungstabelle" auf Seite 246.

Wichtig: Sie müssen alle XML-Spalten inaktivieren, bevor Sie versuchen, eine Datenbank zu inaktivieren. Der XML Extender kann eine Datenbank nicht inaktivieren, wenn diese Spalten oder Objektgruppen enthält, die für XML aktiviert wurden.

```
dxxDisableDB(char(dbName)       dbName,       /* Eingabe */
                 long           returnCode,   /* Ausgabe */
                 varchar(1024) returnMsg)     /* Ausgabe */
```

Parameter

Tabelle 42. Parameter für dxxDisableDB()

Parameter	Beschreibung	IN/OUT-Parameter
dbName	Der Datenbankname.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

dxxEnableColumn()

Zweck

Aktiviert eine XML-Spalte. Beim Aktivieren einer Spalte führt der XML Extender die folgenden Tasks aus:

- Festlegen, ob die XML-Tabelle einen Primärschlüssel hat; falls nicht, ändert der XML Extender die XML-Tabelle und fügt die Spalte DXXROOT_ID hinzu.
- Erstellen der in der DAD-Datei angegebenen Seitentabellen mit einer Spalte für die eindeutige Kennung jeder Zeile in der XML-Tabelle. Diese Spalte ist entweder die vom Benutzer angegebene *root_id* oder die DXXROOT_ID, die vom XML Extender benannt wurde.
- Erstellen einer Standardsicht für die XML-Tabelle und ihrer Seitentabellen; wahlweise wird dabei ein von Ihnen angegebener Name verwendet.

```
dxxEnableColumn(char(dbName) dbName,      /* Eingabe */
                char(tbName) tbName,      /* Eingabe */
                char(colName) colName,    /* Eingabe */
                CLOB(100K)     DAD,        /* Eingabe */
                char(tablespace) tablespace,   /* Eingabe */
                char(defaultView) defaultView, /* Eingabe */
                char(rootID) rootID,       /* Eingabe */
                long           returnCode,  /* Ausgabe */
                varchar(1024)  returnMsg)   /* Ausgabe */
```

Parameter

Tabelle 43. Parameter für dxxEnableColumn()

Parameter	Beschreibung	IN/OUT-Parameter
dbName	Der Datenbankname.	IN
tbName	Der Name der Tabelle, in der sich die XML-Spalte befindet.	IN
colName	Der Name der XML-Spalte.	IN
DAD	Ein CLOB mit der DAD-Datei.	IN
tablespace	Der Tabellenbereich, der die Seitentabelle enthält, sofern es sich dabei nicht um den Standardtabellenbereich handelt. Ist kein Tabellenbereich angegeben, wird der Standard-tabellenbereich verwendet.	IN
defaultView	Der Name der Standardsicht, die die Anwendungstabelle und die Seitentabellen verknüpft.	IN

Tabelle 43. Parameter für dxxEnableColumn() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
rootID	Der Name des einzelnen Primärschlüssels in der Anwendungstabelle, der als *root_id* für die Seitentabelle verwendet werden soll.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

dxxDisableColumn()

Zweck

Inaktiviert die für XML aktivierte Spalte. Wenn eine XNML-Spalte inaktiviert ist, kann sie keine XML-Datentypen mehr enthalten.

```
dxxDisableColumn(char(dbName) dbName,        /* Eingabe */
                 char(tbName) tbName,        /* Eingabe */
                 char(colName) colName,      /* Eingabe */
                 long          returnCode,   /* Ausgabe */
                 varchar(1024)  returnMsg)   /* Ausgabe */
```

Parameter

Tabelle 44. Parameter für dxxDisableColumn()

Parameter	Beschreibung	IN/OUT-Parameter
dbName	Der Datenbankname.	IN
tbName	Der Name der Tabelle, in der sich die XML-Spalte befindet.	IN
colName	Der Name der XML-Spalte.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

dxxEnableCollection()

Zweck

Aktiviert eine XML-Objektgruppe, die einer Anwendungstabelle zugeordnet ist.

```
dxxEnableCollection(char(dbName) dbName,        /* Eingabe */
                    char(colName) colName,      /* Eingabe */
                    CLOB(100K)     DAD,         /* Eingabe */
                    char(tablespace) tablespace,/* Eingabe */
                    long           returnCode,  /* Ausgabe */
                    varchar(1024)  returnMsg)   /* Ausgabe */
```

Parameter

Tabelle 45. Parameter für dxxEnableCollection()

Parameter	Beschreibung	IN/OUT-Parameter
dbName	Der Datenbankname.	IN
colName	Der Name der XML-Objektgruppe.	IN
DAD	Ein CLOB mit der DAD-Datei.	IN
tablespace	Der Tabellenbereich, der die Seitentabelle enthält, sofern es sich dabei nicht um den Standardtabellenbereich handelt. Ist kein Tabellenbereich angegeben, wird der Standard-tabellenbereich verwendet.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

dxxDisableCollection()

Zweck

Inaktiviert eine für XML aktivierte Objektgruppe durch das Entfernen von Markierungen, die Tabellen und Spalten als Teil einer Objektgruppe kennzeichnen.

```
dxxDisableCollection(char(dbName) dbName,        /* Eingabe */
                     char(colName) colName,      /* Eingabe */
                     long          returnCode,   /* Ausgabe */
                     varchar(1024) returnMsg)    /* Ausgabe */
```

Parameter

Tabelle 46. Parameter für dxxDisableCollection()

Parameter	Beschreibung	IN/OUT-Parameter
dbName	Der Datenbankname.	IN
colName	Der Name der XML-Objektgruppe.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Gespeicherte Prozeduren zum Zusammensetzen

Die gespeicherten Prozeduren zum Zusammensetzen, dxxGenXML() und dxxRetrieveXML(), werden zum Generieren von XML-Dokumenten mit Daten in vorhandenen Datenbanktabellen verwendet. Die gespeicherte Prozedur dxxGenXML() verwendet eine DAD-Datei als Eingabe; sie erfordert keine aktivierte XML-Objektgruppe. Die gespeicherte Prozedur dxxRetrieveXML() verwendet den Namen einer aktivierten XML-Objektgruppe als Eingabe.

dxxGenXML()

Zweck

Konstruiert XML-Dokumente mit Daten, die in den XML-Objektgruppendateien gespeichert sind; diese Dateien werden über die <Xcollection> in der DAD-Datei angegeben. Außerdem fügt die Prozedur jedes XML-Dokument als Zeile in die Ergebnistabelle ein. Sie können auch einen Cursor in der Ergebnistabelle öffnen und die Ergebnisgruppe abrufen.

Für eine höhere Flexibilität gibt dxxGenXML() dem Benutzer außerdem die Möglichkeit, die maximale Anzahl der zu generierenden Zeilen in der Ergebnistabelle anzugeben. Dadurch wird die Dauer verringert, die die Anwendung während eines Testprozesses auf die Ergebnisse warten muß. Die gespeicherte Prozedur gibt die Anzahl der tatsächlichen Zeilen in der Tabelle zurück sowie alle Fehlerinformationen einschließlich Fehlercodes und Fehlermeldungen.

Zur Unterstützung dynamischer Abfragen verwendet dxxGenXML() den Eingabeparameter *override*. Entsprechend der Eingabe von *overrideType* kann die Anwendung die SQL_stmt-Anweisung für die SQL-Zuordnung überschreiben oder die Bedingungen in RDB_node für die RDB_node-Zuordnung in der DAD-Datei. Der Eingabeparameter *overrideType* wird verwendet, um den Typ von *override* klarzustellen. Ausführliche Informationen zum Parameter *override* finden Sie im Abschnitt „Werte in der DAD-Datei dynamisch überschreiben" auf Seite 152.

```
dxxGenXML(CLOB(100K)     DAD,                /* Eingabe */
          char(resultTabName) resultTabName, /* Eingabe */
          integer        overrideType        /* Eingabe */
          varchar(1024)  override,           /* Eingabe */
          integer        maxRows,            /* Eingabe */
          integer        numRows,            /* Ausgabe */
          long           returnCode,         /* Ausgabe */
          varchar(1024)  returnMsg)          /* Ausgabe */
```

Parameter

Tabelle 47. Parameter für dxxGenXML()

Parameter	Beschreibung	IN/OUT-Parameter
DAD	Ein CLOB mit der DAD-Datei.	IN
resultTabName	Der Name der Ergebnistabelle, diese Datei sollte vor dem Aufruf bereits vorhanden sein. Die Tabelle enthält nur eine Spalte mit dem Typ XMLVARCHAR oder XMLCLOB.	IN

Tabelle 47. Parameter für dxxGenXML() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
overrideType	Eine Markierung, die den Typ des Parameters *override* angibt: • **NO_OVERRIDE**: Kein Überschreiben. • **SQL_OVERRIDE**: Überschreiben durch eine SQL_stmt-Anweisung. • **XML_OVERRIDE**: Überschreiben durch eine XPath-Bedingung.	IN
override	Überschreibt die Bedingung in der DAD-Datei. Der Eingabewert basiert auf dem *overrideType*. • **NO_OVERRIDE**: Eine Nullzeichenfolge. • **SQL_OVERRIDE**: Eine gültige SQL-Anweisung. Für diesen *overrideType* muß die SQL-Zuordnung in der DAD-Datei verwendet werden. Die SQL-Eingabeanweisung überschreibt die SQL_stmt-Anweisung in der DAD-Datei. • **XML_OVERRIDE**: Eine Zeichenfolge, die einen oder mehrere Ausdrücke in doppelten Anführungszeichen, durch "AND" getrennt, enthält. Für diesen *overrideType* muß die RDB_node-Zuordnung in der DAD-Datei verwendet werden.	IN
maxRows	Die maximale Anzahl von Zeilen in der Ergebnistabelle.	IN
numRows	Die tatsächliche Anzahl generierter Zeilen in der Ergebnistabelle.	OUT

Tabelle 47. Parameter für dxxGenXML() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Beispiele

Im folgenden Beispiel wird davon ausgegangen, daß die Ergebnistabelle mit dem Namen XML_ORDER_TAB erstellt wird und eine Spalte des Typs XMLVARCHAR enthält.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE is CLOB(100K)  dad;              /* DAD */
  SQL TYPE is CLOB_FILE   dadFile;          /* DAD-Datei */
  char                    result_tab[32];   /* Name der Ergebnistabelle */
  char                    override[2];      /* Überschr., auf NULL gesetzt*/
  short                   overrideType;     /* definiert in dxx.h */
  short                   max_row;          /* maximale Anzahl Zeilen */
  short                   num_row;          /* tats. Anzahl Zeilen */
  long                    returnCode;       /* Rückkehr-Fehlercode */
  char                    returnMsg[1024];  /* Fehlernachrichtentext */
  short                   dad_ind;
  short                   rtab_ind;
  short                   ovtype_ind;
  short                   ov_inde;
  short                   maxrow_ind;
  short                   numrow_ind;
  short                   returnCode_ind;
  short                   returnMsg_ind;

  EXEC SQL END DECLARE SECTION;

  /* Tabelle erstellen */
  EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

  /* Daten aus einer Datei in ein CLOB einlesen */
  strcpy(dadfile.name,"e:\dxx\dad\litem3.dad");
  dadfile.name_length = strlen("e:\dxx\dad\litem3.dad");
  dadfile.file_options = SQL_FILE_READ;
  EXEC SQL VALUES (:dadfile) INTO :dad;
  strcpy(result_tab,"xml_order_tab");
  override[0] = '\0';
  overrideType = NO_OVERRIDE;
  max_row = 500;
  num_row = 0;
  returnCode = 0;
```

```
msg_txt[0] = '\0';
collection_ind = 0;
dad_ind = 0;
rtab_ind = 0;
ov_ind = -1;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxGenXML(:dad:dad_ind;
     :result_tab:rtab_ind,
     :overrideType:ovtype_ind,:override:ov_ind,
     :max_row:maxrow_ind,:num_row:numrow_ind,
     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxRetrieveXML()

Zweck

Ermöglicht die Verwendung derselben DAD-Datei beim Zusammensetzen und Zerlegen. Die gespeicherte Prozedur dxxRetrieveXML() dient auch als Mittel zum Abrufen zerlegter XML-Dokumente. Als Eingabe verwendet dxxRetrieveXML() einen Puffer, der die DAD-Datei, den Namen der erstellten Ergebnistabelle und die maximale Anzahl der zurückzugebenden Zeilen enthält. Die Prozedur gibt eine Ergebnisgruppe zurück aus der Ergebnistabelle, der tatsächlichen Anzahl von Zeilen in der Ergebnisgruppe, einem Fehlercode und einem Nachrichtentext.

Zur Unterstützung dynamischer Abfragen verwendet dxxRetrieveXML() den Eingabeparameter *override*. Entsprechend der Eingabe von *overrideType* kann die Anwendung die SQL_stmt-Anweisung für die SQL-Zuordnung überschreiben oder die Bedingungen in RDB_node für die RDB_node-Zuordnung in der DAD-Datei. Der Eingabeparameter *overrideType* wird verwendet, um den Typ von *override* klarzustellen. Ausführliche Informationen zum Parameter *override* finden Sie im Abschnitt „Werte in der DAD-Datei dynamisch überschreiben" auf Seite 152.

Die Anforderungen der DAD-Datei für dxxRetrieveXML() sind die gleichen wie die Anforderungen für dxxGenXML(). Der einzige Unterschied liegt darin, daß die DAD-Datei kein Eingabeparameter für dxxRetrieveXML() ist, sondern der Name einer aktivierten XML-Objektgruppe.

```
dxxRetrieveXML(char(collectionName) collectionName,  /* Eingabe */
               char(resultTabName) resultTabName,    /* Eingabe */
               integer        overrideType,     /* Eingabe */
               varchar(1024)  override,         /* Eingabe */
               integer        maxRows,          /* Eingabe */
               integer        numRows,          /* Ausgabe */
               long           returnCode,       /* Ausgabe */
               varchar(1024)  returnMsg)        /* Ausgabe */
```

Parameter

Tabelle 48. Parameter für dxxRetrieveXML()

Parameter	Beschreibung	IN/OUT-Parameter
collectionName	Der Name einer aktivierten XML-Objektgruppe	IN
resultTabName	Der Name der Ergebnistabelle, diese Datei sollte vor dem Aufruf bereits vorhanden sein. Die Tabelle enthält nur eine Spalte mit dem Typ XMLVARCHAR oder XMLCLOB.	IN

Tabelle 48. Parameter für dxxRetrieveXML() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
overrideType	Eine Markierung, die den Typ des Parameters *override* angibt: • **NO_OVERRIDE**: Kein Überschreiben. • **SQL_OVERRIDE**: Überschreiben durch eine SQL_stmt-Anweisung. • **XML_OVERRIDE**: Überschreiben durch eine XPath-Bedingung.	IN
override	Überschreibt die Bedingung in der DAD-Datei. Der Eingabewert basiert auf dem *overrideType*. • **NO_OVERRIDE**: Eine Nullzeichenfolge. • **SQL_OVERRIDE**: Eine gültige SQL-Anweisung. Für diesen *overrideType* muß die SQL-Zuordnung in der DAD-Datei verwendet werden. Die SQL-Eingabeanweisung überschreibt die SQL_stmt-Anweisung in der DAD-Datei. • **XML_OVERRIDE**: Eine Zeichenfolge, die einen oder mehrere Ausdrücke in doppelten Anführungszeichen, durch "AND" getrennt, enthält. Für diesen *overrideType* muß die RDB_node-Zuordnung in der DAD-Datei verwendet werden.	IN
maxRows	Die maximale Anzahl von Zeilen in der Ergebnistabelle.	IN
numRows	Die tatsächliche Anzahl generierter Zeilen in der Ergebnistabelle.	OUT

Tabelle 48. Parameter für dxxRetrieveXML() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Beispiele

Das folgende Beispiel zeigt einen Aufruf von dxxRetrieveXML(). In diesem Beispiel wird eine Ergebnistabelle mit dem Namen XML_ORDER_TAB erstellt, die eine Spalte des Typs XMLVARCHAR enthält.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char    collection[32];    /* DAD-Puffer */
              char    result_tab[32];    /* Name der Ergebnistabelle */
              char    override[2];       /* Überschreiben, auf NULL gesetzt*/
              short   overrideType;      /* definiert in dxx.h */
              short   max_row;           /* maximale Anzahl Zeilen */
              short   num_row;           /* tatsächliche Anzahl Zeilen */
              long    returnCode;        /* Rückkehr-Fehlercode */
              char    returnMsg[1024];   /* Fehlernachrichtentext */
              short   dadbuf_ind;
              short   rtab_ind;
              short   ovtype_ind;
              short   ov_inde;
              short   maxrow_ind;
              short   numrow_ind;
              short   returnCode_ind;
              short   returnMsg_ind;

              EXEC SQL END DECLARE SECTION;

              /* Tabelle erstellen */
              EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

              /* Host-Variable und Anzeiger initialisieren */
              strcpy(collection,"sales_ord");
              strcpy(result_tab,"xml_order_tab");
              override[0] = '\0';
              overrideType = NO_OVERRIDE;
              max_row = 500;
              num_row = 0;
              returnCode = 0;
              msg_txt[0] = '\0';
              collection_ind = 0;
              rtab_ind = 0;
              ov_ind = -1;
```

```
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
       :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
       :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Gespeicherte Prozeduren zum Zerlegen

Mit den gespeicherten Prozeduren zum Zerlegen dxxInsertXML() und dxxShredXML() werden ankommende XML-Dokumente zerlegt und Daten in neuen oder bereits vorhandenen Datenbanktabellen gespeichert. Die gespeicherte Prozedur dxxInsertXML() verwendet den Namen einer aktivierten XML-Objektgruppe als Eingabe. Die gespeicherte Prozedur dxxShredXML() verwendet eine DAD-Datei als Eingabe; sie erfordert keine aktivierte XML-Objektgruppe.

dxxShredXML()

Zweck

Die gespeicherte Prozedur dxxShredXML() ist das Gegenstück zur gespeicherten Prozedur dxxGenXML(). Damit dxxShredXML() funktioniert, müssen alle in der DAD-Datei angegebenen Tabellen vorhanden sein, und alle in der DAD-Datei angegebenen Spalten und ihre Datentypen müssen mit den vorhandenen Tabellen konsistent sein. Die gespeicherte Prozedur dxxShredXML() erfordert keine Primärschlüsselbeziehung zwischen den verknüpften Tabellen, die vom XML Extender beim Aktivieren der Objektgruppe erstellt wird. Die im RDB_node des Root-element_node angegebenen Spalten mit den Verknüpfungsbedingungen müssen jedoch in den Tabellen vorhanden sein.

```
dxxShredXML(CLOB(100K)    DAD,           /* Eingabe */
            CLOB(1M)      xmlobj,        /* Eingabe */
          long           returncode,     /* Ausgabe */
            varchar(1024) returnMsg)     /* Ausgabe */
```

Parameter

Tabelle 49. Parameter für dxxShredXML()

Parameter	Beschreibung	IN/OUT-Parameter
DAD	Ein CLOB mit der DAD-Datei.	IN
xmlobj	Ein XML-Dokumentobjekt mit dem Typ XMLCLOB.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Beispiele

Das folgende Beispiel zeigt einen Aufruf von dxxShredXML().

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              SQL TYPE is CLOB dad;           /* DAD*/
              SQL TYPE is CLOB_FILE  dadFile; /* DAD-Datei */
              SQL TYPE is CLOB      xmlDoc;   /* XML-Eingabedokument */
              SQL TYPE is CLOB_FILE xmlFile;  /* Eingabe-XMLfile */
              long        returnCode;         /* Fehlercode */
              char        returnMsg[1024];    /* Fehlernachrichtentext */
              short       dad_ind;
              short       xmlDoc_ind;
              short       returnCode_ind;
```

```
short       returnMsg_ind;
EXEC SQL END DECLARE SECTION;

/* Host-Variable und Anzeiger initialisieren */
strcpy(dadFile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
dadFile.name_length=strlen("c:\dxx\samples\dad\
getstart_xcollection.dad");
dadFile.file_option=SQL_FILE_READ;
strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart.xml");
xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart.xml");
xmlFile.file_option=SQL_FILE_READ;
SQL EXEC VALUES (:dadFile) INTO :dad;
SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
returnCode = 0;
returnMsg[0] = '\0';
dad_ind = 0;
xmlDoc_ind = 0;
returnCode_ind = -1;
returnMsg_ind  = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
                :xmlDoc:xmlDoc_ind,
                :returnCode:returnCode_ind,
                :returnMsg:returnMsg_ind);
```

dxxInsertXML()

Zweck

Verwendet zwei Eingabeparameter: den Namen einer aktivierten XML-Objektgruppe und das zu zerlegenden XML-Dokument; sie gibt zwei Ausgabeparameter aus: einen Rückkehrcode und eine Rückkehrnachricht.

```
dxxInsertXML(char(collectionName) collectionName, /* Eingabe */
             CLOB(1M)       xmlobj,           /* Eingabe */
             long           returnCode,       /* Ausgabe */
             varchar(1024)  returnMsg)        /* Ausgabe */
```

Parameter

Tabelle 50. Parameter für dxxInsertXML()

Parameter	Beschreibung	IN/OUT-Parameter
collectionName	Der Name einer aktivierten XML-Objektgruppe	IN
xmlobj	Ein XML-Dokumentobjekt mit dem Typ CLOB.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Beispiele

Im folgenden Beispiel zerlegt der Aufruf von dxxInsertXML() das XML-Eingabedokument `e:\xml\order1.xml` und fügt entsprechend der in der DAD-Datei angegebenen Zuordnung, mit der sie aktiviert wurde, Daten in die Gruppentabellen SALES_ORDER ein.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char           collection[64];  /* Name einer XML-Objektgruppe */
              SQL TYPE is CLOB_FILE xmlDoc;    /* XML-Eingabedokument */
              long           returnCode;       /* Fehlercode */
              char           returnMsg[1024]; /* Fehlernachrichtentext */
              short          collection_ind;
              short          xmlDoc_ind;
              short          returnCode_ind;
              short          returnMsg_ind;
              EXEC SQL END DECLARE SECTION;

              /* Host-Variable und Anzeiger initialisieren */
              strcpy(collection,"sales_ord")
              strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart.xml");
```

```
xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart.xml");
xmlobj.file_option=SQL_FILE_READ;
returnCode = 0;
returnMsg[0] = '\0';
collection_ind = 0;
xmlobj_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
                :xmlobj:xmlobj_ind,
                :returnCode:returnCode_ind,
                :returnMsg:returnMsg_ind);
```

Kapitel 11. Tabellen zur Verwaltungsunterstützung

Beim Aktivieren einer Datenbank werden eine DTD-Referenztabelle (DTD_REF) und eine Tabelle XML_USAGE erstellt. Die Tabelle DTD_REF enthält Informationen zu allen DTDs. Die Tabelle XML_USAGE speichert die gemeinsamen Informationen zu allen für XML aktivierten Spalten.

Die in den Unterstützungstabellen aufgelisteten Parameterbegrenzungen sind in „Anhang D. Die XML Extender-Begrenzungen" auf Seite 305 dokumentiert.

DTD-Referenztabelle

Der XML Extender dient auch als XML DTD-Repository. Beim Aktivieren einer Datenbank für XML wird eine DTD-Referenztabelle DTD_REF erstellt. Jede Zeile dieser Tabelle stellt eine DTD mit zusätzlichen Metadaten-Informationen dar. Die Benutzer können auf diese Tabelle zugreifen und ihre eigenen DTDs einfügen. Die DTDs in der Tabelle DTD_REF werden zum Überprüfen von XML-Dokumenten und zur Unterstützung der Anwendungen bei der Definition einer DAD-Datei verwendet. Sie hat den Schemanamen db2xml. Eine Tabelle DTD_REF kann die in Tabelle 51 aufgeführten Spalten enthalten.

Tabelle 51. DTD_REF-Tabelle

Spaltenname	Datentyp	Beschreibung
DTD-ID	VARCHAR(128)	Der Primärschlüssel (eindeutig und nicht NULL). Er wird zur Kennzeichnung der DTD verwendet. Wenn der Primärschlüssel in der DAD-Datei angegeben ist, muß die DAD-Datei entsprechend dem in der DTD definierten Schema aufgebaut sein.
CONTENT	XMLCLOB	Der Inhalt der DTD.
USAGE_COUNT	INTEGER	Die Anzahl der XML-Spalten und XML-Objektgruppen in der Datenbank, die diese DTD zur Definition ihrer DAD-Dateien verwenden.
AUTHOR	VARCHAR(128)	Autor der DTD, wahlfreie Informationen, die der Benutzer eingeben kann.
CREATOR	VARCHAR(128)	Die Benutzer-ID, über die die erste Einfügung vorgenommen wird. Die Spalte CREATOR ist wahlfrei.

Tabelle 51. DTD_REF-Tabelle (Forts.)

Spaltenname	Datentyp	Beschreibung
UPDATOR	VARCHAR(128)	Die Benutzer-ID, über die die erste Aktualisierung vorgenommen wird. Die Spalte UPDATOR ist wahlfrei.

Einschränkung: Die DTD kann von der Anwendung nur geändert werden, wenn USAGE_COUNT Null ist.

XML-Verwendungstabelle

Speichert die gemeinsamen Informationen zu allen für XML aktivierten Spalten. Der Schemaname der Tabelle XML_USAGE lautet db2xml, und ihr Primärschlüssel lautet (*table_name*, *col_name*). Eine Tabelle XML_USAGE mit den in Tabelle 52 aufgelisteten Spalten wird beim Aktivieren der Datenbank erstellt.

Tabelle 52. XML_USAGE-Tabelle

Spaltenname	Beschreibung
table_schema	Für XML-Spalten der Schemaname der Benutzertabelle, die eine XML-Spalte enthält. Für XML-Objektgruppen der Wert ″DXX_COLL″ als Standardschemaname.
table_name	Für XML-Spalten der Name der Benutzertabelle, die eine XML-Spalte enthält. Für XML-Objektgruppen der Wert ″DXX_COLLECTION,″ der die Entität als Objektgruppe kennzeichnet.
col_name	Der Name der XML-Spalte oder XML-Objektgruppe. Er ist zusammen mit table_name Teil des zusammengesetzten Schlüssels.
DTD-ID	Die ID der DTD in der Tabelle DTD_REF zur Definition der DAD-Datei. Dies ist der Fremdschlüssel.
DAD	Der Inhalt der DAD-Datei, die der Spalte zugeordnet ist.
default_view	Speichert den Namen der Standardsicht, falls eine Standardsicht vorhanden ist.
trigger_suffix	Keine Nullzeichenfolge. Für eindeutige Auslösernamen.

Tabelle 52. XML_USAGE-Tabelle (Forts.)

Spaltenname	Beschreibung
Validation	1 für ja (Gültigkeitsprüfung wird ausgeführt), 0 für Nein.
access_mode	1 für XML-Erfassung, 0 für XML-Spalte

Einschränkung: Die DTD kann von der Anwendung nur geändert werden, wenn USAGE_COUNT Null ist.

Kapitel 12. Diagnoseinformationen

Alle eingebetteten SQL-Anweisungen und Aufrufe der DB2-Befehlszeilenschnittstelle (CLI-Aufrufe) in Ihrem Programm, einschließlich der Aufrufe von benutzerdefinierten Funktionen (UDFs) des DB2 XML Extender, generieren Codes, die angeben, ob die eingebetteten SQL-Anweisung oder der DB2-CLI-Aufruf erfolgreich ausgeführt wurde.

Ihr Programm kann Informationen abrufen, die dieses Codes ergänzen. Hierzu gehören SQLSTATE-Informationen und Fehlernachrichten. Sie können diese Diagnoseinformationen zum Isolieren und Beheben von Problemen in Ihrem Programm verwenden.

Manchmal ist die Ursache eines Problems nicht einfach zu erkennen. In diesen Fällen müssen Sie zum Isolieren und Beheben des Problems Informationen an die Softwareunterstützung weiterleiten. Der XML Extender umfaßt eine Trace-Einrichtung, die die Aktivitäten des XML Extender aufzeichnet. Die Trace-Informationen können für die IBM Softwareunterstützung hilfreiche Angaben enthalten. Verwenden Sie die Trace-Einrichtung nur, wenn Sie von der IBM Softwareunterstützung dazu aufgefordert werden.

Dieses Kapitel beschreibt, wie auf die Diagnoseinformationen zugegriffen werden kann. Es beschreibt folgende Aktivitäten:

- XML Extender UDF-Rückkehrcodes verarbeiten
- Traces steuern

Außerdem werden die vom XML Extender eventuell zurückgegebenen SQLSTATE-Codes und Fehlernachrichten aufgelistet und beschrieben.

UDF-Rückkehrcodes verarbeiten

Rückkehrcodes von eingebetteten SQL-Anweisungen in den Feldern SQLCODE, SQLWARN und SQLSTATE der SQLCA-Struktur. diese Struktur ist in einer Kopfdatei SQLCA definiert. (Weitere Informationen zur SQLCA-Struktur und der Kopfdatei SQLCA finden Sie im Handbuch *DB2 Application Development Guide*.)

DB2-CLI-Aufrufe geben SQLCODE- und SQLSTATE-Werte zurück, die Sie über die Funktion SQLError abrufen können. (Weitere Informationen zum Abrufen von Fehlerinformationen mit der Funktion SQLError finden Sie im Handbuch *CLI Guide and Reference*.)

Der SQLCODE-Wert 0 bedeutet, daß die Anweisung erfolgreich (mit möglichen Warnungsbedingungen) ausgeführt wurde. Ein positiver SQLCODE-Wert bedeutet, daß die Anweisung erfolgreich, aber mit einer Warnung ausgeführt wurde. (Eingebettete SQL-Anweisungen geben Informationen zu der Warnung zurück, die dem SQLCODE-Wert 0 bzw. dem positiven SQLCODE-Wert im Feld SQLWARN zugeordnet ist.) Ein negativer SQLCODE-Wert bedeutet, daß ein Fehler aufgetreten ist.

DB2 ordnet jedem SQLCODE-Wert eine Nachricht zu. Wenn ein XML Extender UDF eine Warnungs- oder Fehlerbedingung feststellt, leitet sie die entsprechenden Informationen an DB2 weiter, wo sie in die SQLCODE-Nachricht eingebunden werden.

SQLSTATE-Werte enthalten Codes zur Ergänzung der SQLCODE-Nachrichten. Eine Beschreibung der vom XML Extender zurückgegebenen SQLSTATE-Codes finden Sie im Abschnitt „SQLSTATE-Codes" auf Seite 251.

Eingebettete SQL-Anweisungen und DB2-CLI-Aufrufe, die die DB2 XML Extender-UDFs aufrufen, geben eventuell SQLCODE-Nachrichten und SQLSTATE-Werte zurück, die eindeutig aus diesen UDFs stammen. DB2 gibt diese Werte jedoch auf die gleiche Weise zurück wie andere eingebettete SQL-Anweisungen oder andere DB2-CLI-Aufrufe. Sie rufen daher diese Werte auf die gleiche Weise auf wie eingebettete SQL-Anweisungen oder DB2-CLI-Aufrufe, die keine DB2 XML Extender-UDFs starten.

Der Abschnitt „SQLSTATE-Codes" auf Seite 251 enthält eine Übersicht über die SQLSTATE-Werte und die Nachrichtennummer der zugeordneten Nachrichten, die vom XML Extender zurückgegeben werden können. Informationen zu den einzelnen Nachrichten finden Sie im Abschnitt „Nachrichten" auf Seite 256.

Rückkehrcodes von gespeicherten Prozeduren verarbeiten

Der XML Extender bietet Rückkehrcodes, die das Lösen von Problemen mit gespeicherten Prozeduren unterstützen. Wenn Sie einen Rückkehrcode von einer gespeicherten Prozedur erhalten, überprüfen Sie die folgende Datei, die die XML Extender-Fehlernachrichtennummer und die symbolischen Konstante zu dem Rückkehrcode abgleicht.

```
DXX_INSTALL/include/dxxrc.h
```

Sie können auf die Fehlernachrichtennummer unter „Nachrichten" auf Seite 256 verweisen und die Diagnoseinformationen in der Erläuterung verwenden.

SQLSTATE-Codes

Tabelle 53 listet die vom XML Extender zurückgegebenen SQLSTATE-Werte und ihre Beschreibungen auf. Die Beschreibung der einzelnen SQLSTATE-Werte enthält die jeweilige symbolische Darstellung. Die Tabelle listet außerdem dem jeweiligen SQLSTATE-Wert zugeordnete Nachrichtennummer auf. Informationen zu den einzelnen Nachrichten finden Sie im Abschnitt „Nachrichten" auf Seite 256.

Tabelle 53. SQLSTATE-Codes und Nachrichtennummern

SQLSTATE	Nachrichtennr.	Beschreibung
00000	DXX*nnnn*I	Kein Fehler aufgetreten.
01HX0	DXXD003W	Das im Pfadausdruck angegebene Element oder Attribut ist im XML-Dokument nicht vorhanden.
38X00	DXXC000E	Der XML Extender kann die angegebene Datei nicht öffnen.
38X01	DXXA072E	Der XML Extender hat versucht, die Datenbank vor dem Aktivieren automatisch zu binden, konnte jedoch die Bindedateien nicht finden.
	DXXC001E	Der XML Extender konnte die angegebene Datei nicht finden.
38X02	DXXC002E	Der XML Extender kann keine Daten in der angegebenen Datei lesen.
38X03	DXXC003E	Der XML Extender kann keine Daten in die Datei schreiben.
	DXXC011E	Der XML Extender kann keine Daten in die Trace-Steuerdatei schreiben.
38X04	DXXC004E	Der XML Extender konnte die angegebene Zeigereinheit nicht bedienen.
38X05	DXXC005E	Die Dateigröße ist größer als die XMLVarchar-Größe, und der XML Extender kann nicht alle Daten aus der Datei importieren.
38X06	DXXC006E	Die Dateigröße ist größer als die XMLCLOB-Größe, und der XML Extender kann nicht alle Daten aus der Datei importieren.

Tabelle 53. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichtennr.	Beschreibung
38X07	DXXC007E	Die Anzahl der Byte in der LOB-Zeigereinheit entspricht nicht der Dateigröße.
38X08	DXXD001E	Eine Skalarfunktion zur Extraktion hat einen Standortpfad verwendet, der mehrfach auftritt. Eine Skalarfunktion kann nur einen Standortpfad verwenden, der nicht mehrfach vorkommt.
38X09	DXXD002E	Der Pfadausdruck ist syntaktisch fehlerhaft.
38X10	DXXG002E	Der XML Extender konnte keinen Speicher vom Betriebssystem zuordnen.
38X11	DXXA009E	Diese gespeicherte Prozedur gilt nur für XML-Spalten.
38X12	DXXA010E	Bei dem Versuch, die Spalte zu aktivieren, konnte der XML Extender die DTD-ID nicht finden. Diese DTD-ID ist die für die DTD in der DAD-Datei angegebene Kennung.
38X14	DXXD000E	Es wurde versucht, ein ungültiges Dokument in einer Tabelle zu speichern. Die Gültigkeitsprüfung ist fehlgeschlagen.
38X15	DXXA056E	Das Prüfungselement in der DAD-Datei ist fehlerhaft oder nicht vorhanden.
	DXXA057E	Das name-Attribut einer Seitentabelle in der DAD-Datei ist fehlerhaft oder nicht vorhanden.
	DXXA058E	Das name-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.
	DXXA059E	Das type-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.
	DXXA060E	Das path-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Tabelle 53. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichtennr.	Beschreibung
	DXXA061E	Das multi_occurrence-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.
	DXXQ000E	Ein verbindliches Element ist in der DAD-Datei nicht vorhanden.
38X16	DXXG004E	Ein Nullwert für einen erforderlichen Parameter wurde an eine gespeicherte XML-Prozedur übergeben.
38X17	DXXQ001E	Die SQL-Anweisung in der DAD-Datei oder der Datei, die sie überschreibt, ist nicht gültig. Zum Generieren von XML-Dokumenten ist eine SELECT-Anweisung erforderlich.
38X18	DXXG001E	Der XML Extender hat einen internen Fehler festgestellt.
	DXXG006E	Der XML Extender hat bei Verwendung der CLI einen internen Fehler festgestellt.
38X19	DXXQ002E	Das System hat nicht mehr genügend Platz im Speicher oder auf der Festplatte. Es ist kein Platz mehr für die resultierenden XML-Dokumente vorhanden.
38X20	DXXQ003W	Die benutzerdefinierte SQL-Abfrage generiert mehr XML-Dokumente als das angegebene Maximum. Es kann nur die angegebene Anzahl von Dokumenten zurückgegeben werden.
38X21	DXXQ004E	Die angegebene Spalte ist keine der Spalten im Ergebnis der SQL-Abfrage.
38X22	DXXQ005E	Die Zuordnung der SQL-Abfrage zu XML ist fehlerhaft.
38X23	DXXQ006E	Ein Element attribute_node in der DAD-Datei hat kein name-Attribut.

Tabelle 53. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichtennr.	Beschreibung
38X24	DXXQ007E	Das Element attribute_node in der DAD-Datei hat kein Spalten-element oder keinen RDB_node.
38X25	DXXQ008E	Ein Element text_node in der DAD-Datei hat kein Spalten-element.
38X26	DXXQ009E	Die angegebene Ergebnistabelle konnte im Systemkatalog nicht gefunden werden.
38X27	DXXQ010E	Der RDB_node des attribute_node oder text_node muß eine Tabelle haben.
	DXXQ011E	Der RDB_node des attribute_node oder text_node muß eine Spalte haben.
	DXXQ017E	Ein vom XML Extender generiertes XML-Dokument ist zu groß für die Spalte der Ergebnis-tabelle.
38X28	DXXQ012E	Der XML Extender konnte das erwartete Element bei der Verarbeitung der DAD nicht finden.
	DXXQ016E	Alle Tabellen müssen im RDB_node des Anfangselements in der DAD-Datei definiert werden. Tabellen der Unter-elemente müssen den im Anfangselement definierten Tabellen entsprechen. Der Tabellenname in diesem RDB_node ist im Anfangselement nicht enthalten.
38X29	DXXQ013E	Die Elementtabelle oder -spalte muß einen Namen in der DAD-Datei haben.
	DXXQ015E	Die Bedingung im condition-Element in der DAD-Datei hat ein ungültiges Format.
38X30	DXXQ014E	Ein Element element_node in der DAD-Datei hat kein name-Attribut.

Tabelle 53. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichtennr.	Beschreibung
	DXXQ018E	Die Klausel ORDER BY fehlt in der SQL-Anweisung in einer DAD-Datei, die eine Zuordnung zwischen SQL und XML herstellt.
38X31	DXXQ019E	Das objids-Element hat kein Spaltenelement in der DAD-Datei, die Zuordnung zwischen SQL und XML herstellt.
38X36	DXXA073E	Die Datenbank war nicht gebunden, als der Benutzer versuchte, sie zu aktivieren.
38X37	DXXG007E	Die länderspezifischen Angaben zum Server-Betriebssystem sind nicht konsistent mit der DB2-Codepage.
38X38	DXXG008E	Die länderspezifischen Angaben zum Server-Betriebssystem können in der Codepage-Tabelle nicht gefunden werden.
38x33	DXXG005E	Dieser Parameter ist in diesem Release nicht unterstützt; er wird in späteren Versionen unterstützt.
38x34	DXXG000E	Es wurde ein ungültiger Name angegeben.

Nachrichten

Der XML Extender bietet Fehlernachrichten zur Unterstützung der Fehlerbestimmung.

Fehlernachrichten

Der XML Extender generiert die folgenden Nachrichten, wenn er eine Operation abschließt oder einen Fehler erkennt.

DXXA000I **Spalte** <*Spaltenname*> **wird aktiviert. Bitte warten Sie.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA001S **Ein unerwarteter Fehler ist in Build** <*build_ID*>**, Datei** <*dateiname*> **und Zeile** <*zeilennummer*> **aufgetreten.**

Erläuterung: Es ist ein unerwarteter Fehler aufgetreten.

Benutzeraktion: Wenn dieser Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter. Geben Sie beim Weitermelden des Fehlers den gesamten Nachrichtentext an und eine Erläuterung, wie der Fehler reproduziert werden kann, und liefern Sie auch die Trace-Datei mit.

DXXA002I **Verbindung zur Datenbank** <*datenbank*> **wird hergestellt.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA003E **Zur Datenbank** <*datenbank*> **kann keine Verbindung hergestellt werden.**

Erläuterung: Die angegebene Datenbank ist eventuell nicht vorhanden oder defekt.

Benutzeraktion:

1. Stellen Sie sicher, daß die Datenbank korrekt angegeben wurde.
2. Stellen Sie sicher, daß die Datenbank vorhanden ist und aufgerufen werden kann.
3. Stellen Sie fest, ob die Datenbank beschädigt ist. Ist dies der Fall, bitten Sie Ihren Administrator, sie von einer Sicherung wiederherzustellen.

DXXA004E **Datenbank** <*datenbank*> **kann nicht aktiviert werden.**

Erläuterung: Die Datenbank wurde eventuell bereits aktiviert, oder sie ist beschädigt.

Benutzeraktion:

1. Stellen Sie fest, ob die Datenbank aktiviert ist.
2. Stellen Sie fest, ob die Datenbank beschädigt ist. Ist dies der Fall, bitten Sie Ihren Administrator, sie von einer Sicherung wiederherzustellen.

DXXA005I **Datenbank** <*datenbank*> **wird aktiviert. Bitte warten Sie.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA006I **Die Datenbank** <*datenbank*> **wurde erfolgreich aktiviert.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA007E **Datenbank** <*datenbank*> **kann nicht inaktiviert werden.**

Erläuterung: Die Datenbank kann von XML Extender nicht inaktiviert werden, wenn sie

XML-Spalten oder -Objektgruppen enthält.

Benutzeraktion: Sichern Sie alle wichtigen Daten, inaktivieren Sie alle XML-Spalten oder -Objektgruppen, und aktivieren Sie alle Tabellen bzw. geben Sie sie frei, bis keine XML-Datentypen mehr in der Datenbank vorhanden sind.

DXXA008I Spalte <*spaltenname*> **wird inaktiviert. Bitte warten Sie.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA009E Befehl Xcolumn wird in der DAD-Datei nicht angegeben.

Erläuterung: Diese gespeicherte Prozedur gilt nur für XML-Spalten.

Benutzeraktion: Stellen Sie sicher, daß der Befehl Xcolumn in der DAD-Datei richtig angegeben ist.

DXXA010E Versuch, DTD-ID <*dtdid*> **zu finden, ist fehlgeschlagen.**

Erläuterung: Bei dem Versuch, die Spalte zu aktivieren, konnte der XML Extender die DTD-ID nicht finden. Diese DTD-ID ist die für die DTD in der DAD-Datei angegebene Kennung.

Benutzeraktion: Stellen Sie sicher, daß der richtige Wert für für die DTD-ID in der DAD-Datei angegeben ist.

DXXA011E Einfügen in Tabelle DB2XML.XML_USAGE ist fehlgeschlagen.

Erläuterung: Bei dem Versuch, die Spalte zu aktivieren, konnte der XML Extender keinen Datensatz in die Tabelle DB2XML.XML_USAGE einfügen.

Benutzeraktion: Stellen Sie sicher, daß die Tabelle DB2XML.XML_USAGE vorhanden ist und daß in der Tabelle nicht bereits ein Datensatz mit diesem Namen vorhanden ist.

DXXA012E Der Versuch, die Tabelle DB2XML.DTD_REF zu aktualisieren, ist fehlgeschlagen.

Erläuterung: Bei dem Versuch, die Spalte zu aktivieren, konnte der XML Extender die Tabelle DB2XML.DTD_REF nicht aktualisieren.

Benutzeraktion: Stellen Sie sicher, daß die Tabelle DB2XML.DTD_REF vorhanden ist. Stellen Sie fest, ob die Datenbank defekt ist oder ob die Verwaltungs-Benutzer-ID eine ausreichende Berechtigung zum Aktualisieren der Tabelle hat.

DXXA013E Der Versuch, die Tabelle <*tabellenname*> **zu ändern, ist fehlgeschlagen.**

Erläuterung: Bei dem Versuch, die Spalte zu aktivieren, konnte der XML Extender die angegebene Tabelle nicht ändern.

Benutzeraktion: Überprüfen Sie die erforderlichen Berechtigungen zum Ändern der Tabelle.

DXXA014E Die angegebene Root-ID-Spalte: <*root_id*> **ist kein einzelner Primärschlüssel von Tabelle** <*tabelle*>**.**

Erläuterung: Die angegebene Root-ID ist entweder kein Schlüssel oder kein singulärer Schlüssel der Tabelle *tabelle*.

Benutzeraktion: Stellen Sie sicher, daß die angegebene Root-ID der singuläre Primärschlüssel der Tabelle ist.

DXXA015E Die Spalte DXXROOT_ID ist in Tabelle <*tabelle*> **bereits vorhanden.**

Erläuterung: Die Spalte DXXROOT_ID ist vorhanden, wurde jedoch nicht vom XML Extender erstellt.

Benutzeraktion: Geben Sie beim Aktivieren einer Spalte eine primäre Spalte für die Root-ID-Option an, und verwenden Sie dabei einen anderen Spaltennamen.

DXXA016E Die Eingabetabelle <*tabelle*> **ist nicht vorhanden.**

Erläuterung: Der XML Extender konnte die angegebene Tabelle im Systemkatalog nicht finden.

Benutzeraktion: Stellen Sie sicher, daß die Tabelle in der Datenbank vorhanden ist, und geben Sie sie korrekt an.

DXXA017E Die Eingabespalte <*spaltenname*> **ist in der angegebenen Tabelle** <*tabellenname*> **nicht vorhanden.**

Erläuterung: Der XML Extender konnte die Spalte im Systemkatalog nicht finden.

Benutzeraktion: Stellen Sie sicher, daß die Spalte in einer Benutzertabelle vorhanden ist.

DXXA018E Die angegebene Spalte ist nicht für XML-Daten aktiviert.

Erläuterung: Bei dem Versuch, die Spalte zu inaktivieren, konnte der XML Extender die Spalte in der Tabelle DB2XML.XML_USAGE nicht finden; dies bedeutet, daß die Tabelle nicht aktiviert ist. Wenn die Spalte nicht für XML aktiviert ist, brauchen Sie sie nicht zu inaktivieren.

Benutzeraktion: Keine Aktion erforderlich.

DXXA019E Ein Eingabeparameter, der für die Aktivierung der Spalte erforderlich ist, hat den Wert Null.

Erläuterung: Ein erforderlicher Eingabeparameter für die gespeicherte Prozedur enable_column() ist Null.

Benutzeraktion: Überprüfen Sie alle Eingabeparameter für die gespeicherte Prozedur enable_column().

DXXA020E In der Tabelle <*tabelle*> **können keine Spalten gefunden werden.**

Erläuterung: Bei dem Versuch, die Standardsicht zu erstellen, konnte der XML Extender keine Spalten in der angegebenen Tabelle finden.

Benutzeraktion: Stellen Sie sicher, daß der Spalten- und Tabellenname korrekt angegeben wurde.

DXXA021E Die Standardsicht <*standardsicht*> **kann nicht erstellt werden.**

Erläuterung: Bei dem Versuch, eine Spalte zu aktivieren, konnte der XML Extender die angegebene Sicht nicht erstellen.

Benutzeraktion: Stellen Sie sicher, daß der Name der Standardsicht eindeutig ist. Wenn bereits eine Sicht mit diesem Namen vorhanden ist, geben Sie einen eindeutigen Namen für die Standardsicht an.

DXXA022I Spalte <*spaltenname*> **aktiviert.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA023E Die DAD-Datei kann nicht gefunden werden.

Erläuterung: Bei dem Versuch, eine Spalte zu inaktivieren, konnte der XML Extender die DAD-Datei nicht finden.

Benutzeraktion: Stellen Sie sicher, daß Sie den richtigen Datenbank-, Tabellen- und Spaltennamen angegeben haben.

DXXA024E Der XML Extender ist beim Zugriff auf die Systemkatalogtabellen auf einen internen Fehler gestoßen.

Erläuterung: Der XML Extender konnte nicht auf die Systemkatalogtabelle zugreifen.

Benutzeraktion: Stellen Sie sicher, daß die Datenbank in einem stabilen Zustand ist.

DXXA025E Die Standardsicht <*standardsicht*> **kann nicht freigegeben (Drop) werden.**

Erläuterung: Bei dem Versuch, eine Spalte zu inaktivieren, konnte der XML Extender die

Standardsicht nicht freigeben.

Benutzeraktion: Stellen Sie sicher, daß die Administrator-ID für den XML Extender die erforderlichen Berechtigungen zum Freigeben der Standardsicht hat.

DXXA026E Die Seitentabelle <*seitentabelle*> kann nicht freigegeben (Drop) werden.

Erläuterung: Bei dem Versuch, eine Spalte zu inaktivieren, konnte der XML Extender die angegebene Tabelle nicht freigeben.

Benutzeraktion: Stellen Sie sicher, daß die Administrator-ID für den XML Extender die erforderlichen Berechtigungen zum Freigeben der Tabelle hat.

DXXA027E Die Spalte konnte nicht inaktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht inaktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA028E Die Spalte konnte nicht inaktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht inaktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA029E Die Spalte konnte nicht inaktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht inaktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA030E Die Spalte konnte nicht inaktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht inaktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA031E Der DXXROOT_ID-Spaltenwert in der Anwendungstabelle konnte nicht auf NULL zurückgesetzt werden.

Erläuterung: Bei dem Versuch, eine Spalte zu inaktivieren, konnte der XML Extender den Wert von DXXROOT_ID in der Anwendungstabelle nicht auf NULL setzen.

Benutzeraktion: Stellen Sie sicher, daß die Administrator-ID für den XML Extender die erforderlichen Berechtigungen zum Ändern der Anwendungstabelle hat.

DXXA032E Verminderung von USAGE_COUNT in der Tabelle DB2XML.XML_USAGE ist fehlgeschlagen.

Erläuterung: Bei dem Versuch, die Spalte zu inaktivieren, konnte der XML Extender den Wert

der Spalte USAGE_COUNT nicht um 1 reduzieren.

Benutzeraktion: Stellen Sie sicher, daß die Tabelle DB2XML.XML_USAGE vorhanden ist und daß die Administrator-ID für den XML Extender die erforderlichen Berechtigungen zum Aktualisieren der Tabelle hat.

DXXA033E Der Versuch, eine Zeile in der Tabelle DB2XML.XML_USAGE zu löschen, ist fehlgeschlagen.

Erläuterung: Bei dem Versuch, eine Spalte zu inaktivieren, konnte der XML Extender die entsprechende Zeile in der Tabelle DB2XML.XML_USAGE nicht löschen.

Benutzeraktion: Stellen Sie sicher, daß die Tabelle DB2XML.XML_USAGE vorhanden ist und daß die Administrator-ID für den XML Extender die erforderlichen Berechtigungen zum Aktualisieren dieser Tabelle hat.

DXXA034I XML Extender hat Spalte <*spaltenname*> **erfolgreich inaktiviert.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA035I XML Extender inaktiviert Datenbank <*datenbank*>**. Bitte warten Sie.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA036I XML Extender hat Datenbank <*datenbank*> **erfolgreich inaktiviert.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA037E Der angegebene Tabellenbereichsname ist länger als 18 Zeichen.

Erläuterung: Der Name des Tabellenbereichs kann nicht länger als 18 alphanumerische Zeichen sein.

Benutzeraktion: Geben Sie einen Namen mit weniger als 18 Zeichen ein.

DXXA038E Der angegebene Name für die Standardsicht ist länger als 18 Zeichen.

Erläuterung: Der Name der Standardsicht kann nicht länger als 18 alphanumerische Zeichen sein.

Benutzeraktion: Geben Sie einen Namen mit weniger als 18 Zeichen ein.

DXXA039E Der angegebene ROOT_ID-Name ist länger als 18 Zeichen.

Erläuterung: Der Name der ROOT_ID kann nicht länger als 18 alphanumerische Zeichen sein.

Benutzeraktion: Geben Sie einen Namen mit weniger als 18 Zeichen ein.

DXXA046E Die Seitentabelle <*seitentabelle*> **konnte nicht erstellt werden.**

Erläuterung: Bei dem Versuch, eine Spalte zu aktivieren, konnte der XML Extender die angegebene Seitentabelle nicht erstellen.

Benutzeraktion: Stellen Sie sicher, daß die Administrator-ID für den XML Extender die erforderlichen Berechtigungen zum Erstellen der Seitentabelle hat.

DXXA047E Die Spalte konnte nicht aktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht aktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an

Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA048E Die Spalte konnte nicht aktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht aktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA049E Die Spalte konnte nicht aktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht aktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA050E Die Spalte konnte nicht aktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht aktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA051E Die Spalte konnte nicht inaktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht inaktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA052E Die Spalte konnte nicht inaktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht inaktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA053E Die Spalte konnte nicht aktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht aktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA054E Die Spalte konnte nicht aktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht aktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

Benutzeraktion: Verwenden Sie die Trace-Einrichtung zum Erstellen einer Trace-Datei, und versuchen Sie, das Problem zu beheben. Wenn der Fehler weiterhin besteht, wenden Sie sich an Ihren Softwareserviceanbieter, und stellen Sie ihm die Trace-Datei zur Verfügung.

DXXA056E Der Gültigkeitsprüfungswert <*prüfungswert*> **in der DAD-Datei ist ungültig.**

Erläuterung: Das Prüfungselement in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, daß das Prüfungselement in der DAD-Datei korrekt angegeben ist.

DXXA057E Ein Seitentabellenname <*seitentabelle*> **in der DAD ist ungültig.**

Erläuterung: Das name-Attribut einer Seitentabelle in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, daß das name-Attribut einer Seitentabelle in der DAD-Datei korrekt angegeben ist.

DXXA058E Ein Spaltenname <*spaltenname*> **in der DAD-Datei ist ungültig.**

Erläuterung: Das name-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, daß das name-Attribut einer Spalte in der DAD-Datei korrekt angegeben ist.

DXXA059E Der Typ <*spaltentyp*> **von Spalte** <*spaltenname*> **in der DAD-Datei ist ungültig.**

Erläuterung: Das type-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, daß das type-Attribut einer Spalte in der DAD-Datei korrekt angegeben ist.

DXXA060E Das path_exp-Attribut <*standortpfad*> **(path_exp=Pfadausdruck) von** <*spaltenname*> **in der DAD-Datei ist ungültig.**

Erläuterung: Das path-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, daß das path-Attribut einer Spalte in der DAD-Datei korrekt angegeben ist.

DXXA061E Das multi_occurrence-Attribut <*mehrfachvorkommen*> **von** <*spaltenname*> **in der DAD-Datei ist ungültig.**

Erläuterung: Das multi_occurrence-Attribut einer Spalte in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, daß das multi_occurrence-Attribut einer Spalte in der DAD-Datei korrekt angegeben ist.

DXXA062E Die Spaltennummer für <*spaltenname*> **in der Tabelle** <*tabelle*> **konnte nicht abgerufen werden.**

Erläuterung: Der XML Extender konnte die Spaltennummer für *spaltenname* in der Tabelle *tabelle* aus dem Systemkatalog nicht abrufen.

Benutzeraktion: Stellen Sie sicher, daß die Anwendungstabelle richtig definiert ist.

DXXA063I Objektgruppe <*gruppe*> **wird aktiviert. Bitte warten Sie.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA064I Objektgruppe <*gruppe*> **wird inaktiviert. Bitte warten Sie.**

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA065E Der Aufruf der gespeicherten Prozedur <*prozedurname*> ist fehlgeschlagen.

Erläuterung: Überprüfen Sie die gemeinsame Bibliothek db2xml und stellen Sie fest, ob die Berechtigung ausreicht.

Benutzeraktion: Stellen Sie sicher, daß der Client die Berechtigung zur Ausführung der gespeicherten Prozedur hat.

DXXA066I XML Extender hat Objektgruppe <*gruppe*> erfolgreich inaktiviert.

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA067I XML Extender hat Objektgruppe <*gruppe*> erfolgreich aktiviert.

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA068I XML Extender hat den Trace erfolgreich eingeschaltet.

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA069I XML Extender hat den Trace erfolgreich ausgeschaltet.

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXA070W Die Datenbank wurde bereits aktiviert.

Erläuterung: Der Befehl zum Aktivieren der Datenbank wurde mit einer bereits aktivierten Datenbank ausgeführt.

Benutzeraktion: Keine Aktion erforderlich.

DXXA071W Die Datenbank wurde bereits inaktiviert.

Erläuterung: Der Befehl zum Inaktivieren der Datenbank wurde mit einer bereits inaktivierten Datenbank ausgeführt.

Benutzeraktion: Keine Aktion erforderlich.

DXXA072E XML Extender konnte die Bindedateien nicht finden. Bitte binden Sie die Datenbank, bevor Sie sie aktivieren.

Erläuterung: Der XML Extender hat versucht, die Datenbank vor dem Aktivieren automatisch zu binden, konnte jedoch die Bindedateien nicht finden.

Benutzeraktion: Bitte binden Sie die Datenbank, bevor Sie sie aktivieren.

DXXA073E Die Datenbank ist nicht gebunden. Bitte binden Sie die Datenbank, bevor Sie sie aktivieren.

Erläuterung: Die Datenbank war nicht gebunden, als der Benutzer versuchte, sie zu aktivieren.

Benutzeraktion: Bitte binden Sie die Datenbank, bevor Sie sie aktivieren.

DXXA074E Falsche Parameterart. Von der gespeicherten Prozedur wird ein Parameter STRING erwartet.

Erläuterung: Die gespeicherte Prozedur erwartet einen Parameter STRING.

Benutzeraktion: Deklarieren Sie den Eingabeparameter als STRING-Typ.

DXXA075E Falsche Parameterart. Der Eingabeparameter sollte in der Art LONG vorliegen.

Erläuterung: Die gespeicherte Prozedur erwartet den Eingabeparameter als LONG-Typ.

Benutzeraktion: Deklarieren Sie den Eingabeparameter als LONG-Typ.

DXXA076E Die Trace-Exemplar-ID des XML Extender ist ungültig.

Erläuterung: Mit der angegebenen Exemplar-ID kann kein Trace gestartet werden.

Benutzeraktion: Stellen Sie sicher, daß es sich bei der Exemplar-ID um eine gültige AS/400-Benutzer-ID handelt.

DXXC000E Die angegebene Datei konnte nicht geöffnet werden.

Erläuterung: Der XML Extender kann die angegebene Datei nicht öffnen.

Benutzeraktion: Stellen Sie sicher, daß die Anwendungs-Benutzer-ID die Lese- und Schreibberechtigung für die Datei hat.

DXXC001E Die angegebene Datei wurde nicht gefunden.

Erläuterung: Der XML Extender konnte die angegebene Datei nicht finden.

Benutzeraktion: Stellen Sie sicher, daß die Datei vorhanden ist und der Pfad richtig angegeben wurde.

DXXC002E Die Datei konnte nicht gelesen werden.

Erläuterung: Der XML Extender kann keine Daten in der angegebenen Datei lesen.

Benutzeraktion: Stellen Sie sicher, daß die Anwendungs-Benutzer-ID die Leseberechtigung für die Datei hat.

DXXC003E In die angegebene Datei konnte nicht geschrieben werden.

Erläuterung: Der XML Extender kann keine Daten in die Datei schreiben.

Benutzeraktion: Stellen Sie sicher, daß die Anwendungs-Benutzer-ID die Schreibberechtigung für die Datei hat und das Dateisystem genügend Speicherplatz aufweist.

DXXC004E Der LOB-Zeiger konnte nicht bedient werden: rc=<*zeigereinheit_rc*>.

Erläuterung: Der XML Extender konnte die angegebene Zeigereinheit nicht bedienen.

Benutzeraktion: Stellen Sie sicher, daß die LOB-Zeigereinheit richtig gesetzt ist.

DXXC005E Die Größe der Eingabedatei übersteigt die Größe von XMLVarchar.

Erläuterung: Die Dateigröße ist größer als die XMLVarchar-Größe, und der XML Extender kann nicht alle Daten aus der Datei importieren.

Benutzeraktion: Verwenden Sie den Spaltentyp XMLCLOB.

DXXC006E Die Eingabedatei übersteigt die DB2-LOB-Begrenzung.

Erläuterung: Die Dateigröße ist größer als die XMLCLOB-Größe, und der XML Extender kann nicht alle Daten aus der Datei importieren.

Benutzeraktion: Zerlegen Sie die Datei in kleinere Objekte, oder verwenden Sie eine XML-Objektgruppe.

DXXC007E Von der Datei konnten keine Daten für den LOB-Zeiger abgerufen werden.

Erläuterung: Die Anzahl der Byte in der LOB-Zeigereinheit entspricht nicht der Dateigröße.

Benutzeraktion: Stellen Sie sicher, daß die LOB-Zeigereinheit richtig gesetzt ist.

DXXC008E Datei <*dateiname*> kann nicht entfernt werden.

Erläuterung: Die Datei weist eine Zugriffsverletzung auf oder ist noch geöffnet.

Benutzeraktion: Schließen Sie die Datei, oder stoppen Sie alle Prozesse, die die Datei geöffnet halten. Eventuell müssen Sie hierzu DB stoppen und erneut starten.

DXXC009E Datei kann nicht in Verzeichnis <*verzeichnis*> erstellt werden.

Erläuterung: Der XML Extender kann keine Datei im Verzeichnis *verzeichnis* erstellen.

Benutzeraktion: Stellen Sie sicher, daß das Verzeichnis vorhanden ist, daß die Anwendungs-Benutzer-ID die Schreibberechtigung für das Verzeichnis Datei hat und das Dateisystem genügend Speicherplatz für die Datei aufweist.

DXXC010E Fehler beim Schreiben in Datei <*dateiname*>.

Erläuterung: Es trat ein Fehler beim Schreiben in die Datei *dateiname* auf.

Benutzeraktion: Stellen Sie sicher, daß das Dateisystem genügend Speicherplatz für die Datei aufweist.

DXXC011E In die Trace-Steuerungsdatei konnte nicht geschrieben werden.

Erläuterung: Der XML Extender kann keine Daten in die Trace-Steuerdatei schreiben.

Benutzeraktion: Stellen Sie sicher, daß die Anwendungs-Benutzer-ID die Schreibberechtigung für die Datei hat und das Dateisystem genügend Speicherplatz aufweist.

DXXC012E Temporäre Datei kann nicht erstellt werden.

Erläuterung: Es kann keine Datei im temporären Systemverzeichnis erstellt werden.

Benutzeraktion: Stellen Sie sicher, daß die Anwendungs-Benutzer-ID die Schreibberechtigung für das temporäre Systemverzeichnis hat und das Dateisystem genügend Speicherplatz für die Datei aufweist.

DXXD000E Ein ungültiges XML-Dokument wird zurückgewiesen.

Erläuterung: Es wurde versucht, ein ungültiges Dokument in einer Tabelle zu speichern. Die Gültigkeitsprüfung ist fehlgeschlagen.

Benutzeraktion: Überprüfen Sie das Dokument und seine DTD mit einem Editor, der nicht sichtbare, ungültige Zeichen anzeigen kann. Schalten Sie zum Unterdrücken dieses Fehlers die Gültigkeitsprüfung in der DAD-Datei aus.

DXXD001E Pfad <*standortpfad*> ist mehrfach vorhanden.

Erläuterung: Eine Skalarfunktion zur Extraktion hat einen Standortpfad verwendet, der mehrfach auftritt. Eine Skalarfunktion kann nur einen Standortpfad verwenden, der nicht mehrfach vorkommt.

Benutzeraktion: Verwenden Sie eine Tabellenfunktion (fügen Sie ein 's' an das Ende des Skalarfunktionsnamens an).

DXXD002E In der Nähe der Position <*position*> im Suchpfad ist ein Syntaxfehler aufgetreten.

Erläuterung: Der Pfadausdruck ist syntaktisch fehlerhaft.

Benutzeraktion: Korrigieren Sie das Suchpfadargument der Abfrage. Schlagen Sie die Syntax für Pfadausdrücke in der Dokumentation nach.

DXXD003W Pfad wurde nicht gefunden. Der Wert Null wird zurückgegeben.

Erläuterung: Das im Pfadausdruck angegebene Element oder Attribut ist im XML-Dokument nicht vorhanden.

Benutzeraktion: Überprüfen Sie, ob der angegebene Pfad richtig ist.

DXXG000E Der Dateiname <*dateiname*> ist ungültig.

Erläuterung: Es wurde ein ungültiger Name angegeben.

Benutzeraktion: Geben Sie einen richtigen Dateinamen ein, und versuchen Sie es erneut.

DXXG001E Ein interner Fehler ist in Build <*build_ID*>, Datei <*dateiname*> und Zeile <*zeilennummer*> aufgetreten.

Erläuterung: Der XML Extender hat einen internen Fehler festgestellt.

Benutzeraktion: Wenden Sie sich an Ihren Softwareserviceanbieter. Geben Sie beim Weitermelden des Fehlers alle Nachrichten an, die Trace-Datei und eine Erläuterung, wie der Fehler reproduziert werden kann.

DXXG002E Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.

Erläuterung: Der XML Extender konnte keinen Speicher vom Betriebssystem zuordnen.

Benutzeraktion: Schließen Sie einige Anwendungen, und versuchen Sie es erneut. Wenn der Fehler weiterhin auftritt, schlagen Sie entsprechende Maßnahmen in der Dokumentation zu Ihrem Betriebssystem nach. Bei manchen Betriebssystemen müssen Sie das System neu starten, um das Problem zu beheben.

DXXG004E Ungültiger Nullparameter.

Erläuterung: Ein Nullwert für einen erforderlichen Parameter wurde an eine gespeicherte XML-Prozedur übergeben.

Benutzeraktion: Überprüfen Sie alle erforderlichen Parameter in der Argumentliste für den Aufruf der gespeicherten Prozedur.

DXXG005E Parameter nicht unterstützt.

Erläuterung: Dieser Parameter ist in diesem Release nicht unterstützt; er wird in späteren Versionen unterstützt.

Benutzeraktion: Setzen Sie diesen Parameter auf NULL.

DXXG006E Interner Fehler CLISTATE=<*clistatus*>, RC=<*cli_rc*>, Build <*build_ID*>, Datei <*dateiname*>, Zeile <*zeilennummer*> CLIMSG=<*CLI_msg*>.

Erläuterung: Der XML Extender hat bei Verwendung der CLI einen internen Fehler festgestellt.

Benutzeraktion: Wenden Sie sich an Ihren Softwareserviceanbieter. Dieser Fehler kann durch eine falsche Benutzereingabe verursacht worden sein. Geben Sie beim Weitermelden des Fehlers alle Nachrichten an, das Trace-Protokoll und eine Erläuterung, wie der Fehler reproduziert werden kann. Senden Sie nach Möglichkeit alls relevanten DADs, XML-Dokumente und Tabellendefinitionen mit.

DXXG007E Länderspezifische Angaben <*locale*> sind nicht konsistent mit der DB2-Codepage <*code_page*>.

Erläuterung: Die länderspezifischen Angaben zum Server-Betriebssystem sind nicht konsistent mit der DB2-Codepage.

Benutzeraktion: Korrigieren Sie die länderspezifischen Angaben zum Server-Betriebssystem, und starten Sie DB2 erneut.

DXXG008E Länderspezifische Angaben <*locale*> sind nicht unterstützt.

Erläuterung: Die länderspezifischen Angaben zum Server-Betriebssystem können in der Codepage-Tabelle nicht gefunden werden.

Benutzeraktion: Korrigieren Sie die länderspezifischen Angaben zum Server-Betriebssystem, und starten Sie DB2 erneut.

DXXQ000E <*Element*> fehlt in der DAD-Datei.

Erläuterung: Ein verbindliches Element ist in der DAD-Datei nicht vorhanden.

Benutzeraktion: Fügen Sie das fehlende Element der DAD-Datei hinzu.

DXXQ001E Ungültige SQL-Anweisung für XML-Generierung.

Erläuterung: Die SQL-Anweisung in der DAD-Datei oder der Datei, die sie überschreibt, ist nicht gültig. Zum Generieren von XML-Dokumenten ist eine SELECT-Anweisung erforderlich.

Benutzeraktion: Korrigieren Sie die SQL-Anweisung.

DXXQ002E Speicherbereich für XML-Dokumente kann nicht generiert werden.

Erläuterung: Das System hat nicht mehr genügend Platz im Speicher oder auf der Festplatte. Es ist kein Platz mehr für die resultierenden XML-Dokumente vorhanden.

Benutzeraktion: Begrenzen Sie die Anzahl der zu generierenden Dokumente. Verringern Sie die Größe der einzelnen Dokumente, indem Sie einige nicht erforderliche Element- und Attributknoten aus der Dokumentzugriffsdefinition (DAD-Datei) entfernen.

DXXQ003W Ergebnis übersteigt Maximum.

Erläuterung: Die benutzerdefinierte SQL-Abfrage generiert mehr XML-Dokumente als das angegebene Maximum. Es kann nur die angegebene Anzahl von Dokumenten zurückgegeben werden.

Benutzeraktion: Keine Aktion erforderlich. Wenn alle Dokumente erforderlich sind, geben Sie Null als maximale Anzahl von Dokumenten an.

DXXQ004E Die Spalte <*spaltenname*> ist nicht im Ergebnis der Abfrage.

Erläuterung: Die angegebene Spalte ist keine der Spalten im Ergebnis der SQL-Abfrage.

Benutzeraktion: Ändern Sie den angegebenen Spaltennamen in der Dokumentzugriffsdefinition (DAD-Datei), um sie als eine der Spalten im Ergebnis der SQL-Abfrage zu kennzeichnen. Alternativ dazu können Sie auch die SQL-Abfrage ändern, so daß sie die angegebene Spalte im Ergebnis enthält.

DXXQ004W DTD-ID nicht in DAD gefunden.

Erläuterung: In der DAD ist VALIDATION auf YES gesetzt, aber die DTD-ID wurde nicht angegeben. Es wird keine Gültigkeitsprüfung durchgeführt.

Benutzeraktion: Keine Aktion erforderlich. Wenn eine Gültigkeitsprüfung durchgeführt werden soll, geben Sie das DTD-ID-Element in der DAD-Datei an.

DXXQ005E Falsche relationale Zuordnung. Das Element <*elementname*> befindet sich auf einer niedrigeren Ebene als seine untergeordnete Spalte <*spaltenname*>.

Erläuterung: Die Zuordnung der SQL-Abfrage zu XML ist fehlerhaft.

Benutzeraktion: Stellen Sie sicher, daß die Spalten im Ergebnis der SQL-Abfrage von oben nach unten in der relationalen Hierarchie angeordnet sind. Stellen Sie außerdem, sicher, daß es für den Beginn jeder Stufe einen einspaltigen Kandidatenschlüssel gibt. Wenn in einer Tabelle kein solcher Schlüssel vorhanden ist, sollte die Abfrage mit einem Tabellenausdruck und der integrierten DB2-Funktion generate_unique() einen Schlüssel für diese Tabelle erstellen.

DXXQ006E Ein attribute_node-Element (attribute_node=Attributknoten) hat keinen Namen.

Erläuterung: Ein Element attribute_node in der DAD-Datei hat kein name-Attribut.

Benutzeraktion: Stellen Sie sicher, daß jeder attribute_node einen Namen in der DAD-Datei enthält.

DXXQ007E **attribute_node** <*attributname*> **(attribute_node=Attributknoten) verfügt über kein Spaltenelement bzw. keinen RDB_node (RDB-Knoten).**

Erläuterung: Das Element attribute_node in der DAD-Datei hat kein Spaltenelement oder keinen RDB_node.

Benutzeraktion: Stellen Sie sicher, daß jeder attribute_node ein Spaltenelement oder einen RDB_node in der DAD-Datei enthält.

DXXQ008E **Ein text_node-Element (text_node=Textknoten) verfügt über kein Spaltenelement.**

Erläuterung: Ein Element text_node in der DAD-Datei hat kein Spaltenelement.

Benutzeraktion: Stellen Sie sicher, daß jeder text_node ein Spaltenelement in der DAD-Datei enthält.

DXXQ009E **Ergebnistabelle** <*tabelle*> **ist nicht vorhanden.**

Erläuterung: Die angegebene Ergebnistabelle konnte im Systemkatalog nicht gefunden werden.

Benutzeraktion: Erstellen Sie die Ergebnistabelle, bevor Sie die gespeicherte Prozedur aufrufen.

DXXQ010E **RDB_node (RDB-Knoten) von** <*knotenname*> **verfügt über keine Tabelle in der DAD-Datei.**

Erläuterung: Der RDB_node des attribute_node oder text_node muß eine Tabelle haben.

Benutzeraktion: Geben Sie die Tabelle des RDB_node für attribute_node oder text_node in der DAD-Datei an.

DXXQ011E **RDB_node-Element von** <*knotenname*> **(RDB_node=RDB-Knoten) verfügt über keine Spalte in der DAD-Datei.**

Erläuterung: Der RDB_node des attribute_node oder text_node muß eine Spalte haben.

Benutzeraktion: Geben Sie die Spalte des RDB_node für attribute_node oder text_node in der DAD-Datei an.

DXXQ012E **In der DAD sind Fehler aufgetreten.**

Erläuterung: Der XML Extender konnte das erwartete Element bei der Verarbeitung der DAD nicht finden.

Benutzeraktion: Stellen Sie sicher, daß die DAD ein gültiges XML-Dokument ist und alle für die DAD-DTD erforderlichen Elemente enthält. Schlagen Sie in der Veröffentlichung zum XML Extender für die DAD-DTD nach.

DXXQ013E **Das Tabellen- oder Spaltenelement hat in der DAD-Datei keinen Namen.**

Erläuterung: Die Elementtabelle oder -spalte muß einen Namen in der DAD-Datei haben.

Benutzeraktion: Geben Sie den Namen eines Tabellen- oder Spaltenelements in der DAD an.

DXXQ014E **Ein element_node-Element (element_node=Elementknoten) hat keinen Namen.**

Erläuterung: Ein Element element_node in der DAD-Datei hat kein name-Attribut.

Benutzeraktion: Stellen Sie sicher, daß jedes Element element_node einen Namen in der DAD-Datei enthält.

DXXQ015E **Das Bedingungsformat ist ungültig.**

Erläuterung: Die Bedingung im condition-Element in der DAD-Datei hat ein ungültiges Format.

Benutzeraktion: Stellen Sie sicher, daß das Format der Bedingung gültig ist.

DXXQ016E Der Tabellenname in diesem RDB_node (RDB_node=RDB-Knoten) ist nicht im Anfangselement der DAD-Datei definiert.

Erläuterung: Alle Tabellen müssen im RDB_node des Anfangselements in der DAD-Datei definiert werden. Tabellen der Unterelemente müssen den im Anfangselement definierten Tabellen entsprechen. Der Tabellenname in diesem RDB_node ist im Anfangselement nicht enthalten.

Benutzeraktion: Stellen Sie sicher, daß die Tabelle des RDB-Knotens im Anfangselement der DAD-Datei definiert ist.

DXXQ017E Die Spalte in der Ergebnistabelle <*tabelle*> **ist zu klein.**

Erläuterung: Ein vom XML Extender generiertes XML-Dokument ist zu groß für die Spalte der Ergebnistabelle.

Benutzeraktion: Geben Sie die Ergebnistabelle frei. Erstellen Sie eine andere Ergebnistabelle mit einer größeren Spalte. Starten Sie die gespeicherte Prozedur erneut.

DXXQ018E In der SQL-Anweisung fehlt die Klausel ORDER BY.

Erläuterung: Die Klausel ORDER BY fehlt in der SQL-Anweisung in einer DAD-Datei, die eine Zuordnung zwischen SQL und XML herstellt.

Benutzeraktion: Editieren Sie die DAD-Datei. Fügen Sie eine Klausel ORDER BY hinzu, die die Spalten zur Kennzeichnung der Entitäten enthält.

DXXQ019E Das Element objids hat kein Spaltenelement in der DAD-Datei.

Erläuterung: Das objids-Element hat kein Spaltenelement in der DAD-Datei, die die Zuordnung zwischen SQL und XML herstellt.

Benutzeraktion: Editieren Sie die DAD-Datei. Fügen Sie die Schlüsselspalten als Unterelemente des Elements objids hinzu.

DXXQ020I XML erfolgreich generiert.

Erläuterung: Die angeforderten XML-Dokumente wurden erfolgreich aus der Datenbank generiert.

Benutzeraktion: Keine Aktion erforderlich.

DXXQ021E Tabelle <*tabellenname*> **verfügt nicht über die Spalte** <*spaltenname*>**.**

Erläuterung: Die Tabelle enthält nicht die in der Datenbank angegebene Spalte.

Benutzeraktion: Geben Sie in der DAD einen anderen Spaltennamen an, oder fügen Sie die angegebene Spalte in der Tabellendatenbank ein.

DXXQ022E Spalte <*spaltenname*> **von** <*tabellenname*> **sollte den Typ** <*typname*> **haben.**

Erläuterung: Der Typ der Spalte ist falsch.

Benutzeraktion: Korrigieren Sie den Typ der Spalte in der DAD-Datei.

DXXQ023E Spalte <*spaltenname*> **von** <*tabellenname*> **kann nicht länger als** <*länge*> **sein.**

Erläuterung: Die angegebene Länge für die Spalte in der DAD ist zu lang.

Benutzeraktion: Korrigieren Sie die Spaltenlänge in der Dokumentzugriffsdefinition (DAD).

DXXQ024E Tabelle <*tabellenname*> **kann nicht erstellt werden.**

Erläuterung: Die angegebene Tabelle kann nicht erstellt werden.

Benutzeraktion: Stellen Sie sicher, daß der Benutzer eine ausreichende Berechtigung zum Erstellen einer Tabelle in der Datenbank hat.

DXXQ025I Zerlegung von XML verlief erfolgreich.

Erläuterung: Ein XML-Dokument wurde erfolgreich zerlegt und in einer Objektgruppe gespeichert.

Benutzeraktion: Keine Aktion erforderlich.

DXXQ026E XML-Daten <*xml_name*> **sind zu groß für Spalte** <*spaltenname*>**.**

Erläuterung: Die angegebene Datenkomponente aus einem XML-Dokument ist zu groß für die angegebene Spalte.

Benutzeraktion: Vergrößern Sie die Spalte mit der Anweisung ALTER TABLE, oder verringern Sie die Größe der Daten, indem Sie das XML-Dokument editieren.

DXXQ028E Objektgruppe <*gruppenname*> **kann nicht in der Tabelle XML_USAGE gefunden werden.**

Erläuterung: Ein Datensatz für die Objektgruppe wurde in der Tabelle XML_USAGE nicht gefunden.

Benutzeraktion: Stellen Sie sicher, daß Sie die Objektgruppe aktiviert haben.

DXXQ029E DAD kann in der Tabelle XML_USAGE, Objektgruppe: <*objektgruppenname*> **nicht gefunden werden.**

Erläuterung: Ein DAD-Datensatz für die Objektgruppe wurde in der Tabelle XML_USAGE nicht gefunden.

Benutzeraktion: Stellen Sie sicher, daß Sie die Objektgruppe richtig aktiviert haben.

DXXQ030E Falsche XML-Überschreibung.

Erläuterung: Der Wert für XML_override value wurde in der gespeicherten Prozedur falsch angegeben.

Benutzeraktion: Stellen Sie sicher, daß die Syntax von XML_override korrekt ist.

DXXQ031E Tabellenname kann nicht länger als die in DB2 zulässige Höchstlänge sein.

Erläuterung: Der im Bedingungselement in der DAD angegebene Tabellenname ist zu lang.

Benutzeraktion: Korrigieren Sie die Länge des Tabellennamens in der Dokumentzugriffsdefinition (DAD).

DXXQ032E Spaltenname kann nicht länger als die in DB2 zulässige Höchstlänge sein.

Erläuterung: Der im Bedingungselement in der DAD angegebene Spaltenname ist zu lang.

Benutzeraktion: Korrigieren Sie die Länge des Spaltennamens in der Dokumentzugriffsdefinition (DAD).

DXXQ033E Bei <*kennung*> *beginnt eine ungültige Kennung*

Erläuterung: Die Zeichenfolge ist keine gültige DB2 SQL-Kennung.

Benutzeraktion: Korrigieren Sie die Zeichenfolge in der DAD, so daß sie den Regeln für DB2 SQL-Kennungen entspricht.

DXXQ034E Ungültiges Bedingungselement im Angangs-RDB_node (RDB-Knoten) der DAD: <*bedingung*>

Erläuterung: Das Bedingungselement muß eine gültige WHERE-Klausel aus Verknüpungsbedingungen, die über die Verknüpfung AND verbunden sind, sein.

Benutzeraktion: Siehe die Dokumentation zum XML Extender für die richtige Syntax der Verknüpfungsbedingung in einer DAD.

DXXQ035E Ungültige Verknüpfungsbedingung im Angangs-RDB_node (RDB-Knoten) der DAD: <*bedingung*>

Erläuterung: Spaltennamen in dem Bedingungselement des Anfangs-RDB_node müssen mit dem

Tabellennamen angegeben werden, wenn die DAD mehrere Tabellen angibt.

Benutzeraktion: Siehe die Dokumentation zum XML Extender für die richtige Syntax der Verknüpfungsbedingung in einer DAD.

DXXQ036E Ein in einem DAD-Bedingungsbefehl angegebener Schemaname ist länger als zulässig.

Erläuterung: Bei der Syntaxanalyse wurde ein Fehler in einem Bedingungsbefehl in der DAD festgestellt. Der Bedingungstext enthält eine ID, die durch einen zu langen Schemanamen gekennzeichnet ist.

Benutzeraktion: Korrigieren Sie den Text der Bedingungsbefehle in der Dokumentzugriffsdefinition (DAD).

DXXQ037E *<element>* **kann nicht mit mehrfachem Vorkommen generiert werden.**

Erläuterung: Der Elementknoten und seine untergeordneten Elemente haben keine Zuordnung in der Datenbank, aber das Attribut multi_occurrence ist auf YES gesetzt.

Benutzeraktion: Korrigieren Sie die DAD, indem Sie entweder das Attribut multi_occurrence auf NO setzen, oder erstellen Sie einen RDB_node in einem seiner abhängigen Elemente.

Diagnose-Tracing

Der XML Extender umfaßt eine Trace-Einrichtung, die die Aktivitäten des XML Extender-Server aufzeichnet. Verwenden Sie die Trace-Einrichtung nur, wenn Sie von der IBM Softwareunterstützung dazu aufgefordert werden.

Die Trace-Einrichtung zeichnet in einer Server-Datei Informationen über eine Vielzahl von Ereignissen auf, z. B. den Aufruf oder das Beenden einer XML Extender-Komponente oder die Rückgabe eines Fehlercodes von einer XML Extender-Komponente. Da Informationen für zahlreiche Ereignisse aufgezeichnet werden, sollten Sie diese Trace-Einrichtung nur bei Bedarf verwenden, etwa beim Überprüfen von Fehlerbedingungen. Darüber hinaus sollten Sie bei Verwendung der Trace-Einrichtung die Anzahl der aktiven Anwendungen begrenzen. Die Begrenzung der Anzahl aktiver Anwendungen kann die Isolierung der Problemursache erleichtern.

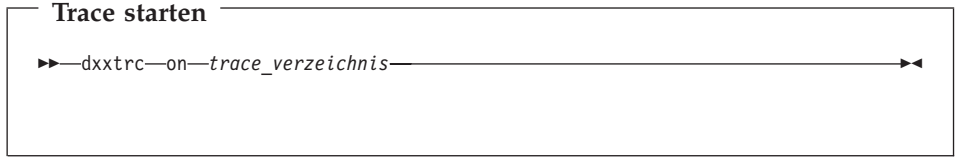
Verwenden Sie den Befehl **dxxtrc** zum Starten oder Stoppen des Trace. Sie können den Befehl über eine Befehlszeile auf einem AIX-, Windows NT- oder Solaris-Server eingeben. Zur Eingabe des Befehls ist eine der Berechtigungen SYSADM, SYSCTRL oder SYSMINT erforderlich.

Trace starten

Zweck

Zeichnet die Aktivitäten des XML Extender-Server auf. Verwenden Sie zum Starten des Trace die Option `on` mit **dxxtrc**, zusammen mit dem Namen des Verzeichnisses, das die Trace-Datei enthält. Wenn der Trace eingeschaltet ist, wird die Datei `dxx`*`INSTANCE`*`.trc`, in dem angegebenen Verzeichnis abgelegt. *INSTANCE* ist der Wert von DB2INSTANCE. Jedes DB2-Exemplar hat seine eigene Protokolldatei.

Format



Parameter

Tabelle 54. Trace-Parameter

Parameter	Beschreibung
trace_verzeichnis	Name des Verzeichnisses, in dem `dxx`*`INSTANCE`*`.trc` abgelegt wird. Erforderlich, kein Standardwert.

Beispiele

Das folgende Beispiel zeigt, wie der Trace für ein Exemplar db2inst1 unter AIX gestartet wird. Die Trace-Datei `dxxdb2inst1.trc`, wird im Verzeichnis `/home/db2inst1/sqllib/log` abgelegt.

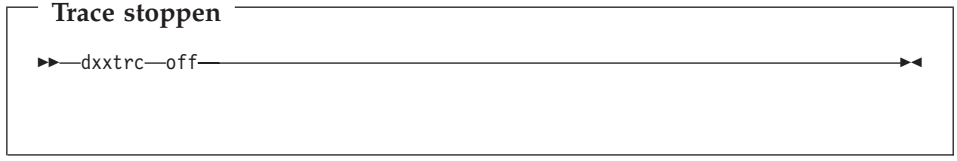
```
dxxtrc on /home/db2inst1/sqllib/log
```

Trace stoppen

Zweck

Schaltet den Trace aus. Es werden keine Trace-Informationen protokolliert. Da das Aktivieren des Trace die Leistung beeinträchtigt, wird empfohlen, in einer Fertigungsumgebung den Trace auszuschalten.

Format



Beispiele

Dieses Beispiel zeigt, daß die Trace-Einrichtung ausgeschaltet ist.

```
dxxtrc off
```

Teil 5. Anhänge und Schlussteil

Anhang A. DTD für die DAD-Datei

Dieser Abschnitt beschreibt die Dokumenttypdeklarationen (DTD) für die DAD-Datei. Die DAD-Datei selbst ist ein mit einer Baumstruktur definiertes XML-Dokument; sie erfordert eine DTD. Der Name der DTD-Datei lautet `dxxdad.dtd`. Abb. 13 auf Seite 278 zeigt die DTD für die DAD-Datei. Die Elemente dieser Datei sind im Anschluß an die Abbildung beschrieben.

```
<?xml encoding="US-ASCII"?>
  <!ELEMENT DAD (dtdid?, validation, (Xcolumn | Xcollection))>
  <!ELEMENT dtdid (#PCDATA)>
  <!ELEMENT validation (#PCDATA)>
  <!ELEMENT Xcolumn (table*)>
  <!ELEMENT table (column*)>
  <!ATTLIST table name CDATA #REQUIRED
                              key CDATA #IMPLIED
                              orderBy CDATA #IMPLIED>
  <!ELEMENT column EMPTY>
  <!ATTLIST column
                      name CDATA #REQUIRED
                      type CDATA #IMPLIED
                      path CDATA #IMPLIED
                      multi_occurrence CDATA #IMPLIED>
  <!ELEMENT Xcollection (SQL_stmt?, objids?, prolog, doctype, root_node)>
  <!ELEMENT SQL_stmt (#PCDATA)>
  <!ELEMENT objids (column+)>
  <!ELEMENT prolog (#PCDATA)>
  <!ELEMENT doctype (#PCDATA | RDB_node)*>
  <!ELEMENT root_node (element_node)>
  <!ELEMENT element_node (RDB_node*,
                                    attribute_node*,
                                    text_node?,
                                    element_node*,
                                    namespace_node*,
                                    process_instruction_node*,
                                    comment_node*)>
  <!ATTLIST element_node
                     name CDATA #REQUIRED
                     ID CDATA #IMPLIED
                     multi_occurrence CDATA "NO"
                     BASE_URI CDATA #IMPLIED>
  <!ELEMENT attribute_node (column | RDB_node)>
  <!ATTLIST attribute_node
                     name CDATA #REQUIRED>
  <!ELEMENT text_node (column | RDB_node)>
  <!ELEMENT RDB_node (table+, column?, condition?)>
  <!ELEMENT condition (#PCDATA)>
  <!ELEMENT comment_node (#PCDATA)>
  <!ELEMENT namespace_node (EMPTY)>
  <!ATTLIST namespace_node
                     name CDATA #IMPLIED
                     value CDATA #IMPLIED>
  <!ELEMENT process_instruction_node (#PCDATA)>
```

Abbildung 13. Die DTD für die Dokumentzugriffsdefinition (DAD)

Die DAD-Datei umfaßt vier wichtige Elemente:

- DTD-ID
- Gültigkeitsprüfung
- Xcolumn
- Xcollection

Xcolumn und Xcollection haben untergeordnete Elemente und Attribute, die die Zuordnung von XML-Daten zu relationalen Tabellen in DB2 erleichtern. Die folgende Liste beschreibt die wichtigsten Elemente und ihre untergeordneten Elemente und Attribute. Die Syntaxbeispiele stammen aus Abb. 13 auf Seite 278.

DTD-ID-Element

Gibt die ID der in der Tabelle DTD_REF gespeicherten DTD an. `DTD-ID` verweist auf die DTD, die die Gültigeit der XML-Dokumente prüft oder die Zuordnung zwischen XML-Objektgruppentabellen und XML-Tabellen steuert. `DTD-ID` muß für XML-Objektgruppen angegeben werden. Für XML-Spalten ist diese Angabe wahlfrei und nur erforderlich, wenn Sie Seitentabellen zum Indexieren von Elementen oder Attributen erstellen oder die Gültigkeit von XML-Eingabedokumenten prüfen wollen. `DTD-ID` muß mit der SYSTEM-ID übereinstimmen, die im `doctype` der XML-Dokumente angegeben wurde.

Syntax: `<!ELEMENT dtdid (#PCDATA)>`

validation-Element

Gibt an, ob die Gültigkeit des XML-Dokuments mit der DTD für die DAD geprüft werden soll. Bei Angabe von YES muß die `DTD-ID` ebenfalls angegeben werden.

Syntax: `<!ELEMENT validation(#PCDATA)>`

Xcolumn-Element

Definiert das Indexierungsschema für eine XML-Spalte. Dieses Element umfaßt Null oder mehr Tabellen.

Syntax: `<!ELEMENT Xcolumn (table*)>Xcolumn` hat ein untergeordnetes Element, `table`.

table-Element

Definiert ein oder mehrere relationale Tabellen, die für die Indexierung von Elementen oder Attributen der in einer XML-Spalte gespeicherten XML-Dokumente erstellt wurden.

Syntax:

```
<!ELEMENT table (column+)>
 <!ATTLIST table name CDATA #REQUIRED
key CDATA #IMPLIED
orderBy CDATA #IMPLIED>
```

Das Element `table` hat ein Attribut:

name-Attribut

Gibt den Namen der Seitentabelle an.

Das Element `table` hat ein untergeordnetes Element:

key-Attribut
Der singuläre Primärschlüssel der Tabelle

orderBy-Attribut
Die Namen der Spalten, die die Reihenfolge der mehrfach vorkommenden Elementtexte oder Attributwerte beim Generieren von XML-Dokumenten angibt.

column-Element
Gibt die Spalte der Tabelle an, die den Wert eines Standortpfads des angegebenen Typs enthält.

Syntax:

```
<!ATTLIST column
                    name CDATA #REQUIRED
                    type  CDATA #IMPLIED
                    path  CDATA #IMPLIED
                    multi_occurrence CDATA #IMPLIED>
```

Das Element `column` hat die folgenden Attribute:

name-Attribut
Gibt den Namen der Spalte an. Dies ist der Aliasname des Standortpfads, der ein Element oder ein Attribut angibt.

type-Attribut
Definiert den Datentyp der Spalte. Dies kann ein beliebiger SQL-Datentyp sein.

path-Attribut
Zeigt den Standortpfad eines XML-Elements oder -Attributs an; muß der einfache Standortpfad sein, wie in Tabelle 3.1.a.

multi_occurrence-Attribut
Gibt an, ob dieses Element bzw. Attribut in einem Dokument mehrfach auftreten kann. Gültige Werte sind YES oder NO.

Xcollection
Definiert die Zuordnung zwischen XML-Dokumenten und einer XML-Objektgruppe relationaler Tabellen.

Syntax: `<!ELEMENT Xcollection(SQL_stmt*, prolog, doctype, root_node)>Xcollection` hat die folgenden untergeordneten Elemente:

SQL_stmt
Gibt die SQL-Anweisung an, die der XML Extender zur Definition der Objektgruppe verwendet. Die Anweisung wählt insbesondere XML-Daten aus den XML-Objektgruppentabellen aus und verwendet die Daten zum Generieren der XML-Dokumente in der Objektgruppe. Der Wert dieses Elements muß eine gültige SQL-Anweisung sein. Dieser Angabe wird

nur zum Zusammensetzen verwendet, und es ist nur eine einzige SQL_stmt zulässig. Zum Zerlegen können für SQL_stmt mehrere Werte angegeben werden, um das erforderliche Erstellen und Einfügen der Tabelle durchzuführen.

Syntax: `<!ELEMENT SQL_stmt  #PCDATA >`

prolog Der Text für den XML-Prolog. Es wird für alle Dokumente in der gesamten Objektgruppe derselbe Prolog verwendet. Der Wert von prolog ist festgelegt.

Syntax: `<!ELEMENT prolog  #PCDATA>`

doctype
Definiert den Text für die Dokumenttypdefinition des XML-Dokuments.

Syntax: `<!ELEMENT doctype  #PCDATA | RDB_node>doctype` kann auf eine der folgenden Arten angegeben werden:

- Definieren eines expliziten Werts. Dieser Wert wird für alle Dokumente in der gesamten Objektgruppe verwendet.
- Geben Sie beim Zerlegen das untergeordnete Element RDB_node an, das den Spaltendaten einer Tabelle zugeordnet und auch so gespeichert werden kann.

doctype hat ein untergeordnetes Element:

RDB_node
Noch nicht implementiert.

root_node
(Stammknoten) Definiert den virtuellen Root-Knoten. root_node muß ein untergeordnetes Element element_node haben, das nur ein Mal verwendet werden kann. Der element_node unter dem root_node ist der root_node des XML-Dokuments.

Syntax: `<!ELEMENT root_node(element_node)>`

element_node
(Elementknoten) Steht für ein XML-Element. Muß in der für die Objektgruppe angegebenen DTD definiert sein. Für die RDB_node-Zuordnung muß der Root-element_node einen RDB_node haben, der alle Tabellen mit XML-Daten und alle untergeordneten Knoten angibt.

Er kann Null oder mehr attribute_nodes und untergeordnete element_nodes sowie Null oder einen text_node haben. Für andere Element als das Root-Element ist kein RDB_node erforderlich.

Syntax:

Ein `element_node` ist durch die folgenden untergeordneten Elemente definiert:

RDB_node
(Wahlfrei) Gibt Tabellen, Spalten und Bedingungen für XML-Daten an. Der RDB_node für ein Element muß nur für die RDB_node-Zuordnung definiert werden. In diesem Fall müssen eine oder mehrere Tabellen angegeben werden. Die Spalte ist nicht erforderlich, da der Elementinhalt durch seinen text_node angegeben ist. Die Bedingung ist wahlfrei, je nach der DTD und der Abfragebedingung.

child nodes
(Wahlfrei) Ein `element_node` kann auch die folgenden untergeordneten Knoten haben:

element_node
(Elementknoten) Steht für untergeordnete Elemente des aktuellen XML-Elements

attribute_node
(Attributknoten) Steht für Attribute des aktuellen XML-Elements

text_node
(Textknoten) Steht für den CDATA-Text des aktuellen XML-Elements

attribute_node
(Attributknoten) Steht für ein XML-Attribut. Dies ist der Knoten, der die Zuordnung zwischen einem XML-Attribut und den Spaltendaten in einer relationalen Tabelle definiert.

Syntax:

Der `attribute_node` muß Definitionen für ein `name`-Attribut haben sowie ein untergeordnetes Element `column` oder `RDB_node`. `attribute_node` hat die folgenden Attribute:

name Der Name des Attributs.

`attribute_node` hat die folgenden untergeordneten Elemente:

Column
Für die SQL-Zuordnung verwendet. Die Spalte muß in der SELECT-Klausel von `SQL_stmt` angegeben werden.

RDB_node

Für die RDB_node-Zuordnung verwendet. Der Knoten definiert die Zuordnung zwischen diesem Attribut und den Spaltendaten in der relationalen Tabelle. Die Tabelle und die Spalte müssen angegeben werden. Die Bedingung ist wahlfrei.

text_node

(Textknoten) Steht für den Textinhalt eines XML-Elements. Dies ist der Knoten, der die Zuordnung zwischen einem XML-Elementinhalt und den Spaltendaten in einer relationalen Tabelle definiert.

Syntax: Er muß über ein untergeordnetes Element `column` oder `RDB_node` definiert werden:

Column

Für die SQL-Zuordnung erforderlich. In diesem Fall muß die Spalte in der SELECT-Klausel von `SQL_stmt` angegeben sein.

RDB_node

Für die RDB_node-Zuordnung erforderlich. Der Knoten definiert die Zuordnung zwischen diesem Textinhalt und den Spaltendaten in der relationalen Tabelle. Die `Tabelle` und die `Spalte` müssen angegeben werden. Die Bedingung ist wahlfrei.

Anhang B. Beispiele

Dieser Anhang zeigt die mit den Beispielen in diesem Handbuch verwendeten Objekte.

XML DTD

Die folgende DTD wird für das Dokument `getstart.xml` verwendet, auf das in diesem Handbuch verwiesen wird und das in Abb. 15 auf Seite 286 dargestellt wird.

```
<!xml encoding="US-ASCII"?>

<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
```

Abbildung 14. Beispiel XML DTD: getstart.dtd

XML-Dokument: getstart.xml

Das folgende XML-Dokument `getstart.xml` ist das in den Beispielen in diesem Handbuch verwendete Beispiel-XML-Dokument. Es enthält XML-Befehle zur Bildung einer Bestellung.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd">
<Order key="1">
  <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>
  <Part color="black ">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
  </Part>
  <Part color="red   ">
    <key>128</key>
    <Quantity>28</Quantity>
    <ExtendedPrice>38000.00</ExtendedPrice>
    <Tax>7.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-12-30</ShipDate>
      <ShipMode>TRUCK </ShipMode>
    </Shipment>
  </Part>
</Order>
```

Abbildung 15. Beispiel-XML-Dokument: `getstart.xml`

Dokumentzugriffsdefinitionsdateien

Die folgenden Abschnitte enthalten Dokumentzugriffsdefinitionsdateien (″Document Access Definition, DAD), die über XML-Spalten- oder XML-Objektgruppenzugriffsmodi eine Zuordnung zwischen XML-Daten und relationalen DB2-Tabellen herstellen.

- „DAD-Datei: XML-Spalte” auf Seite 287
- „DAD-Datei: XML-Objektgruppe - SQL-Zuordnung” auf Seite 288 zeigt eine DAD-Datei für eine XML-Objektgruppe mit SQL-Zuordnung.
- „DAD-Datei: XML - RDB_node-Zuordnung” auf Seite 290 zeigt eine DAD-Datei für eine XML-Objektgruppe mit RDB_node-Zuordnung.

DAD-Datei: XML-Spalte

Diese DAD-Datei enthält die Zuordnung für eine XML-Spalte, die die Tabelle, die Seitentabellen und Spalten definiert, die die XML-Daten enthalten sollen.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>

<Xcolumn>
     <table name="order_side_tab">
         <column name="order_key"
            type="integer"
            path="/Order/@key"
            multi_occurrence="NO"/>
         <column name="customer"
            type="varchar(50)"
            path="/Order/Customer/Name"
            multi_occurrence="NO"/>
</table>
     <table name="part_side_tab">
         <column name="price"
            type="decimal(10,2)"
            path="/Order/Part/ExtendedPrice"
            multi_occurrence="YES"/>
</table>
     <table name="ship_side_tab">
         <column name="date"
            type="DATE"
            path="/Order/Part/Shipment/ShipDate"
            multi_occurrence="YES"/>
</table>

</Xcolumn>

</DAD>
```

Abbildung 16. Beispiel-DAD-Datei für eine XML-Spalte

DAD-Datei: XML-Objektgruppe - SQL-Zuordnung

Diese DAD-Datei enthält eine SQL-Anweisung, die die DB2-Tabellen, Spalten und Bedingungen enthält, die die XML-Daten enthalten sollen.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO</validation>
<Xcollection>
<SQL_stmt>SELECT o.order_key, customer_name, customer_email, p.part_key, color,
   quantity, price, tax, ship_id, date, mode from order_tab o, part_tab p,
   table (select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab) s
         WHERE o.order_key = 1 and
               p.price > 20000 and
               p.order_key = o.order_key and
               s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id</SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

Abbildung 17. Beispiel-DAD-Datei für eine XML-Objektgruppe mit SQL-Zuordnung (Teile- 1 von 2)

```
<root_node>
<element_node name="Order">
  <attribute_node name="key">
    <column name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node><column name="customer_name"/></text_node>
    </element_node>
    <element_node name="Email">
      <text_node><column name="customer_email"/></text_node>
    </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
      <column name="color"/>
    </attribute_node>
    <element_node name="key">
      <text_node><column name="part_key"/></text_node>
    </element_node>
    <element_node name="Quantity">
      <text_node><column name="quantity"/></text_node>
    </element_node>
    <element_node name="ExtendedPrice">
      <text_node><column name="price"/></text_node>
    </element_node>
    <element_node name="Tax">
      <text_node><column name="tax"/></text_node>
    </element_node>
    <element_node name="Shipment" multi_occurence="YES">
      <element_node name="ShipDate">
        <text_node><column name="date"/></text_node>
      </element_node>
      <element_node name="ShipMode">
        <text_node><column name="mode"/></text_node>
      </element_node>
    </element_node>
  </element_node>
</element_node>
</root_node>
</Xcollection>
</DAD>
```

Abbildung 17. Beispiel-DAD-Datei für eine XML-Objektgruppe mit SQL-Zuordnung (Teile- 2 von 2)

DAD-Datei: XML - RDB_node-Zuordnung

Diese DAD-Datei verwendet <RDB_node>-Elemente zum Definieren der DB2-Tabellen, Spalten und Bedingungen, die die XML-Daten enthalten sollen.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>
<Xcollection>
      <prolog>?xml version="1.0"?</prolog>
      <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
      <root_node>
         <element_node name="Order">
           <RDB_node>
              <table name="order_tab"/>
              <table name="part_tab"/>
              <table name="ship_tab"/>
              <condition>
                order_tab.order_key = part_tab.order_key AND
                part_tab.part_key = ship_tab.part_key
              </condition>
           </RDB_node>
           <attribute_node name="key">
              <RDB_node>
                 <table name="order_tab"/>
                 <column name="order_key"/>
              </RDB_node>
           </attribute_node>
           <element_node name="Customer">
              <text_node>
                 <RDB_node>
                    <table name="order_tab"/>
                    <column name="customer"/>
                 </RDB_node>
              </text_node>
           </element_node>
           <element_node name="Part">
              <RDB_node>
                 <table name="part_tab"/>
                 <table name="ship_tab"/>
                 <condition>
                    part_tab.part_key = ship_tab.part_key
                 </condition>
```

Abbildung 18. Beispiel-DAD-Datei für eine XML-Objektgruppe mit RDB_node-Zuordnung (Teile- 1 von 3)

```
   </RDB_node>
   <attribute_node name="key">
      <RDB_node>
         <table name="part_tab"/>
         <column name="part_key"/>
      </RDB_node>
   </attribute_node>
<element_node name="Quantity">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
         <column name="quantity"/>
      </RDB_node>
   </text_node>
</element_node>
<element_node name="ExtendedPrice">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
            <column name="price"/>
            <condition>
               price > 2500.00
            </condition>
      </RDB_node>
   </text_node>
</element_node>
<element_node name="Tax">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
         <column name="tax"/>
      </RDB_node>
   </text_node>
</element_node>
```

Abbildung 18. Beispiel-DAD-Datei für eine XML-Objektgruppe mit RDB_node-Zuordnung (Teile- 2 von 3)

```
            <element_node name="shipment">
               <RDB_node>
                  <table name="ship_tab"/>
                        <condition>
                     part_key = part_tab.part_key
                  </condition>
               </RDB_node>
               <element_node name="ShipDate">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="date"/>
                        <condition>
                           date > "1966-01-01"
                        </condition>
                     </RDB_node>
                  </text_node>
               </element_node>
               <element_node name="ShipMode">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="mode"/>
                     </RDB_node>
                  </text_node>
               </element_node>
               <element_node name="Comment">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="comment"/>
                     </RDB_node>
                  </text_node>
               </element_node>
             </element_node> <! -- end of element Shipment>
           </element_node> <! -- end of element Part --->
         </element_node> <! -- end of element Order --->
       </root_node>
  </Xcollection>
</DAD>
```

Abbildung 18. Beispiel-DAD-Datei für eine XML-Objektgruppe mit RDB_node-Zuordnung (Teile- 3 von 3)

Anhang C. Überlegungen zur Codepage

Es muß unbedingt sichergestellt werden, daß XML-Dokumente und andere zugehörige Dateien für jeden Client oder Server, der auf die Dateien zugreift, richtig codiert werden. Es daher wichtig, daß Sie verstehen, von welchen Annahmen der XML Extender beim Verarbeiten der Dateien ausgeht und wie er die Konvertierung der Codepages verarbeitet. Die wichtigsten Punkte dabei sind:

- Sicherstellen, daß die tatsächliche Codepage des Clients, der ein XML-Dokument von DB2 abruft, mit der Codierung des Dokuments übereinstimmt.
- Sicherstellen, daß bei der Verarbeitung des Dokuments durch einen XML-Parser die Codierungsdeklaration des XML-Dokuments ebenfalls mit der Codierung konsistent ist.

In den folgenden Abschnitten werden verschiedene Punkte zu diesen Überlegungen beschrieben, wie Sie mit eventuell auftretenden Problemen umgehen können und wie der XML Extender und DB2 Codepages unterstützen, wenn Dokumente vom Client an den Server und an die Datenbank übergeben werden.

Terminologie

In diesem Abschnitt werden die folgenden Fachbegriffe verwendet:

Dokumentcodierung
Die tatsächliche Codepage eines XML-Dokuments.

Dokumentcodierungsdeklaration
Der Name der in der XML-Deklaration angegebenen Codepage. Beispiel:

```
<?xml version="1.0" encoding="ibm037"?>
```

Konsistentes Dokument
Ein Dokument, in dem die Dokumentcodierung der in der Dokumentcodierungsdeklaration angegebenen Codepage entspricht.

Inkonsistentes Dokument
Ein Dokument, in dem die Dokumentcodierung nicht mit der in der Dokumentcodierungsdeklaration angegebenen Codepage übereinstimmt.

DB2CODEPAGE-Registrierungsdatenbank (Umgebungsvariable)
Gibt die Codepage der Daten an, die DB2 von einer Datenbank-Client-Anwendung dargestellt werden. DB2 ruft die Client-Codepage aus der

Ländereinstellung des Client-Betriebssystems ab, wenn diese Variable nicht gesetzt ist. Für DB2 überschreibt dieser Wert die Ländereinstellung des Client-Betriebssystems, sofern die Variable gesetzt ist.

Client-Codepage
Die Anwendungs-Codepage. Wenn die Variable DB2CODEPAGE gesetzt ist, entspricht die Client-Codepage dem Wert von DB2CODEPAGE. Andernfalls entspricht die Client-Codepage der Ländereinstellung des Client-Betriebssystems.

Server-Codepage der Codepage der Ländereinstellung des Server-Betriebssystems
Die Ländereinstellung des Betriebssystems, auf dem die DB2-Datenbank installiert ist.

Datenbank-Codepage:
Die Codierung der gespeicherten Daten; wird beim Erstellen der Datenbank angegeben. Sofern er nicht in der Klausel USING CODESET explizit angegeben ist, gilt für diesen Wert der Wert der Ländereinstellung des Server-Betriebssystems als Standardwert.

Annahmen zur DB2- und XML Extender-Codepage

Wenn DB2 ein XML-Dokument empfängt oder sendet, wird die Codierungsdeklaration nicht geprüft. Statt dessen wird die Codepage für den Client geprüft, um festzustellen, ob sie der Codepage der Datenbank entspricht. Falls die Codepages unterschiedlich sind, setzt DB2 die Daten in dem XML-Dokument um und nimmt folgende Anpassungen vor:

- Anpassung der Codepage der Datenbank beim Importieren des Dokuments oder eines Teils davon in eine Datenbanktabelle
- Anpassung der Codepage der Datenbank beim Zerlegen eines Dokuments in eine oder mehrere Datenbanktabellen
- Anpassung der Codepage des Clients beim Exportieren des Dokuments aus der Datenbank und beim Darstellen des Dokuments für den Client
- Anpassung der Codepage des Servers beim Verarbeiten einer Datei mit einer UDF, die Daten in eine Datei im Dateisystem des Servers zurückgibt.

Beim Importieren eines XML-Dokuments in die Datenbank wird es normalerweise als XML-Dokument importiert, um in einer XML-Spalte abgelegt zu werden, oder zum Zerlegen für eine XML-Objektgruppe; hierbei werden Element- und Attributinhalt als DB2-Daten gespeichert. Beim Importieren eines Dokuments konvertiert DB2 die Dokumentcodierung in die Codierung der Datenbank. DB2 geht davon aus, daß das Dokument die Codepage verwendet, die in der Spalte „Ausgangs-Codepage" in der Tabelle unten angegeben ist. In Tabelle 55 auf Seite 295 sind die Konvertierungen zusammengefaßt, die DB2 beim Importieren eines XML-Dokuments vornimmt.

Tabelle 55. UDFs und gespeicherte Prozeduren verwenden, wenn die XML-Datei in die Datenbank importiert wird

Verarbeitungsmethode	Ausgangs-Codepage für die Konvertierung	Ziel-Codepage für die Konvertierung	Kommentar
DTD-Datei in DTD_REF-Tabelle einfügen	Client-Codepage	Datenbank-Codepage	
Spalte aktivieren oder Objektgruppe der gespeicherten Prozedur aktivieren, oder Verwaltungsbefehle, die eine DAD-Datei importieren	Client-Codepage	Datenbank-Codepage	
UDFs: • XMLVarcharFromFile() • XMLCLOBFromFile() • Content(): Abrufen von XMLFILE in ein CLOB	Server-Codepage	Datenbank-Codepage	
Gespeicherte Prozeduren zum Zerlegen	Client-Codepage	Datenbank-Codepage	Es wird davon ausgegangen, daß das zu zerlegende Dokument die Client-Codepage verwendet. Daten aus der Zerlegung werden in Tabellen in der Datenbank-Codepage gespeichert.

Wenn ein XML-Dokument aus der Datenbank exportiert wird, dann meistens aufgrund einer Client-Anforderung, ein Dokument darzustellen, aufgrund einer Anforderung nach dem Inhalt eines XML-Dokuments oder beim Zerlegen eines Dokuments aus DB2-Daten. Beim Exportieren eines Dokuments konvertiert DB2 die Dokumentcodierung in die Codierung des Clients oder Servers, je nachdem, wo die Anforderung generiert wurde und wo die Daten dargestellt werden sollen. In Tabelle 56 auf Seite 296 sind die Konvertierungen zusammengefaßt, die DB2 beim Exportieren eines XML-Dokuments vornimmt.

Tabelle 56. UDFs und gespeicherte Prozeduren verwenden, wenn die XML-Datei aus der Datenbank exportiert wird

Verarbeitungsmethode	Konvertierung	Kommentar
UDF • XMLFileFromVarchar() • XMLFileFromCLOB()	Datenbank-Codepage in Client-Codepage, wenn die Daten dem Client dargestellt werden	
UDF • Content(): Abrufen von XMLVARCHAR in eine externe Server-Datei	Datenbank-Codepage in Server-Codepage	
Gespeicherte Prozedur zum Zusammensetzen: resultierende Tabellen werden in der Ergebnistabelle gespeichert, die abgefragt und exportiert werden kann.	Datenbank-Codepage in Client-Codepage, wenn die Daten dem Client dargestellt werden	Beim Zusammensetzen von Dokumenten kopiert der XML Extender die von dem Befehl in der DAD angegebene Codierungsdeklaration in das neu erstellte Dokument. Sie sollte bei der Darstellung mit der Client-Codepage übereinstimmen.

Überlegungen zur Codierungsdeklaration

Die Codierungsdeklaration deklariert die Codierung des XML-Dokuments; sie erscheint in der XML-Deklarationsanweisung. Beim Verwenden des XML Extender muß unbedingt sichergestellt werden, daß die Codierung des Dokuments der Codepage des Clients bzw. Servers entspricht, je nachdem, wo sich die Datei befindet.

Gültige Codierungsdeklarationen

Sie können innerhalb bestimmter Richtlinien jede beliebige Codierungsdeklaration in XML-Dokumenten verwenden. In diesem Abschnitt sind diese Richtlinien beschrieben, zusammen mit den unterstützten Codierungsdeklarationen.

Die empfohlenen portierbaren Codierungen für XML-Daten sind UTF-8 und UTF-16, entsprechend der XML-Spezifikation. Ihre Anwendung kann über verschiedene Unternehmen hinweg verwendet werden, wenn Sie diese Codierungen verwenden. Sofern Sie die in Tabelle 57 auf Seite 297 aufgelisteten Codierungen verwenden, ist Ihre Anwendung wahrscheinlich über verschiedene IBM Betriebssysteme hinweg portierbar. Verwenden Sie andere Codierungen, ist die Portierbarkeit Ihrer Daten weniger wahrscheinlich.

Für alle Betriebssysteme werden die folgenden Codierungsdeklarationen unterstützt. Die folgende Liste beschreibt die Bedeutung der einzelnen Spalten:

- **Codierung** gibt die in der XML-Deklaration zu verwendende Codierungszeichenfolge an.
- **OS** gibt das Betriebssystem an, unter dem DB2 die angegebene Codepage unterstützt.
- **Codepage** zeigt die von IBM definierte Codepage zu der angegebenen Codierung an

Tabelle 57. Von XML Extender unterstützte Codierungsdeklarationen

Kategorie	Codierung	OS	Codepage
Unicode	UTF-8	AIX, SUN, Linux	1208
	UTF-16	AIX, SUN, Linux	1200
ASCII	iso-8859-1	AIX, Linux, Sun	819
	ibm-1252	Windows NT	1252
	iso-8859-2	AIX, Linux, Sun	912
	iso-8859-5	AIX, Linux	915
	iso-8859-6	AIX	1089
	iso-8859-7	AIX, Linux	813
	iso-8859-8	AIX, Linux	916
	iso-8859-9	AIX, Linux	920
MBCS	gb2312	Windows NT	1386
	ibm-932, shift_jis78	AIX	932
	Shift_JIS	Nur AIX 4.3, Windows NT	943
	IBM-eucCN	AIX, Sun	1383
	IBM-eucJP, EUC-JP	AIX, Linux, Sun	954, 33722
	euc-tw, IBM-eucTW	AIX, Sun	964
	euc-kr, IBM-eucKR	AIX	970
	big5	AIX, Sun, Windows NT	950

Die Codierungszeichenfolge muß mit der Codepage des Dokumentziels kompatibel sein. Wenn ein Dokument von einem Server an einen Client zurückgegeben wird, muß seine Codierungszeichenfolge mit der Client-Codepage kompatibel sein.

Im Abschnitt „Konsistente Codierungen und Codierungsdeklarationen" sind die Folgen inkompatibler Codierungen beschrieben. Unter der folgenden URL finden Sie eine Liste der Codepages, die von dem von XML Extender verwendeten XML-Parser unterstützt werden:

`http://www.ibm.com/software/data/db2/extenders/xmlext/moreinfo/encoding.html`

Konsistente Codierungen und Codierungsdeklarationen

Wenn ein XML-Dokument verarbeitet oder mit einem anderen System ausgetauscht wird, muß darauf geachtet werden, daß die Codierungsdeklarationen der tatsächlichen Codierung des Dokuments entspricht. Es ist wichtig, daß die Konsistenz zwischen der Codierung eines Dokuments mit dem Client sichergestellt wird. Ist diese Konsistenz nicht gesichert, generieren manche XML-Tools wie beispielsweise Parser einen Fehler für eine Definitionseinheit mit einer Codierungsdeklaration, die sich von der in der Deklaration unterscheidet.

Abb. 19 zeigt, daß die Clients Codepages verwenden, die mit der Dokumentcodierung und der Codierungsdeklaration konsistent sind.

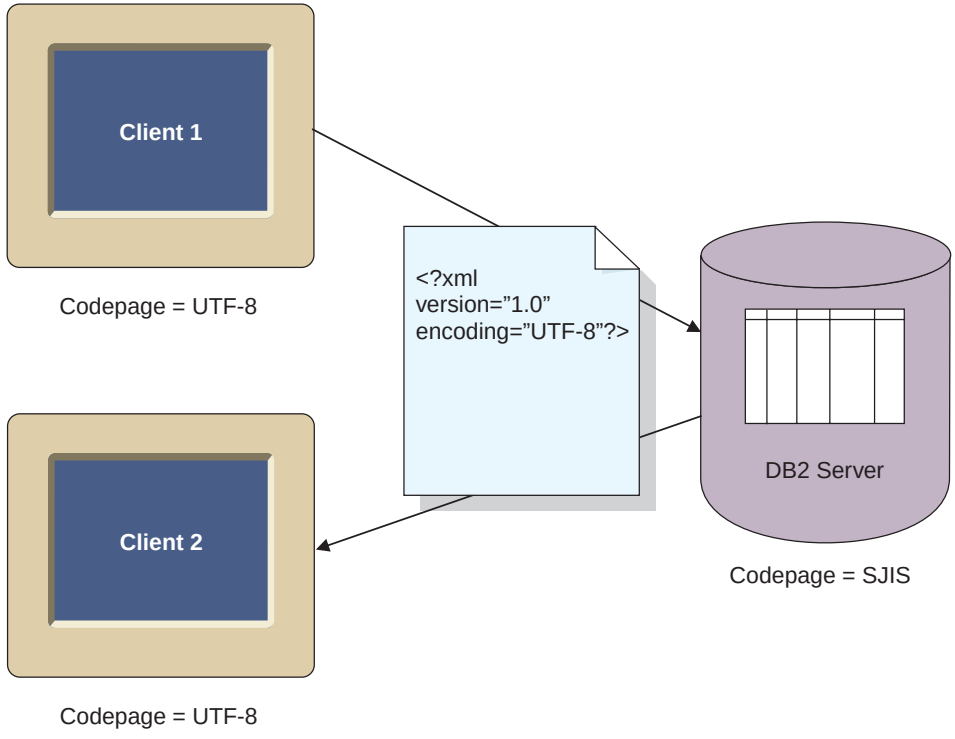


Abbildung 19. Die Clients haben übereinstimmende Codepages

Die Verwendung unterschiedlicher Codepages kann zu folgenden Situationen führen:

- Es können Umsetzungen durchgeführt werden, bei denen Daten verlorengehen, insbesondere wenn die Quellen-Codepage Unicode ist und die Ziel-Codepage eine andere Codepage. Unicode enthält das vollständige Set der Zeichenkonvertierungen. Wenn eine Datei von UTF-8 in eine Codepage konvertiert wird, die nicht alle in dem Dokument verwendeten Zeichen unterstützt, können bei der Konvertierung Daten verlorengehen.
- Die deklarierte Codierung des XML-Dokuments ist eventuell nicht mehr konsistent mit der tatsächlichen Dokumentcodierung, wenn das Dokument von einem Client mit einer anderen Codepage abgerufen wird.

Abb. 20 zeigt eine Umgebung, in der die Codepages der Clients nicht konsistent sind.

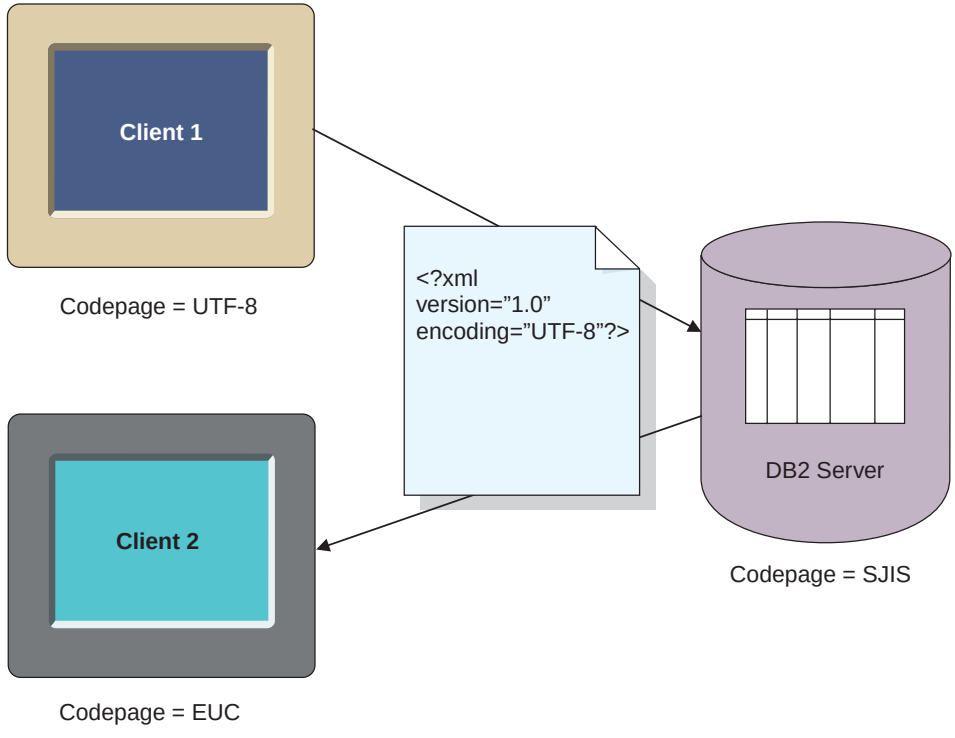


Abbildung 20. Die Codepages der Clients stimmen nicht überein.

Client2 empfängt das Dokument in EUC, das Dokument hat jedoch die Codierungsdeklaration UTF-8.

Eine Codierung deklarieren

Der Standardwert der Codierungsdeklaration lautet UTF-8, und eine fehlende Angabe einer Codierungsdeklaration bedeutet, daß das Dokument in UTF-8 ist.

So deklarieren Sie einen Codierungswert:

Geben Sie in der XML-Dokumentdeklaration die Codierungsdeklaration mit dem Namen der Codepage des Clients an. Beispiel:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Szenarien für die Konvertierung

Der XML Extender verarbeitet die XML-Dokumente in folgenden Fällen:

- Beim Speichern und Abrufen von XML-Spaltendaten mit der XML-Spaltenspeicherungs- und Zugriffsmethode
- Beim Zusammensetzen und Zerlegen von XML-Dokumenten

Die Codepage eines Dokuments wird konvertiert, wenn das Dokument von einem Client oder Server in eine Datenbank übergeben wird. Inkonsistenzen oder Beschädigungen von XML-Dokumenten treten meist bei der Konvertierung der Codepages des Clients, Servers oder der Datenbank auf. Bei der Auswahl der Codierungsdeklaration des Dokuments sowie bei der Planung, welche Clients und Server Dokumente aus der Datenbank importieren bzw. exportieren können, sollten Sie die in den obigen Tabellen beschriebenen Konvertierungen und die nachfolgend beschriebenen Szenarien beachten.

Die folgenden Szenarien beschreiben allgemeine Konvertierungsszenarien, die auftreten können:

Szenario 1: Dieses Szenario ist eine Konfiguration mit konsistenten Codierungen, ohne DB2-Konvertierung und mit einem vom Server importierten Dokument. Die Dokumentcodierungsdeklaration ist UTF-8, der Server ist UTF-8 und die Datenbank ist UTF-8.

1. Das Dokument wird über die UDF XMLClobfromFile in DB2 importiert.
2. Das Dokument wird auf den Server extrahiert.
3. DB2 braucht das Dokument nicht zu konvertieren, da die Server-Codepage und die Datenbank-Codepage identisch sind. Die Codierung und die Deklaration sind konsistent.

Szenario 2: Dieses Szenario ist eine Konfiguration mit konsistenten Codierungen, DB2-Konvertierung und mit einem vom Server importierten und auf den Client exportierten Dokument. Die Dokumentcodierung und -Deklaration ist SJIS, die Client-Codepage ist SJIS, und die Server- und Client-Codepage ist jeweils UTF-8.

1. Das Dokument wird mit der UDF XMLClobfromfile vom Server in DB2 importiert.
2. DB2 konvertiert das Dokument von SJIS und speichert es in UTF-8. Die Codierungsdeklaration und die Codierung sind in der Datenbank nicht konsistent.
3. Ein Client, der SJIS verwendet, fordert das Dokument zur Darstellung im Web-Browser an.
4. DB2 konvertiert das Dokument in SJIS, die Client-Codepage. Die Dokumentcodierung und die Deklaration sind jetzt auf dem Client konsistent.

Szenario 3: Dieses Szenario ist eine Konfiguration mit inkonsistenten Codierungen, DB2-Konvertierung und mit einem vom Server importierten und auf den Client exportierten Dokument. Die Dokumentcodierungsdeklaration ist SJIS für das eingehende Dokument. Die Server-Codepage ist SJIS, und Client und die Datenbank verwenden UTF-8.

1. Das Dokument wird über eine Speicherungs-UDF in die Datenbank importiert.
2. DB2 konvertiert das Dokument von SJIS in UTF-8. Die Codierung und die Deklaration sind inkonsistent.
3. Ein Client mit einer UTF-8-Codepage fordert das Dokument zur Darstellung in einem Web-Browser an.
4. DB2 führt keine Konvertierung durch, da die Client- und die Datenbank-Codepages identisch sind.
5. Die Dokumentcodierung und Deklaration sind inkonsistent, da die Deklaration SJIS verwendet und die Codierung UTF-8 ist. Das Dokument kann von einem XML-Parser oder anderen XML-Verarbeitungs-Tools nicht verarbeitet werden.

Szenario 4: Dieses Szenario ist eine Konfiguration mit Datenverlust, DB2-Konvertierung und mit einem von einem UTF-8-Server importierten Dokument: Die Dokumentcodierungsdeklaration ist UTF-8, der Server verwendet UTF-8, und die Datenbank verwendet SJIS.

1. Das Dokument wird über die UDF XMLClobfromFile in DB2 importiert.
2. DB2 konvertiert die Codierung in SJIS. Beim Importieren des Dokuments kann das in der Datenbank gespeicherte Dokument beschädigt werden, da es eventuell für die in UTF-8 dargestellten Zeichen keine entsprechende Darstellung in SJIS gibt.

Szenario 5:

Dieses Szenario ist eine Konfiguration mit einer Windows NT-Einschränkung. Unter Windows NT können die Ländereinstellungen nicht auf UTF-8 gesetzt werden; DB2 ermöglicht es dem Client jedoch, mit `db2set DB2CODEPAGE=1208` die Codepage auf UTF-8 zu setzen. In diesem Szenario befinden sich der Client und der Server auf der gleichen Maschine. Der Client ist UTF-8, der Server kann jedoch nicht auf UTF-8 gesetzt werden; er verwendet die Codepage 1252. Das Dokument ist als 1252 codiert, und die Codierungsdeklaration ist ibm-1252. Die Datenbank-Codepage ist UTF-8.

1. Das Dokument wird über eine Speicherungs-UDF vom Server importiert und von 1252 in 1208 konvertiert.
2. Das Dokument wird vom Client mit der UDF Content() von DB2 exportiert.
3. DB2 konvertiert das Dokument von UTF-8 in 1252, obwohl der Client eventuell 1208 erwartet. Der Grund hierfür ist, daß der Client sich auf dem gleichen System befindet wie der Server und auf 1208 gesetzt ist.

Inkonsistente XML-Dokumente vermeiden

In den oben beschriebenen Abschnitten wurde erläutert, wie ein XML-Dokument inkonsistente Codierungen haben kann, so daß ein Konflikt zwischen der Codierungsdeklaration und der Dokumentcodierung vorliegt. Inkonsistente Codierungen können Datenverluste oder unbrauchbare XML-Codierungen zur Folge haben.

Über die folgenden Empfehlungen können Sie sicherstellen, daß die Codierung des XML-Dokuments mit der Client-Codepage konsistent ist, bevor Sie das Dokument an einen XML-Prozessor wie beispielsweise einen Parser übergeben:

- Probieren Sie beim Exportieren eines Dokuments aus der Datenbank mit den XML Extender-UDFs eine der folgenden Techniken aus: (Annahme: der XML Extender hat die Datei in der Server-Codepage in das Dateisystem auf dem Server exportiert).
  - Wandeln Sie das Dokument in die deklarierte Codepage um.
  - Überschreiben Sie die deklarierte Codierung, falls das Tool über eine Überschreibungsfunktion verfügt.
  - Ändern Sie manuell die Codierungsdeklaration des exportierten Dokuments in die tatsächliche Codierung des Dokuments (also in die Server-Codepage)
- Probieren Sie beim Exportieren eines Dokuments aus der Datenbank mit den gespeicherten Prozeduren des XML Extender eine der folgenden Techniken aus: (Annahme: der Client fragt die Ergebnistabelle ab, in der das zusammengesetzte Dokument gespeichert ist)
  - Wandeln Sie das Dokument in die deklarierte Codepage um.
  - Überschreiben Sie die deklarierte Codierung, falls das Tool über eine Überschreibungsfunktion verfügt.
  - Vor dem Abfragen der Ergebnistabelle muß die Umgebungsvariable DB2CODEPAGE auf dem Client auf eine Codepage gesetzt werden, die mit der Codierungsdeklaration des XML-Dokuments kompatibel ist.
  - Ändern Sie manuell die Codierungsdeklaration des exportierten Dokuments in die tatsächliche Codierung des Dokuments (also in die Client-Codepage)

- **Einschränkung beim Verwenden von Unicode und Windows NT:** Unter Windows NT kann die Ländereinstellung des Betriebssystems nicht auf UTF-8 gesetzt werden. Beachten Sie beim Importieren und Exportieren von Dokumenten die folgenden Richtlinien:
  - Setzen Sie beim Importieren von Dateien und DTDs, die in UTF-8 codiert sind, die Client-Codepage auf UTF-8. Verwenden Sie hierzu den folgenden Befehl:

    ```
    db2set DB2CODEPAGE=1208
    ```

    Verwenden Sie diese Technik in folgenden Fällen:
    - Beim Einfügen einer DTD in die Tabelle db2xml.DTD_REF
    - Beim Aktivieren einer Spalte oder Objektgruppe
    - Beim Zerlegen gespeicherter Prozeduren
  - Beim Verwenden der UDFs Content() oder XMLxxxfromFile zum Importieren der XML-Dokumente müssen Dokumente in der Codepage entsprechend der Ländereinstellung des Server-Betriebssystems codiert sein; dies kann nicht UTF-8 sein.
  - Beim Exportieren einer XML-Datei aus der Datenbank. Setzen Sie die Client-Codepage mit dem folgenden Befehl, damit DB2 die Ergebnisdaten in UTF-8 codiert:

    ```
    db2set DB2CODEPAGE=1208
    ```

    Verwenden Sie diese Technik in folgenden Fällen:
    - Beim Abfragen der Ergebnistabelle nach dem Zusammensetzen
    - Beim Extrahieren von Daten aus einer XML-Spalte mit den Extraktions-UDFs
  - Beim Verwenden der UDFs Content() oder XMLxxxfromFile zum Exportieren von XML-Dokumenten in Dateien im Dateisystem des Servers sind die resultierenden Dokumente in der Codepage entsprechend der Ländereinstellung des Server-Betriebssystems codiert; dies kann nicht UTF-8 sein.

Anhang D. Die XML Extender-Begrenzungen

Die XML Extender-DAD-Datei, die gespeicherten Prozeduren und Tabellen haben folgende Begrenzungen.

Tabelle 58. XML Extender-Begrenzungen

Wert	Begrenzung
Tabelle in einer XML-Objektgruppe zum Zerlegen	1024 Zeilen aus jedem zerlegten XML-Dokument
Parameter für gepeicherte Prozeduren:	
XML-Dokument-CLOB	1 MB[2]
Dokumentzugriffsdefinition (DAD) CLOB	100 KB[2]
collectionName	30
spltName	30
dbName	18
defaultView	128
rootID	128
resultTabName	128
tablespace	128
tbName	128[1]
db2xml.DTD_REF-Tabellenspalten	
AUTHOR	128
CREATOR	128
UPDATOR	128
DTD-ID	128
CLOB	100 KB

Anmerkungen:

1. Wenn *tbName* über einen Schemanamen angegeben ist, darf der gesamte Name (einschließlich des Trennzeichens) nicht länger als 128 Zeichen sein.
2. Diese Größe kann geändert werden. Im Abschnitt „Die CLOB-Begrenzung vergrößern" auf Seite 222 finden Sie Informationen dazu, wie Sie diese Änderung vornehmen können.

Namen können erweitert werden, wenn DB2 sie von der Client-Codepage in die Datenbank-Codepage konvertiert. Ein Name kann auf dem Client innerhalb der Größenbegrenzung liegen und dennoch die Begrenzung überschreiten, wenn die gespeicherte Prozedur den konvertierten Namen erhält. Hinweise hierzu finden Sie im Abschnitt „National Language Support Application Development" des Kapitels „Programming in Complex Environments" im Handbuch *DB2 Application Development Guide*.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, daß nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen Schutzrechte der IBM verletzen. Die Verwantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices 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. Lizenzanfragen sind schriftlich an folgende Adresse zu richten:

IBM Europe
Director of Licensing
92066 Paris la Defense, Cedex
France
Anfragen an obige Adresse müssen auf englisch formuliert werden.

Lizenzanfragen zu Doppelbyteinformationen (DBCS) sind an die IBM Patentabteilung im jeweiligen Land zu richten oder schriftlich in englischer Sprache an folgende Adresse:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Änderungen des Textes bleiben vorbehalten.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließ-

lich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

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 Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM oder einer äquivalenten Vereinbarung.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche. Fragen hinsichtlich des Leistungsspektrums von Produkten anderer Hersteller als IBM sind an den jeweiligen Hersteller des Produkts zu richten.

Die oben genannten Erklärungen bezüglich der Produktstrategien und Absichtserklärungen von IBM stellen die gegenwärtige Absicht der IBM dar, unterliegen Änderungen oder können zurückgenommen werden, oder repräsentieren nur die Ziele der IBM.

COPYRIGHT-LIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Beispielprogramme geschrieben sind. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet.

Marken

Folgende Namen sind in gewissen Ländern Marken der IBM Corporation:

DB2	Net.Data
DB2 Extenders	OS/2
DB2 Universal Database	OS/390
IBM	OS/400
IMS	VTAM

Java und alle Java-basierenden Marken und Logos sind in gewissen Ländern Marken oder eingetragene Marken von Sun Microsystems, Inc.

Microsoft, Windows, Windows NT und das Windows-Logo sind in gewissen Ländern Marken oder eingetragene Marken von Microsoft Corporation.

UNIX ist eine eingetragene Marke und wird ausschließlich von X/Open Company Limited lizenziert.

Alle weiteren Namen von Firmen, Produkten oder Dienstleistungen können Marken oder Dienstleistungsmarken anderer Unternehmen sein.

Glossar

Abfrage. Ein Anfordern von Informationen aus der Datenbank entsprechend spezifischer Bedingungen; eine Abfrage kann beispielsweise aus einer Kundentabelle eine Liste aller Kunden abrufen, deren Umsatz eine bestimmte Größe hat.

Abschnittssuche. Ermöglicht die Textsuche innerhalb eines Abschnitts, der von der Anwendung definiert werden kann. Zur Unterstützung der strukturellen Textsuche kann ein Abschnitt über den abgekürzten Xpath-Standortpfad angegeben werden.

Absoluter Standortpfad. Der vollständige Pfadname eines Objekts. Der absolute Pfadname beginnt auf der höchsten Ebene, einem "Root"-Element, das durch einen Schrägstrich (/) oder umgekehrten Schrägstrich (\) gekennzeichnet ist.

Anwendungsprogrammierschnittstelle (Application Programming Interface, API).

(1) Eine vom Betriebssystem oder einem separat bestellbaren Lizenzprogramm bereitgestellte Funktionsschnittstelle. Eine API ermöglicht einem in einer höheren Programmiersprache geschriebenen Anwendungsprogramm die Verwendung spezifischer Daten oder Funktionen des Betriebssystems bzw. des Lizenzprogramms.

(2) In DB2 eine Funktion innerhalb der Schnittstelle, z. B. die API zum Abrufen von Fehlernachrichten.

(3) In DB2 eine Funktion innerhalb der Schnittstelle. Beispiel: Die API zum Abrufen von Fehlernachrichten.

API. Siehe *Anwendungsprogrammierschnittstelle*.

Attribut. Siehe *XML-Attribut*.

attribute_node. (Attributknoten) Steht für ein Attribut eines Elements.

B-Baumstruktur-Indexierung. Das native Indexierungsschema der DB2-Steuerkomponente. Sie erstellt Indexeinträge in der B-Baumstruktur. Unterstützt DB2-Basisdatentypen.

Bedingung. Die Angabe der Kriterien zur Auswahl der XML-Daten oder die Angabe, wie die XML-Objektgruppentabellen verbunden werden sollen.

Benutzerdefinierte Funktion (User-Defined Function, UDF). Eine Funktion, die für das Datenbankverwaltungssystem definiert und auf die später in SQL-Abfragen verwiesen wird. Folgende Funktionen sind möglich:

- Eine externe Funktion, bei der der Hauptteil der Funktion in einer Programmiersprache geschrieben ist, deren Argumente Skalarwerte sind und deren Aufruf einen Skalarwert ergibt.
- Eine Funktion, die von einer anderen integrierten oder benutzerdefinierten Funktion implementiert wurde, die dem DBMS bereits bekannt ist. Diese Funktion kann eine skalare Funktion oder eine Spaltenfunktion (Zusammenfassung) sein; sie gibt einen einzelnen Wert aus einer Wertegruppe (z. B. MAX oder AVG) zurück.

Benutzerdefinierter Typ (User-Defined Type, UDT). Ein nicht nativer Datentyp im Datenbankmanager; er wurde von einem Benutzer erstellt. Siehe *eindeutiger Typ*.

Benutzertabelle. Eine Tabelle, die für eine Anwendung erstellt wurde und von dieser verwendet wird.

Browser. Siehe *Web-Browser*.

CLOB. Großes Zeichenobjekt (Character Large Object).

DAD. Siehe *Dokumentzugriffsdefinition*.

Datenaustausch. Die gemeinsame Benutzung von Daten über verschiedene Anwendungen hin-

weg. XML unterstützt den Austausch von Daten, ohne daß diese zunächst in ein geeignetes Format umgewandelt werden müssen.

Datenquelle. Ein lokaler oder ferner relationaler oder nicht relationaler Datenmanager, der Datenzugriffe über einen ODBC-Treiber unterstützt, der wiederum die ODBC-APIs unterstützt.

Datentyp. Ein Attribut von Spalten und Literalen.

Datenverbindung. Ein DB2-Datentyp, der logische Verweise von der Datenbank auf eine Datei außerhalb der Datenbank ermöglicht.

DBCLOB. Großes Doppelbyte-Zeichenobjekt (Double-byte Large Object).

Dokumenttypdefinition (DTD). Eine Gruppe von Deklarationen für XML-Elemente und -Attribute. Die DTD definiert, welche Elemente in dem XML-Element verwendet werden, in welcher Reihenfolge sie verwendet werden und welche Elemente weitere Elemente enthalten können. Sie können eine DTD einer DAD-Datei zuordnen, um die Gültigkeit von XML-Dokumenten zu prüfen.

Dokumentzugriffsdefinition (Document Access Definition, DAD). Wird zur Definition des Indexierungsschemas für eine XML-Spalte oder ein Zuordnungsschema einer XML-Objektgruppe verwendet. Mit der DAD kann eine XML Extender-Spalte einer XML-Objektgruppe, die für XML formatiert ist, aktiviert werden.

DTD. (1) . (2) Siehe *Dokumenttypdefinition*.

DTD-Referenztabelle (DTD_REF-Tabelle). Eine Tabelle mit DTDs, die zum Überprüfen von XML-Dokumenten und zur Unterstützung der Anwendungen bei der Definition einer DAD-Datei verwendet werden. Benutzer können ihre eigenen DTDs in die Tabelle DTD_REF einfügen. Diese Tabelle wird beim Aktivieren einer Datenbank für XML erstellt.

DTD_REF-Tabelle. DTD-Referenztabelle.

DTD-Repository. Eine DB2-Tabelle mit dem Namen DTD_REF. Jede Zeile dieser Tabelle stellt eine DTD mit zusätzlichen Metadaten-Informationen dar.

EDI. Elektronischer Datenaustausch.

Eindeutiger Typ. Siehe *benutzerdefinierter Typ*.

Einfacher Standortpfad. Eine Folge von Elementtypnamen, die durch einen einzelnen Schrägstrich (/) verbunden sind.

Eingebettetes SQL. SQL-Anweisungen, die in einem Anwendungsprogramm codiert wurden. Siehe *Statisches SQL*.

Elektronischer Datenaustausch (Electronic Data Interchange, EDI). Ein Standard für den elektronischen Datenaustausch für Business-to-Business-Anwendungen (B2B-Anwendungen).

Element. Siehe *XML-Element*.

element_node. (Elementknoten) Steht für ein Element. Bei dem element_node kann es sich um das Stammelement oder ein untergeordnetes Element handeln.

Ergebnisgruppe. Eine Gruppe von Zeilen, die von einer gespeicherten Prozedur zurückgegeben wurde.

Ergebnistabelle. Eine Tabelle, die Zeilen als Ergebnis einer SQL-Abfrage oder der Ausführung einer gespeicherten Prozedur enthält.

Extensible Stylesheet Language (XSL). Eine Sprache für Formatvorlagen. XSL besteht aus zwei Teilen: einer Sprache zur Umformung von XML-Dokumenten und einem XML-Vokabular zur Angabe der Semantik für die Formatierung.

Extensive Stylesheet Language Transformation (XSLT). Eine Sprache zur Umsetzung von XML-Dokumenten in andere XML-Dokumente. XSLT wurde zur Verwendung als Teil von XSL konzipiert, einer Sprache für Formatvorlagen für XML.

Externe Datei. Eine Datei, die in einem von DB2 aus gesehen externen Dateisystem vorhanden ist.

Fremdschlüssel. Ein Schlüssel, der Teil der Definition einer referentiellen Integritätsbedingung ist und aus einer oder mehreren Spalten einer abhängigen Tabelle besteht.

Gespeicherte Prozedur. Ein Block von Prozeduranweisungen und eingebetteten SQL-Anweisungen, der in einer Datenbank gespeichert ist und mit seinem Namen aufgerufen werden kann. Gespeicherte Prozeduren bieten die Möglichkeit, ein Anwendungsprogramm in zwei Teilen auszuführen. Ein Teil wird auf dem Client ausgeführt, der andere auf dem Server. Somit kann ein einziger Aufruf mehrere Zugriffe auf die Datenbank generieren.

Großes Doppelbyte-Zeichenobjekt (Double-Byte Character Large Object, DBCLOB). Eine Zeichenfolge aus Doppelbytezeichen oder eine Kombination aus Einzel- und Doppelbytezeichen, wobei die Zeichenfolge bis zu 2 GB lang sein kann. DBCLOBs haben eine zugeordnete Codepage. Textobjekte, die Doppelbytezeichen enthalten, werden in der DB2-Datenbank als DBCLOBs gespeichert.

Großes Objekt (Large Object, LOB). Eine Bytefolge, deren Länge bis zu 2 GB betragen kann. Ein LOB kann einen von drei Typen haben: *Großes Binärobjekt* (BLOB), *Großes Zeichenobjekt (CLOB)* oder *Großes Doppelbytezeichenobjekt (DBCLOB)*.

Großes Zeichenobjekt (Character Large Object, CLOB). Eine Zeichenfolge aus Einzelbytezeichen, wobei die Zeichenfolge bis zu 2 GB lang sein kann. CLOBs haben eine zugeordnete Codepage. Textobjekte, die Einzelbytezeichen enthalten, werden in einer DB2-Datenbank als CLOBs gespeichert.

Gültiges Dokument. Ein XML-Dokument mit einer zugeordneten DTD. Damit das XML-Dokument gültig ist, dürfen die in seiner DTD definierten syntaktischen Regeln nicht verletzt werden.

Gültigkeitsprüfung. Das Sicherstellen mit Hilfe einer DTD, daß das XML-Dokument gültig ist. Außerdem das Ermöglichen einer strukturellen Suche in XML-Daten. Die DTD wird im DTD-Repository gespeichert.

Index. Eine Gruppe von Zeigern, die nach dem Wert eines Schlüssels logisch geordnet sind. Indizes bieten einen schnellen Zugriff auf Daten und können die Eindeutigkeit der Zeilen in der Tabelle erzwingen.

Java Database Connectivity (JDBC). Eine Anwendungsprogrammierschnittstelle (API) mit den gleichen Merkmalen wie Open Database Connectivity (ODBC), die jedoch speziell zur Verwendung mit Java-Datenbankanwendungen konzipiert wurde. Für Datenbanken ohne JDBC-Treiber umfaßt JDBC außerdem eine JDBC-zu-ODBC-Brücke; diese Brücke ist ein Mechanismus zum Umsetzen von JDBC in ODBC. JDBC stellt die JDBC-API für Java-Datenbankanwendungen dar und setzt diesen Code in ODBC um. JDBC wurde von Sun Microsystems, Inc. sowie verschiedenen Partnern und Lieferanten entwickelt.

JDBC. Java Database Connectivity.

Knoten. Bei der Datenbank-Partitionierung gleichbedeutend mit ″Datenbank-Partition″.

Knoten einer relationalen Datenbank (RDB_node). Ein Knoten, der eine oder mehrere Elementdefinitionen für Tabellen, wahlfreie Spalten und wahlfreie Bedingungen enthält. Mit den Tabellen und Spalten wird festgelegt, wie die XML-Daten in der Datenbank gespeichert werden. Die Bedingung gibt entweder die Kriterien zur Auswahl der XML-Daten an oder dazu, wie die XML-Objektgruppentabellen verbunden werden soll.

LOB. Großes Objekt.

Lokales Dateisystem. Ein in DB vorhandenes Dateisystem

Mehrfaches Vorkommen. Ein Hinweis darauf, ob ein Spaltenelement oder Attribut mehr als einmal in einem Dokument verwendet werden kann. Ein mehrfaches Vorkommen wird in der DAD-Datei angegeben.

Metadatentabelle. Siehe *Tabellen zur Verwaltungsunterstützung*.

Objekt. Bei der objektorientierten Programmierung eine Abstraktion aus Daten und den diesen Daten zugeordneten Operationen.

ODBC. Open Database Connectivity.

Open Database Connectivity. Eine Standard-Anwendungsprogrammierschnittstelle (API für den Zugriff auf Daten in relationalen und nicht relationalen Datenbankverwaltungssystemen. Mit dieser API können Datenbankanwendungen auf Daten zugreifen, die in Datenbankverwaltungssystemen auf verschiedenen Computern gespeichert sind, selbst wenn jedes Datenbankverwaltungssystem ein anderes Datenspeicherungsformat und eine andere Programmierschnittstelle verwendet. ODBC basiert auf der CLI-Spezifikation (Call Level Interface) der X/Open SQL Access Group; es wurde von Digital Equipment Corporation (DEC), Lotus, Microsoft und Sybase entwickelt. Gegensatz zu *Java Database Connectivity*.

Partition. Ein Speicherabschnitt mit einer festen Größe.

Pfadausdruck. Siehe *Standortpfad*.

Prädikat. Ein Element einer Suchbedingung, das eine Vergleichsoperation ausdrückt oder impliziert.

Primärschlüssel. Ein eindeutiger Schlüssel, der Teil der Definition einer Tabelle ist. Ein Primärschlüssel ist der übergeordnete Schlüssel einer referentiellen Integritätsbedingung.

Prozedur. Siehe *gespeicherte Prozedur*.

RDB_node. (RDB-Knoten) Knoten einer relationalen Datenbank.

RDB_node-Zuordnung. Der Standort des Inhalts eines XML-Elements oder des Werts eines XML-Attributs, die über den RDB_node definiert sind. Der XML Extender verwendet diese Zuordnung, um festzustellen, wo die XML-Daten gespeichert bzw. von wo sie abgerufen werden sollen.

Root-ID. Eine eindeutige Kennung, die alle Seitentabellen der Anwendungstabelle zuordnet.

Schema. Eine Objektgruppe von Datenbankobjekten wie beispielsweise Tabellen, Sichten, Indizes oder Auslösern. Ein Schema bietet eine logische Klassifizierung von Datenbankobjekten.

Seitentabelle. Zusätzliche Tabellen, die vom XML Extender erstellt werden, um die Leistung beim Suchen von Elementen und Attributen in einer XML-Spalte zu verbessern.

Skalarfunktion. Eine SQL-Operation, die einen einzigen Wert aus einem anderen Wert erstellt. Sie wird als Funktionsname, gefolgt von einer in Klammern gesetzte Liste von Argumenten, dargestellt.

Spaltendaten. Die in einer DB2-Spalte gespeicherten Daten. Der Typ der Daten kann ein beliebiger von DB2 unterstützter Datentyp sein.

SQL-Zuordnung. Eine Definition der Beziehung zwischen dem Inhalt eines XML-Elements oder -Werts und den relationalen Daten; hierbei werden ein oder mehrere SQL-Anweisungen und das XSLT-Datenmodell verwendet. Der XML Extender verwendet die Definition, um festzustellen, wo die XML-Daten gespeichert bzw. von wo sie abgerufen werden sollen. Die SQL-Zuordnung wird im Element SQL_stmt in der DAD-Datei definiert.

Stammelement. Das Anfangselement eines XML-Dokuments.

Standardsicht. Eine Darstellung von Daten, in denen eine XML-Tabelle und alle ihre zugehörigen Seitentabellen miteinander verknüpft sind.

Standardumsetzungsfunktion. Setzt den SQL-Basistyp in einen UDT um.

Standortpfad. Der Standortpfad ist eine Folge von XML-Befehlen, die ein XML-Element oder -Attribut kennzeichnen. Der Standort kennzeichnet die Struktur des XML-Dokuments und gibt den Kontext für das Element bzw. Attribut an. Ein einfacher Schrägstrich (/) als Pfad gibt an, daß der Kontext das gesamte Dokument ist. Der Standortpfad wird in den Extraktions-UDFs zur

Identifikation der zu extrahierenden Elemente und Attribute verwendet. Der Standortpfad wird auch in der DAD-Datei zum Angeben der Zuordnungsdatei zwischen einem XML-Element bzw. -attribut und einer DB2-Spalte beim Definieren des Indexierungsschemas für die XML-Spalte verwendet. Darüber hinaus wird der Standortpfad vom Text Extender für eine strukturelle Textsuche verwendet.

Statisches SQL. SQL-Anweisungen, die in ein Programm eingebettet sind und bei der Vorbereitung des Programms vorbereitet werden, bevor das Programm ausgeführt wird. Nach der Vorbereitung ändert sich eine statische SQL-Anweisung nicht, obwohl sich die Werte der in der Anweisung angegebenen Host-Variablen ändern können.

Struktureller Textindex. Indexieren von Textzeichenfolgen entsprechend der Baumstruktur des XML-Dokuments mit dem DB2 Text Extender.

Tabellenbereich. Eine Abstraktion einer Sammlung von Behältern, in denen Datenbankobjekte gespeichert werden. Ein Tabellenbereich bietet eine indirekte Zuordnung zwischen einer Datenbank und den darin gespeicherten Tabellen.

- Ein Tabellenbereich enthält Speicherplatz auf den ihm zugeordneten Medienspeichereinheiten.
- In einem Tabellenbereich werden Tabellen erstellt. Diese Tabellen belegen Speicherplatz in den zu dem Tabellenbereich gehörenden Behältern. Die Bereiche Daten, Index, Langfeld und LOB einer Tabelle können in dem gleichen Tabellenbereich gespeichert oder in separate Tabellenbereiche heruntergebrochen werden.

Tabellen zur Verwaltungsunterstützung. Eine Tabelle, mit der ein DB2 Extender Benutzeranfragen zu XML-Objekten verarbeitet. Manche Tabellen zur Verwaltungsunterstützung kennzeichnen Benutzertabellen und -spalten, die für einen Extender aktiviert wurden. Andere Tabellen zur Verwaltungsunterstützung enthalten Attributinformationen zu Objekten in aktivierten Spalten. Synonym zu Metadatentabelle.

text_node. (Textknoten) Eine Darstellung des CDATA-Texts eines Elements.

Top element_node. (Anfangselementknoten) Eine Darstellung des Stammelements des XML-Dokuments in der DAD.

Überladene Funktion. Ein Funktionsname, für den mehrere Funktionsexemplare vorhanden sind.

UDF. Siehe *benutzerdefinierte Funktion*.

UDT. Siehe *benutzerdefinierter Typ*.

Umsetzungsfunktion. Eine Funktion zum Umsetzen von Exemplaren eines (Quellen-) Datentyps in Exemplare eines anderen (Ziel-) Datentyps. Im allgemeinen hat eine Umsetzungsfunktion den Namen des Zieldatentyps. Sie hat ein einziges Argument, dessen Typ der Quellendatentyp ist; der Rückgabetyp ist der Zieldatentyp.

UNION. Eine SQL-Operation, die die Ergebnisse aus zwei Auswahlanweisungen kombiniert. UNION wird häufig zum Zusammenfügen von Listen von Werten aus verschiedenen Tabellen verwendet.

Unterabfrage. Eine vollständige SELECT-Anweisung, die in einer Suchbedingung einer SQL-Anweisung verwendet wird.

URL. Uniform Resource Locator.

URL-Adresse (Uniform Resource Locator). Eine Adresse, die einen HTTP-Server und wahlweise ein Verzeichnis und einen Dateinamen kennzeichnet, z. B.:
http://www.ibm.com/data/db2/extenders.

Verknüpfen. Eine relationale Operation, die das Abrufen von Daten aus zwei oder mehr Tabellen entsprechend den übereinstimmenden Spaltenwerten ermöglicht.

Verknüpfte Sicht. Eine mit der Anweisung "CREATE VIEW" erstellte DB2-Sicht, die zwei oder mehr Tabellen miteinander verknüpft.

Volltextsuche. Mit dem DB2 Text Extender eine Suche nach Textzeichenfolgen an einer beliebigen Stelle ohne Rücksicht auf die Dokumentstruktur.

Web-Browser. Ein Client-Programm, das Anfragen an einen Web-Server startet und die vom Server zurückgegebenen Informationen anzeigt.

Wohlgeformtes Dokument. Ein XML-Dokument, das keine DTD enthält. Obwohl es die XML-Spezifikation erfüllt, muß ein Dokument mit einer gültigen DTD außerdem auch wohlgeformt sein.

XML. eXtensible Markup Language.

XML-Attribut. Ein beliebiges durch ATTLIST im XML-Element in der DTD angegebenes Attribut. Der XML Extender verwendet den Standortpfad zum Kennzeichnen eines Attributs.

XML-Befehl. Ein gültiger Befehl der XML-Formatierungssprache, im wesentlichen das XML-Element. Die Begriffe Befehl und Element sind in diesem Zusammenhang austauschbar.

XML-Element. Ein beliebiger XML-Befehl oder ein ELEMENT, wie in der XML-DTD angegeben. Der XML Extender verwendet den Standortpfad zum Kennzeichnen eines Elements.

XML-Objekt. Entspricht einem XML-Dokument.

XML-Objektgruppe. Eine Objektgruppe aus relationalen Tabellen, die die Daten zum Zusammensetzen der XML-Dokumente bzw. die Daten aus den zerlegten XML-Dokumenten enthalten.

XML Path Language. Eine Sprache zur Adressierung von Teilen eines XML-Dokuments. XML Path Language wurde zur Verwendung durch XSLT konzipiert. Jeder Standortpfad kann mit Hilfe der für XPath definierten Syntax ausgedrückt werden.

XML-Spalte. Eine Spalte in der Anwendungstabelle, die für die UDTs des XML Extender aktiviert wurde.

XML-Tabelle. Eine Anwendungstabelle, die eine oder mehrere XML Extender-Spalten enthält.

XML-UDF. Eine benutzerdefinierte DB2-Funktion, die vom XML Extender bereitgestellt wird.

XML-UDT. Ein benutzerdefinierter DB2-Typ, der vom XML Extender bereitgestellt wird.

XPath. Eine Sprache zur Adressierung von Teilen eines XML-Dokuments.

XPath-Datenmodell. Die Baumstruktur zum Modellieren und Navigieren eines XML-Dokuments mit Hilfe von Knoten.

XSL. XML Stylesheet Language.

XSLT. XML Stylesheet Language Transformation.

Zeigereinheit. Ein Zeiger, der zum Lokalisieren eines Objekts verwendet werden kann. In DB2 ist die LOB-Zeigereinheit (Large Object Block) der Datentyp zum Lokalisieren von LOBs.

Zerlegen. Zerlegt XML-Dokumente in eine Objektgruppe relationaler Tabellen in einer XML-Objektgruppe.

Zugriffs- und Speichermethode. Ordnet XML-Dokumente einer DB2-Datenbank über zwei wesentliche Zugriffs- und Speichermethoden zu: XML-Spalten und XML-Objektgruppen. Siehe auch *XML-Spalte* und *XML-Objektgruppe*.

Zuordnungsschema. Eine Definition, die festlegt, wie XML-Daten in einer relationalen Datenbank dargestellt werden. Das Zuordnungsschema wird in der DAD-Datei angegeben. Der XML Extender bietet zwei Arten von Zuordnungsschemata: *SQL-Zuordnung* und *Zuordnung über die relationale Datenbank (RDB_node)*.

Zusammensetzen. Das Generieren von XML-Dokumenten aus relationalen Daten in einer XML-Objektgruppe.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Kontaktaufnahme mit IBM

Bei technischen Problemen lesen Sie bitte die entsprechenden Korrekturmaßnahmen im Handbuch *Troubleshooting Guide* und führen Sie diese aus, bevor Sie sich mit der IBM Kundenunterstützung in Verbindung setzen.Mit Hilfe dieses Handbuchs können Sie Informationen sammeln, die die DB2-Kundenunterstützung zur Fehlerbehebung verwenden kann.

Wenn Sie weitere Informationen benötigen oder eines der DB2 Universal Database-Produkte bestellen möchten, setzen Sie sich mit einem IBM Ansprechpartner in einer lokalen Geschäftsstelle oder einem IBM Software-Vertriebspartner in Verbindung.

Telefonische Unterstützung erhalten Sie unter der folgenden Nummer:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.

Produktinformationen

Telefonische Unterstützung erhalten Sie unter den folgenden Nummern:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180/55 090 können Sie Handbücher telefonisch bestellen.

http://www.ibm.com/software/data/
Auf den DB2-World Wide Web-Seiten erhalten Sie aktuelle DB2-Informationen wie Neuigkeiten, Produktbeschreibungen, Schulungspläne und vieles mehr.

http://www.ibm.com/software/data/db2/library/
Mit **DB2 Product and Service Technical Library** können Sie auf häufig gestellte Fragen, Berichtigungen, Handbücher und aktuelle technische DB2-Informationen zugreifen.

Anmerkung: Diese Informationen stehen möglicherweise nur auf Englisch zur Verfügung.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
Auf der Web-Site für die Bestellung internationaler Veröffentlichungen (International Publications) finden Sie Informationen zum Bestellverfahren.

http://www.ibm.com/education/certify/
Das 'Professional Certification Program' auf der IBM Web-Site stellt Zertifizierungstestinformationen für eine Reihe von IBM Produkten, u. a. auch DB2, zur Verfügung.

ftp.software.ibm.com
Melden Sie sich als *anonymous* an. Im Verzeichnis `/ps/products/db2` finden Sie Demo-Versionen, Berichtigungen, Informationen und Tools zu DB2 und vielen zugehörigen Produkten.

comp.databases.ibm-db2, bit.listserv.db2-l
Über diese Internet-Newsgroups können DB2-Benutzer Ihre Erfahrungen mit den DB2-Produkten austauschen.

Für Compuserve: GO IBMDB2
Geben Sie diesen Befehl ein, um auf IBM DB2 Family Forums zuzugreifen. Alle DB2-Produkte werden über diese Foren unterstützt.

In Anhang A des Handbuchs *IBM Software Support Handbook* finden Sie Informationen dazu, wie Sie sich mit IBM in Verbindung setzen können. Rufen Sie die folgende Web-Seite auf, um auf dieses Dokument zuzugreifen: http://www.ibm.com/support/. Wählen Sie anschließend die Verbindung zum IBM Software Support Handbook am unteren Rand der Seite aus.

Anmerkung: In einigen Ländern sollten sich die IBM Vertragshändler an die innerhalb ihrer Händlerstruktur vorgesehene Unterstützung wenden, nicht an die IBM Unterstützungsfunktion.

Antwort

IBM DB2 Universal Database
XML Extender
Verwaltung und Programmierung
Version 7

Anregungen zur Verbesserung und Ergänzung dieser Veröffentlichung nehmen wir gerne entgegen. Bitte informieren Sie uns über Fehler, ungenaue Darstellungen oder andere Mängel.

Zur Klärung technischer Fragen sowie zu Liefermöglichkeiten und Preisen wenden Sie sich bitte entweder an Ihre IBM Geschäftsstelle, Ihren IBM Geschäftspartner oder Ihren Händler.

Unsere Telefonauskunft ″HALLO IBM″ (Telefonnr.: 01803/31 32 33) steht Ihnen ebenfalls zur Klärung allgemeiner Fragen zur Verfügung.

Kommentare:

Danke für Ihre Bemühungen.

Sie können ihre Kommentare betr. dieser Veröffentlichung wie folgt senden:
- Als Brief an die Postanschrift auf der Rückseite dieses Formulars
- Als FAX an die folgende Nummer: USA und Kanada: 800-426-7773
- Als E-Mail an die folgende Adresse: COMMENTS@VNET.IBM.COM

Name	Adresse
Firma oder Organisation	
Rufnummer	E-Mail-Adresse

Antwort

IBM

IBM CORPORATION
Department BWE/H3
PO Box 49023
San Jose, CA

Printed in Denmark