IBM DB2 Universal Database

XML Extender Verwaltung und Programmierung

Version 8.2

SC12-3062-01

IBM DB2 Universal Database

XML Extender Verwaltung und Programmierung

Version 8.2

SC12-3062-01

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter *Bemerkungen* 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 Version 8.2,
IBM Form SC27-1234-1,
herausgegeben von International Business Machines Corporation, USA

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

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

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
April 2004

Inhaltsverzeichnis

Zu diesem Handbuch

Dieser Abschnitt enthält die folgenden Informationen:

Zielgruppe

Dieses Handbuch hat folgende Zielgruppe:

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

  http://www.w3.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 UDB vertraut sind.
- DB2-Anwendungsprogrammierer, die mit SQL sowie einer oder mehreren Programmiersprachen für DB2 UDB-Anwendungen vertraut sind.

Aktuelle Version dieses Handbuchs abrufen

Sie können die neueste Version dieses Handbuchs über die XML Extender-Website 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 zu XML Extender und darüber, wie Sie ihn in Ihren Geschäftsanwendungen einsetzen können. Er enthält ein Szenario "Erste Schritte", das Ihnen den Einstieg erleichtert.

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

Teil 3. Programmierung
Dieser Teil beschreibt die Verwaltung Ihrer XML-Daten. Lesen Sie diesen Teil, wenn Sie in einem DB2 UDB-Anwendungsprogramm auf XML-Daten zugreifen und diese 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 von XML Extender ausgegeben werden. Lesen Sie diesen Teil, wenn Sie mit den Konzepten und Tasks von 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.

Hervorhebungskonventionen

In diesem Handbuch werden folgende Konventionen verwendet:

Text in Fettdruck kennzeichnet:

- Befehle
- Feldnamen
- Menünamen
- Druckknöpfe

Text in Kursivschrift kennzeichnet:

- Variable Parameter, die durch einen Wert zu ersetzen sind
- Hervorgehobene Begriffe
- Erstes Auftreten eines Glossareintrags

Text in Großbuchstaben kennzeichnet:

- Datentypen
- Spaltennamen
- Tabellennamen

Beispieltext kennzeichnet:

- Systemnachrichten
- Von Ihnen eingegebene Werte
- Codierungsbeispiele
- Verzeichnisnamen
- Dateinamen

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.

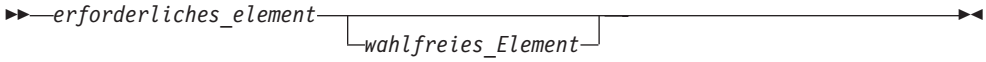
  Das Symbol —► gibt an, dass die Syntax der Anweisung auf der nächsten Zeile fortgesetzt wird.

  Das Symbol ►— gibt an, dass eine Anweisung von der vorherigen Zeile fortgesetzt wird.

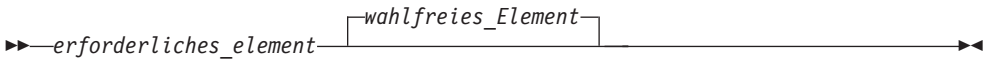
  Das Symbol —►◄ kennzeichnet das Ende einer Anweisung.

  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.

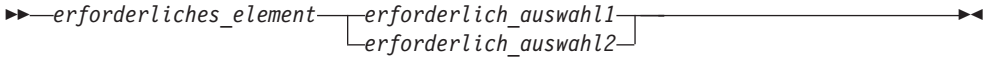


  Wenn ein wahlfreies Element oberhalb des Hauptpfads erscheint, hat dieses Element keine Auswirkung auf die Ausführung der Anweisung und dient lediglich der besseren Lesbarkeit.

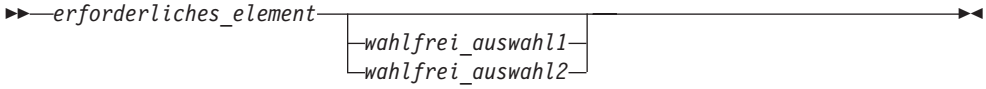


- Wenn Sie aus zwei oder mehr Elementen auswählen können, werden diese übereinander als Stapel angezeigt.

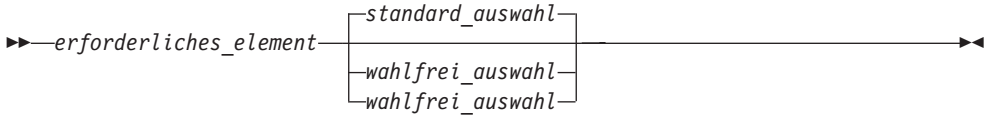
  Wenn Sie eines dieser Elemente auswählen *müssen*, wird eines der Elemente des Stapels im Hauptpfad angezeigt.



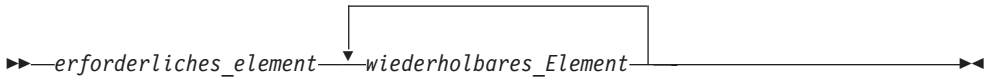
  Wenn die Auswahl eines dieser Elemente wahlfrei ist, erscheint der gesamte Stapel unterhalb des Hauptpfads.



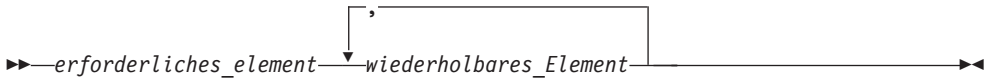
Wenn eines der Elemente den Standardwert darstellt, erscheint es oberhalb des Hauptpfads, und die weiteren Auswahloptionen werden darunter angezeigt.



- Ein Pfeil, der oberhalb der Hauptlinie zurückkehrt, weist darauf hin, dass ein Element 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). In XML Extender können Schlüsselwörter beliebig in Groß- und Kleinbuchstaben verwendet 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.

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

Einführung in XML Extender

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 Einfügen von Zeilen aus XML-Dokumenten in relationale Tabellen. XML Extender bietet neue Datentypen, Funktionen und gespeicherte Prozeduren zum Verwalten Ihrer XML-Daten in DB2 UDB.

XML Extender ist für die folgenden Betriebssysteme verfügbar:

- Windows NT
- Windows 2000
- AIX
- Solaris™-Betriebsumgebung
- Linux
- HP-UX auf einer PA-RISC-Plattform
- OS/390® und z/OS™
- iSeries™

Die HP-UX-Plattform ist die momentan einzige 64-Bit-Plattform, die DB2 XML Extender unterstützt.

Bevor Sie DB2 UDB XML Extender verwenden, sollten Sie die Speicherposition für temporäre Dateien festlegen. Die Umgebungsvariable DB2DXXTEMP steuert die Speicherposition der temporären Dateien von XML Extender. Wenn die Variable nicht gesetzt ist, bestimmt der Wert von TMP die Speicherposition der temporären Dateien. Gehen Sie wie folgt vor, um unter Windows 2000 den Wert von DB2DXXTEMP festzulegen:

1. Stellen Sie sicher, dass Sie mit der Benutzer-ID angemeldet sind, die Sie mit DB2 verwenden.
2. Klicken Sie **Start —> Einstellungen —> Systemsteuerung** an.
3. Klicken Sie das Symbol **System** doppelt an.
4. Klicken Sie auf der Seite **Erweitert** des Notizbuchs **Systemeigenschaften Umgebungsvariablen** an.
5. Klicken Sie im Bereich **Systemvariablen Neu** an. Geben Sie als Variablennamen `DB2DXXTEMP` und einen Wert für die Variable ein, z. B. `C:\temp`.
6. Schließen Sie alle Fenster, und starten Sie das System erneut.

Zugehörige Konzepte:

- „XML-Dokumente" auf Seite 4
- „Funktionen von XML Extender" auf Seite 5
- „Lektion: XML-Dokument in einer XML-Spalte speichern" auf Seite 9
- „Lektion: XML-Dokument zusammensetzen" auf Seite 20
- „Lernprogrammübungen zu XML Extender" auf Seite 8

XML-Dokumente

Da Firmen in der Regel Daten zwischen verschiedenen Anwendungen gemeinsam nutzen, ist die Vervielfältigung und Umwandlung von Daten sowie das Exportieren oder Speichern in Formaten, die wiederum in andere Anwendungen importiert werden können, eine permanente Herausforderung. Bei vielen dieser Umwandlungsprozesse geht ein Teil der Daten verloren oder zumindest müssen die Benutzer in einem relativ aufwendigen Prozess sicherstellen, dass die Daten konsistent bleiben. Diese manuelle Prüfung kosten sowohl Zeit als auch Geld.

Anwendungsentwickler können dieses Problem unter anderem dadurch lösen, dass sie ODBC-Anwendungen *(Open Database Connectivity)* schreiben; ODBC ist eine Standard-Anwendungsprogrammierschnittstelle (API) für den Zugriff auf Daten in relationalen und nicht relationalen Datenbankverwaltungssystemen. Diese Anwendungen speichern die Daten in einem Datenbankverwaltungssystem. Von hier aus können die Daten verarbeitet und in der Form dargestellt werden, in der sie in einer anderen Anwendung benötigt werden. Datenbankanwendungen müssen so geschrieben werden, dass sie die Daten in ein für eine bestimmte Anwendung erforderliches Format umwandeln. Anwendungen ändern sich schnell 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. Eine Möglichkeit, die Daten von ihrer Präsentation zu trennen, ist notwendig, um ein geeignetes Format für den Austausch zwischen Anwendungen bereitzustellen.

Mit XML - *eXtensible Markup Language* - lässt sich dieses Problem lösen. XML ist erweiterbar, da die Sprache 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. Obwohl es nicht das einzige Datenaustauschformat ist, wird XML mittlerweile als Standard akzeptiert. Durch die Einhaltung dieses Standards können Anwendungen Daten gemeinsam nutzen, ohne sie zunächst in ein spezielles Format umwandeln zu müssen.

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. Ihre Projektverwaltungsanwendung könnte Tasks in XML exportieren, die wiederum in unveränderter Form in Ihre Kalenderanwendung übernommen werden können. In der heutigen miteinander verbundenen Welt haben Anwendungslieferanten ein starkes Interesse daran, ein XML-Austauschformat als Basisfunktion in ihre Anwendungen zu integrieren.

XML-Daten in DB2 bearbeiten

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

- Wie häufig sollen die Daten repliziert werden?
- Welche Arten von Informationen müssen 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 auslösen?

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 schneller und besser die XML-Ergebnisse abrufen, die Ihre anderen Anwendungen benötigen. Mit XML Extender nutzen Sie die Leistung von DB2 in vielen XML-Anwendungen.

Mit dem Inhalt Ihrer strukturierten XML-Dokumente in einer DB2 UDB-Datenbank können Sie strukturierte XML-Informationen mit traditionellen relationalen Daten kombinieren. Entsprechend der Anwendung können Sie auswählen, ob ganze XML-Dokumente in DB2 als benutzerdefinierte Typen für XML-Daten (XML-Datentypen) gespeichert werden sollen, oder Sie können den XML-Inhalt als Basisdatentypen in relationalen Tabellen zuordnen. Für XML-Datentypen bietet XML Extender zusätzlich zu der strukturellen Textsuche von DB2 Universal Database™ die Möglichkeit, umfangreiche Datentypen von XML-Element- oder -Attributwerten zu suchen.

XML Extender bietet in DB2 zwei Methoden zum Speichern von und Zugreifen auf XML-Daten:

XML-Spaltenmethode

Speichern von ganzen XML-Dokumenten als Spaltendaten oder extern als Datei und Extrahieren und Speichern des erforderlichen XML-Element- oder -Attributwerts in *Nebentabellen* (indexierte Untertabellen für eine sehr schnelle Suche). Durch das Speichern der Dokumente als Spaltendaten können Sie Folgendes:

- Ausführen einer Schnellsuche nach XML-Elementen oder -Attributen, die extrahiert und in Nebentabellen als SQL-Basisdatentypen gespeichert 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 Net Search Extender.

XML-Objektgruppenmethode

Zusammensetzen oder Zerlegen des Inhalts von XML-Dokumenten mit einer oder mehreren relationalen Tabellen.

Funktionen von XML Extender

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

- Verwaltungstools zur Verwaltung der Integration von XML-Daten in relationalen Tabellen
- Speicher- und Zugriffsmethoden für XML-Daten innerhalb der Datenbank
- Ein DTD-Repository (Data Type Definition; Datentypdefinition), in dem Sie DTDs zur Prüfung von XML-Daten speichern können

- Die Fähigkeit, XML-Dokumente mit einem Schema zu prüfen
- Eine DAD-Datei (Document Access Definition; Dateizugriffsdefinition) für die Zuordnung von XML-Dokumenten zu relationalen Daten
- Standordpfade zur Angabe des Standorts eines Elements oder Attributs innerhalb eines XML-Dokuments

Verwaltungstools: Die Verwaltungstools von XML Extender erleichtern das Aktivieren der Datenbank und der Tabellenspalten für XML und das Zuordnen der XML-Daten zu den relationalen DB2-Strukturen.

XML Extender stellt die folgenden Tools zur Ausführung von Verwaltungstasks zur Verfügung:

- Der Befehl **dxxadm** bietet eine Befehlszeilenoption für Verwaltungstasks.
- Mit den gespeicherten Prozeduren der XML Extender-Verwaltung können Sie Verwaltungsbefehle von einem Programm aus aufrufen.

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

XML-Spaltenmethode
: Diese Methode ermöglicht das Speichern intakter XML-Dokumente in DB2. Die XML-Spaltenmethode eignet sich gut zum Archivieren von Dokumenten. Die Dokumente werden in Spalten eingefügt, die für XML aktiviert sind, und können aktualisiert, abgerufen und durchsucht werden. Element- und Attributdaten können DB2 UDB-Tabellen (Nebentabellen) zugeordnet werden, die für eine Schnellsuche indexiert werden können.

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

DTDs: XML Extender ermöglicht Ihnen außerdem das Speichern von DTDs, der Gruppe von Deklarationen für XML-Elemente und -Attribute. Beim *Aktivieren* einer Datenbank für XML wird die DTD-Repository-Tabelle (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 werden zur Prüfung der Struktur von XML-Dokumenten verwendet.

DAD-Dateien: Sie geben an, wie strukturierte XML-Dokumente von XML Extender verarbeitet werden sollen, indem Sie eine *Dokumentzugriffsdefinitionsdatei (DAD-Datei)* verwenden. Die DAD-Datei ist ein XML-Dokument, das die XML-Dokumentstruktur einer DB2 UDB-Tabelle zuordnet. Sie verwenden eine DAD-Datei sowohl beim Speichern von XML-Dokumenten in einer Spalte als auch beim Zusammensetzen oder Zerlegen von XML-Daten. Die DAD-Datei gibt an, ob Sie Dokumente unter Verwendung der XML-Spaltenmethode speichern oder eine XML-Objektgruppe für die Zusammensetzung oder Zerlegung definieren.

Standortpfade: Ein *Standortpfad* gibt den Standort eines Elements oder Attributs innerhalb eines XML-Dokuments an. XML Extender verwendet den Standortpfad zum Navigieren in der Struktur des XML-Dokuments und zum Suchen von Elementen und Attributen.

Beispielsweise zeigt der Standortpfad von `/Order/Part/Shipment/ShipDate` auf das Element 'ShipDate', das den Elementen 'Shipment', 'Part' und 'Order' untergeordnet ist, wie das folgende Beispiel zeigt:

```
<Order>
  <Part>
    <Shipment>
      <ShipDate>
+...
```

Abb. 1 zeigt ein Beispiel eines Standortpfads und seiner Beziehung zur Struktur des XML-Dokuments.

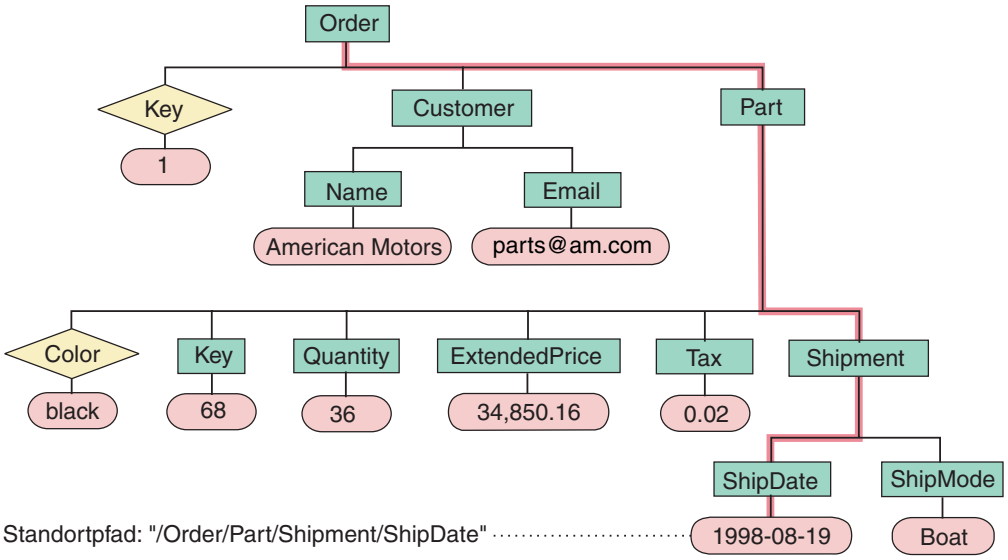


Abbildung 1. Speichern von Dokumenten als strukturierte XML-Dokumente in einer DB2 UDB-Tabelle

Der Standortpfad wird in den folgenden Situationen verwendet:

XML-Spalten

- Zur Kennzeichnung von Elementen und Attributen, die extrahiert oder aktualisiert werden sollen, wenn die benutzerdefinierten Funktionen von XML Extender verwendet werden.
- Zum Zuordnen des Inhalts eines XML-Elements oder Attributs zu einer Nebentabelle.

XML-Objektgruppen

Zum Überschreiben von Werten in der DAD-Datei aus einer gespeicherten Prozedur.

Zum Angeben des Standortpfads verwendet XML Extender eine Untermenge der *XML Path Language (XPath)*, der Sprache für die Adressierung von Teilen eines XML-Dokuments.

Weitere Informationen zu Xpath finden Sie auf der folgenden Webseite:

http://www.w3.org/TR/xpath

Zugehörige Konzepte:
- „XML-Daten in DB2 bearbeiten" auf Seite 4
- „Lektion: XML-Dokument in einer XML-Spalte speichern" auf Seite 9
- „Lektion: XML-Dokument zusammensetzen" auf Seite 20
- „Lernprogrammübungen zu XML Extender" auf Seite 8

Lernprogrammübungen zu XML Extender

Dieses Lernprogramm beschreibt die ersten Schritte bei der Verwendung von XML Extender zum Aufrufen und Ändern von XML-Daten für Ihre Anwendungen. Drei Lerneinheiten sind enthalten:
- Ein XML-Dokument in einer XML-Spalte speichern
- Ein XML-Dokument zusammensetzen
- Die Datenbank bereinigen

Anhand der 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 Lerneinheiten zur Verwaltung verwenden Sie das DB2®-Befehlsfenster mit den Verwaltungsbefehlen von XML Extender. In den Lerneinheiten zur XML-Datenverwaltung verwenden Sie die UDFs und gespeicherten Prozeduren von XML Extender. Die meisten Beispiele im weiteren Verlauf dieses Handbuchs verweisen auf die in diesem Abschnitt verwendeten Musterdaten.

Voraussetzungen

Zum Ausführen der Übungen in diesem Lernprogramm müssen die folgenden Komponenten installiert sein:
- DB2 UDB Version 8.1
- DB2 für z/OS™ Version 8
- XML Toolkit für z/OS Version 1.4
- USS-Setup
- odb2-Befehlszeile oder der Job DXXGPREP JCL

Szenario für die Lektionen

In diesen Lektionen arbeiten Sie für ACME Auto Direct, ein Unternehmen, das Automobile und LKW an Händlerniederlassungen verteilt. Sie müssen zwei Aufgaben erfüllen. Zuerst richten Sie ein System ein, in dem Bestellungen in der Datenbank SALES_DB zum Abfragen nach Vertriebsabteilung archiviert werden können. Anschließend extrahieren Sie Informationen in einer vorhandenen Bestellungsdatenbank mit dem Namen SALES_DB.

Zugehörige Konzepte:
- „Verwaltungstools für XML Extender" auf Seite 39
- „Übersicht: XML Extender-Verwaltung" auf Seite 41
- „Lektion: XML-Dokument in einer XML-Spalte speichern" auf Seite 9
- „Lektion: XML-Dokument zusammensetzen" auf Seite 20

Lektion: XML-Dokument in einer XML-Spalte speichern

XML Extender bietet eine Methode zum Speichern und Aufrufen ganzer XML-Dokumente in der Datenbank. Mit der XML-Spaltenmethode können Sie das Dokument mit XML-Dateitypen speichern, die Spalte in Nebentabellen indexieren und das XML-Dokument anschließend abfragen oder durchsuchen. Diese Speichermethode ist besonders nützlich für Archivierungsanwendungen, in denen Dokumente nicht sehr häufig aktualisiert werden.

In dieser Lektion lernen Sie die Verwendung der XML-Spaltenzugriffs- und -speichermethode.

Szenario:

Sie haben die Aufgabe, die Verkaufsdaten für die Serviceabteilung zu archivieren. Die notwendigen Verkaufsdaten sind in XML-Dokumenten gespeichert, die dieselbe DTD verwenden.

Die Serviceabteilung hat eine empfohlene Struktur für die XML-Dokumente bereitgestellt und angegeben, welche Elementdaten voraussichtlich am häufigsten abgefragt werden. Die Serviceabteilung möchte, dass die XML-Dokumente in der Tabelle SALES_TAB in der Datenbank SALES_DB gespeichert werden. Außerdem soll die Möglichkeit gegeben sein, die Dokumente schnell zu durchsuchen. Die Tabelle SALES_TAB enthält zwei Spalten mit Daten zu den einzelnen Verkäufen und eine dritte Spalte mit dem XML-Dokument. Diese Spalte hat den Namen ORDER.

Zum Speichern dieses XML-Dokuments in der Tabelle SALES_TAB führen Sie folgende Schritte aus:

1. Legen Sie die benutzerdefinierten Typen (UDTs) von XML Extender fest, in denen das XML-Dokument gespeichert werden soll, und geben an, welche XML-Elemente und -Attribute häufig abgefragt werden.
2. Richten Sie die Datenbank SALES_DB für XML ein.
3. Erstellen Sie die Tabelle SALES_TAB, und aktivieren Sie die Spalte ORDER, so dass Sie das intakte Dokument in DB2® speichern können.
4. Fügen Sie eine DTD für das XML-Dokument zur Überprüfung ein.
5. Speichern Sie das Dokument als Datentyp XMLVARCHAR.

Wenn Sie die Spalte aktivieren, definieren Sie Nebentabellen zum Indexieren für die strukturelle Suche des Dokuments in einer Dokumentzugriffsdefinitionsdatei (DAD), einem XML-Dokument, das die Struktur der Nebentabellen festlegt.

Die Tabelle SALES_TAB wird in Tabelle 1 beschrieben. Die XML-Spalte, die für XML aktiviert werden soll (ORDER), ist kursiv dargestellt.

Tabelle 1. Tabelle SALES_TAB

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

Prozeduren und Beispiele:

Verwenden Sie für dieses Lernprogramm eine Gruppe von Prozeduren, um Ihre Umgebung einzurichten und die Schritte in diesen Lektionen auszuführen. Diese Prozeduren befinden sich im Verzeichnis *dxx_install_verz*/samples/db2xml/cmd. Dabei ist *dxx_install_verz* das Verzeichnis, in dem Sie die XML Extender-Dateien installiert haben.

Diese Prozeduren sind:

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

getstart_prep.cmd
Bindet die Datenbank an die gespeicherten Prozeduren von XML Extender und aktiviert die Datenbank für XML Extender.

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 Nebentabellen 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_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.

Speichern des Dokuments planen:

Bevor Sie XML Extender zum Speichern von Dokumenten verwenden, müssen Sie die folgenden Voraussetzungen erfüllen:
- Sie müssen die Struktur eines XML-Dokuments verstehen.
- Sie müssen den benutzerdefinierten XML-Typ festlegen, in dem das XML-Dokument gespeichert werden soll.
- Sie müssen die XML-Elemente und -Attribute festlegen, die häufig von der Serviceabteilung durchsucht werden, so dass deren Inhalt in Nebentabellen gespeichert und indexiert werden kann, um eine höhere Leistung zu erzielen.

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

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.

In dieser Lektion wird die Beispiel-DTD bereitgestellt, mit deren Hilfe Sie die Struktur von XML-Dokumenten verstehen und prüfen können.

XML-Datentyp für die XML-Spalte festlegen:

XML Extender bietet benutzerdefinierte XML-Typen, die Sie zum Definieren einer Spalte für die Speicherung von XML-Dokumenten verwenden können. Diese Datentypen lauten wie folgt:

XMLVARCHAR
Wird für Dokumente verwendet, deren Größe unter drei Kilobyte liegt und die in DB2 gespeichert werden. Die maximale Größe von XMLVARCHAR-Dokumenten beträgt 32672 Byte.

XMLCLOB
Wird für Dokumente verwendet, deren Größe über drei Kilobyte liegt und die in DB2 gespeichert werden. Die maximale Dokumentengröße beträgt zwei Gigabyte.

XMLFILE
Wird für außerhalb von DB2 gespeicherte Dokumente verwendet.

In dieser Lektion speichern Sie ein kleines Dokument in DB2 und verwenden 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 am häufigsten durchsucht und extrahiert werden oder welche am aufwendigsten zu durchsuchen sind. Die Serviceabteilung wird häufig den Bestellschlüssel, den Kundennamen, den Preis und das Versanddatum einer Bestellung abfragen. Hierzu ist ein schneller Suchlauf erforderlich. 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
Kundenname	/Order/Customer/Name
Preis	/Order/Part/ExtendedPrice
Versanddatum	/Order/Part/Shipment/ShipDate

XML-Dokument den Nebentabellen zuordnen:

Um Ihre XML-Dokumente einer Nebentabelle zuzuordnen, müssen Sie eine DAD-Datei für die XML-Spalte erstellen. Die DAD-Datei wird zum Speichern des XML-Dokuments in DB2 verwendet. Diese Datei ordnet auch den Inhalt der XML-Elemente und -Attribute den für die Indexierung verwendeten DB2 UDB-Nebentabellen zu, was die Suchleistung verbessert.

Nach dem Kennzeichnen der zu durchsuchenden Elemente und Attribute legen Sie fest, wie diese in den Nebentabellen organisiert werden sollen, wie viele Tabellen verwendet werden und welche Spalten sich in welcher Tabelle befinden sollen. Organisieren Sie die Nebentabellen durch Ablegen ähnlicher Informationen in der gleichen Tabelle. Die Dokumentstruktur wird auch dadurch festgelegt, ob der Standortpfad der Elemente mehr als einmal im Dokument wiederholt werden kann. Im Dokument kann das Element "part" beispielsweise mehrfach wiederholt werden. Die Elemente "price" und "date" können somit mehrfach vorkommen. Elemente, die mehrfach vorkommen können, müssen jeweils in ihren eigenen Nebentabellen stehen.

Sie müssen auch festlegen, welche DB2 UDB-Basistypen die Element- und Attributwerte verwenden sollen. Dies wird über das Format der Daten ermittelt.
- Wenn es sich bei den Daten um Text handelt, verwenden Sie VARCHAR.
- Wenn es sich bei den Daten um eine ganze Zahl handelt, verwenden Sie INTEGER.
- Wenn es sich bei den Daten um ein Datum handelt und Sie Bereichssuchen durchführen wollen, verwenden Sie DATE.

In diesem Lernprogramm sind die Elemente und Attribute entweder ORDER_SIDE_TAB, PART_SIDE_TAB oder SHIP_SIDE_TAB zugeordnet. Die nachfolgenden Tabellen zeigen, welcher Tabelle die einzelnen Elemente oder Attribute zugeordnet sind.

ORDER_SIDE_TAB

Spaltenname	Datentyp	Standortpfad	Mehrfaches Vorkommen?
ORDER_KEY	INTEGER	/Order/@Key	Nein
CUSTOMER	VARCHAR(16)	/Order/Customer/Name	Nein

PART_SIDE_TAB

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

SHIP_SIDE_TAB

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

Datenbank SALES_DB erstellen:

In dieser Task erstellen Sie eine Beispieldatenbank und aktivieren die Datenbank für XML.

Gehen Sie wie folgt vor, um die Datenbank zu erstellen:

1. Stellen Sie sicher, dass der Datenbankserver vom DB2-Administrator aktiviert wurde.
2. Wechseln Sie in das Verzeichnis `dxx_install_verz/samples/db2xml/cmd`, wobei 'dxx_install_verz' das Verzeichnis angibt, in dem Sie die XML Extender-Dateien installiert haben. Die Beispieldateien enthalten Verweise auf Dateien, die absolute Pfadnamen verwenden. Prüfen Sie die Beispieldateien und ändern Sie diese Werte in Ihren Verzeichnispfaden.
3. Öffnen Sie auf Windows®-Plattformen ein DB2 UDB-Befehlsfenster, indem Sie Folgendes eingeben:

   ```
   DB2CMD
   ```

4. Führen Sie den Befehl **getstart_db** aus.

Server aktivieren:

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

- Erstellen benutzerdefinierter Typen (UDTs), benutzerdefinierter Funktionen (UDFs) und gespeicherter Prozeduren.
- Erstellen der Steuertabellen und Füllen dieser Tabellen mit den erforderlichen Metadaten für XML Extender.
- Erstellen des DB2XML-Schemas und Zuordnen der erforderlichen Berechtigungen.

Gehen Sie wie folgt vor, um die Datenbank für XML zu aktivieren:

Verwenden Sie eine der folgenden Methoden, um die Datenbank zu aktivieren.

Führen Sie die folgende Prozedur aus:

```
getstart_prep.cmd
```

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

```
dxxadm enable_dbSALES_DB
```

XML-Spalte aktivieren und Dokument speichern:

In dieser Lektion aktivieren Sie eine Spalte für XML Extender und speichern ein XML-Dokument in der Spalte. Für diese Tasks führen Sie folgende Schritte aus:

1. Speichern der DTD im DTD-Repository.
2. Erstellen einer DAD-Datei für die XML-Spalte.
3. Erstellen der Tabelle SALES_TAB.
4. Hinzufügen der Spalte vom Typ XML.
5. Aktivieren der XML-Spalte.
6. Anzeigen der Spalte und der Nebentabellen.
7. Indexieren der Nebentabellen für die strukturelle Suche.
8. Speichern des XML-Dokuments.
9. Abfragen des XML-Dokuments.

DTD im DTD-Repository speichern:

Sie können mit einer DTD die XML-Daten in einer XML-Spalte überprüfen. XML Extender erstellt eine Tabelle in der für XML aktivierten Datenbank mit dem Namen DTD_REF. Die Tabelle wird als DTD-Repository bezeichnet und steht zum Speichern von DTDs zur Verfügung. Wenn Sie XML-Dokumente überprüfen, müssen Sie die DTD in diesem Repository speichern. Die DTD für diese Lektion befindet sich im Verzeichnis

dxx_install_verz/samples/db2xml/dtd/getstart.dtd

Dabei ist *dxx_install_verz* das Verzeichnis, in dem DB2 XML Extender installiert wurde.

- Geben Sie den SQL-Befehl INSERT in einer einzigen DB2-Befehlszeile ein:

```
DB2 CONNECT TO SALES_DB
INSERT into DB2XML.DTD_REF values
('dxx_install_verz/samples/db2xml/dtd/getstart.dtd',
 DB2XML.XMLClobFromFile
('dxx_install_verz/samples/db2xml/dtd/getstart.dtd'),
 0, 'user1',   'user1', 'user1')
```

- Führen Sie die folgende Befehlsdatei aus, um die DTD einzufügen:

```
getstart_insertDTD.cmd
```

DAD-Datei für die XML-Spalte erstellen:

In diesem Abschnitt wird erläutert, wie Sie eine DAD-Datei für die XML-Spalte erstellen. In der DAD-Datei geben Sie an, dass die Zugriffs- und Speichermethode, die Sie verwenden, die XML-Spalte ist. In der DAD-Datei definieren Sie die Tabellen und Spalten zum Indexieren.

In den folgenden Schritten werden die Elemente in der DAD als *Befehle* und die Elemente Ihrer XML-Dokumentstruktur als *Elemente* bezeichnet. Ein Beispiel für eine DAD-Datei ähnlich der von Ihnen zu erstellenden finden Sie in dxx_install_verz/samples/db2xml/dad/getstart_xcolumn.dad . Dieses Beispiel weist einige kleine Unterschiede gegenüber der in den folgenden Schritten generierten Datei auf. Wenn Sie das Beispiel für die Lektion verwenden, können die Dateipfade für Ihre Umgebung anders sein; und der Wert von <validation> ist auf NO statt auf YES gesetzt.

Gehen Sie wie folgt vor, um eine DAD-Datei für die Verwendung mit der XML-Spalte zu erstellen:

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

   Bei allen in der DAD-Datei verwendeten Befehlen muss zwischen Groß- und Kleinbuchstaben unterschieden werden.
2. Erstellen Sie die Kopfdaten der DAD-Datei, mit der XML- und der DOCTYPE-Deklaration.

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "/dxx_install_verz/samples/DB2XML/dtd/dad.dtd ">
   ```

   Die DAD-Datei ist ein XML-Dokument und erfordert XML-Deklarationen.
3. Fügen Sie Anfangs- und Endebefehle (<DAD> und </DAD>) für das Dokument ein. Alle weiteren Befehle befinden sich innerhalb dieser Befehle.
4. Fügen Sie Anfangs- und Endebefehle (<DTDID> und </DTDID>) mit einer DTD-ID ein, um eine DTD anzugeben, falls das Dokument geprüft wird:

   ```
   <dtdid>dxx_install_verz/samples/db2xml/dtd/getstart.dtd</dtdid>
   ```

Prüfen Sie, dass diese Zeichenfolge mit dem Wert übereinstimmt, der als erster Parameterwert verwendet wird, wenn die DTD in die DTD-Repositorytabelle eingefügt wird. Wenn Sie mit einem anderen Laufwerk arbeiten, kann der Pfad, den Sie für die DTD-ID verwenden, beispielsweise ein anderer als in der Zeichenfolge sein, die Sie in die DTD-Referenztabelle eingefügt haben.

5. Fügen Sie Anfangs- und Endebefehle (<validation> und </validation>) und das Schlüsselwort YES bzw. NO ein, um festzulegen, ob XML Extender die XML-Dokumentstruktur mit der in der DTD-Referenztabelle eingefügten DTD überprüfen soll. Beispiel:

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

   Der Wert von <validation> kann Groß- und Kleinbuchstaben enthalten.

6. Fügen Sie Anfangs- und Endebefehle (<Xcolumn> und </Xcolumn>) ein, um anzugeben, dass die Speichermethode die XML-Spalte ist.

7. Erstellen Sie Nebentabellen. Führen Sie für jede Nebentabelle, die Sie erstellen wollen, folgende Schritte aus:

   a. Fügen Sie für jede Nebentabelle, die generiert werden soll, Anfangs- und Endebefehle (<table> und </table>) ein. Geben Sie den Namen der Nebentabelle in doppelten Anführungszeichen an, und verwenden Sie hierbei das Attribut ″name=″. Beispiel:

      ```
      <Xcolumn>
      <table name="order_side_tab">
      </table>
      <table name="part_side_tab">
      </table>
      <table name="ship_side_tab">
      </table>
      </Xcolumn>
      ```

   b. Fügen Sie innerhalb der Tabellenbefehle jeweils einen Befehl `<column>` für jede Spalte ein, die die Nebentabelle enthalten soll. Jede Spalte hat vier Attribute: name (Name), type (Typ), path (Pfad) und multi_occurrence (mehrfaches Vorkommen).

      name Gibt den Namen der Spalte an, die in der Nebentabelle erstellt wird.

      type Gibt den Datentyp in der Nebentabelle für jedes indexierte Element oder Attribut an.

      path Gibt den Standortpfad im XML-Dokument für jedes zu indexierende Element oder Attribut an.

      multi_occurrence
      Gibt an, ob das Element oder Attribut, auf das durch das Pfadattribut verwiesen wird, mehr als einmal im XML-Dokument vorkommen kann. Die möglichen Werte für *multi_occurrence* sind *YES* oder *NO*. Wenn der Wert *NO* ist, können Sie mehr als einen Spaltenbefehl in der Nebentabelle aufführen. Wenn der Wert *YES* ist, können Sie nur eine Spalte in der Nebentabelle aufführen.

```
<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>
```

8. Stellen Sie sicher, dass Sie über die erforderlichen Endebefehle verfügen:
   - Einen Endebefehl </Xcolumn> nach dem letzten Befehl </table>.
   - Einen Endebefehl </DAD> nach dem Befehl </Xcolumn>.
9. Speichern Sie die Datei unter dem folgenden Namen:

   ```
   getstart_xcolumn.dad
   ```

Sie können die eben erstellte Datei mit der Beispieldatei `dxx_install_verz/samples/db2xml/dad/getstart_xcolumn.dad` vergleichen. Diese Datei ist eine Arbeitskopie der zum Aktivieren der XML-Spalte und zum Erstellen der Nebentabellen erforderlichen DAD-Datei. Die Beispieldateien enthalten Verweise auf Dateien, die absolute Pfadnamen verwenden. Prüfen Sie die Beispieldateien und ändern Sie diese Werte in Ihren Verzeichnispfaden.

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.

Gehen Sie wie folgt vor, um die Tabelle zu erstellen:

Geben Sie die folgende Anweisung CREATE TABLE ein, indem Sie eine der folgenden Methoden verwenden:

- Geben Sie die folgenden DB2 UDB-Befehle ein:

  ```
  DB2 CONNECT TO SALES_DB

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

- Führen Sie die folgende Befehlsdatei aus, um die Tabelle zu erstellen:

  ```
  getstart_createTabCol.cmd
  ```

Spalte vom Typ XML hinzufügen:

Fügen Sie eine neue Spalte zur Tabelle SALES_TAB hinzu. Diese Spalte enthält das intakte XML-Dokument, das Sie zuvor erstellt haben. Es muss ein benutzerdefinierter XML-Typ sein. XML Extender bietet mehrere Datentypen. In dieser Lektion speichern Sie das Dokument als XMLVARCHAR.

Gehen Sie wie folgt vor, um die Spalte vom Typ XML hinzuzufügen:

Führen Sie die SQL-Anweisung ALTER TABLE aus, indem Sie eine der folgenden Methoden verwenden:

- Geben Sie die folgende SQL-Anweisung ein:

  ```
  DB2 ALTER TABLE SALES_TAB ADD ORDER DB2XML.XMLVARCHAR
  ```
- Führen Sie die folgende Befehlsdatei aus, um die Tabelle zu ändern:

  ```
  getstart_alterTabCol.cmd
  ```

XML-Spalte aktivieren:

Nachdem Sie die Spalte vom Typ XML erstellt haben, aktivieren Sie sie für XML Extender. Wenn Sie die Spalte aktivieren, liest XML Extender die DAD-Datei und erstellt die Nebentabellen. Vor dem Aktivieren der Spalte müssen Sie folgende Aufgaben durchführen:
- Festlegen, ob eine Standardsicht der XML-Spalte erstellt werden soll, die das XML-Dokument, verknüpft mit den Spalten der Nebentabelle, enthält. Sie können beim Abfragen des XML-Dokuments die Standardsicht verwenden. In dieser Lektion geben Sie eine 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 Nebentabellen der Anwendungstabelle zuordnet, angegeben werden soll. Wenn Sie keinen Primärschlüssel angeben, fügt XML Extender der Anwendungstabelle und den Nebentabellen die Spalte DXXROOT_ID hinzu.

  Die Spalte ROOT_ID wird als Schlüssel verwendet, um die Anwendungs- und Nebentabellen zu verbinden. Sie gibt XML Extender die Möglichkeit, die Nebentabellen 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. XML Extender verwendet die angegebene Spalte als ROOT_ID und fügt die Spalte den Nebentabellen hinzu.
- Festlegen, ob Sie einen Tabellenbereich angeben oder den Standardtabellenbereich verwenden wollen. In dieser Lektion verwenden Sie den Standardtabellenbereich.

Gehen Sie wie folgt vor, um die Spalte für XML zu aktivieren:

Führen Sie den Befehl **dxxadm enable_column** aus, indem Sie eine der folgenden Methoden verwenden:

Befehlszeile:
- Geben Sie folgenden Befehl ein:

  ```
  dxxadm enable_column SALES_DB SALES_TAB ORDER getstart_xcolumn.dad
          -v SALES_ORDER_VIEW -r INVOICE_NUM
  ```
- Führen Sie die folgende Befehlsdatei aus, um die Spalte zu aktivieren:

  ```
  getstartenableCol.cmd
  ```

XML Extender erstellt die Nebentabellen mit der Spalte INVOICE_NUM und erstellt die Standardsicht.

Wichtig: Ändern Sie die Nebentabellen nicht. Aktualisierungen an den Nebentabellen sollten nur über Aktualisierungen am XML-Dokument selbst vorgenommen werden. XML Extender aktualisiert die Nebentabellen automatisch, wenn Sie das XML-Dokument in der XML-Spalte aktualisieren.

Spalten und Nebentabellen anzeigen:

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

Gehen Sie wie folgt vor, um die XML-Spalte und die Spalten der Nebentabelle anzuzeigen:

Geben Sie von der Befehlszeile aus die folgende SQL-Anweisung SELECT ein:

```
SELECT * FROM SALES_ORDER_VIEW
```

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

Nebentabellen für die strukturelle Suche indexieren:

Das Erstellen von Indizes zu Nebentabellen ermöglicht ein schnelles strukturelles Durchsuchen des XML-Dokuments. In diesem Abschnitt erstellen Sie Indizes zu Schlüsselspalten in den Nebentabellen, 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 Nebentabellen

Spalte	Nebentabelle
ORDER_KEY	ORDER_SIDE_TAB
CUSTOMER	ORDER_SIDE_TAB
PRICE	PART_SIDE_TAB
DATE	SHIP_SIDE_TAB

Gehen Sie wie folgt vor, um die Nebentabellen zu indexieren:

Führen Sie die folgenden SQL-Befehle CREATE INDEX aus, wobei Sie eine der folgenden Methoden verwenden:

- Geben Sie die folgenden 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)
```

- Führen Sie die folgende Befehlsdatei aus, um die Indizes zu erstellen:

```
getstart_createIndex.cmd
```

XML-Dokument speichern:

Sie haben eine Spalte aktiviert, die das XML-Dokument enthalten kann, sowie die Nebentabellen indexiert. Nun können Sie das Dokument mit den Funktionen von XML Extender speichern. Beim Speichern von Daten in einer XML-Spalte verwenden Sie entweder die Standardumsetzungsfunktionen oder XML Extender-UDFs. Da Sie ein Objekt des Basistyps VARCHAR in einer Spalte des XML-UDT XMLVARCHAR speichern, verwenden Sie die Standardumsetzungsfunktion.

Gehen Sie wie folgt vor, um das XML-Dokument zu speichern:

1. Öffnen Sie das XML-Dokument `dxx_install_verz/samples/db2xml/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 Standard abweichende Laufwerks- und Verzeichnisstruktur verwenden, müssen Sie den Pfad in der DOCTYPE-Deklaration ändern, damit die Übereinstimmung mit der Verzeichnisstruktur gegeben ist.
2. Führen Sie den SQL-Befehl INSERT aus, indem Sie eine der folgenden Methoden verwenden:
   - Geben Sie den folgenden SQL-Befehl INSERT ein:

     ```
     DB2 INSERT INTO SALES_TAB (INVOICE_NUM, SALES_PERSON, ORDER) VALUES
     ('123456', 'Sriram Srinivasan', DB2XML.XMLVarcharFromFile
     ('dxx_install_verz/samples/db2xml/
     /xml/getstart.xml'))
     ```
   - Führen Sie die folgende Befehlsdatei aus, um das Dokument zu speichern:

     ```
     getstart_insertXML.cmd
     ```

Überprüfen Sie, ob die Tabellen aktualisiert wurden. Führen Sie über die Befehlszeile die folgenden SELECT-Anweisungen für die Tabellen aus.

```
SELECT * FROM SALES_TAB
SELECT * FROM PART_SIDE_TAB
SELECT * FROM ORDER_SIDE_TAB
SELECT * FROM SHIP_SIDE_TAB
```

XML-Dokument abfragen:

Sie können das XML-Dokument mit einer direkten Abfrage für die Nebentabellen durchsuchen. In diesem Schritt suchen Sie nach allen Bestellungen mit einem Preis über 2500,00.

Gehen Sie wie folgt vor, um die Nebentabellen abzufragen:

Führen Sie die SQL-Anweisung SELECT aus, indem Sie eine der folgenden Methoden verwenden:

- Geben Sie die folgende SQL-Anweisung SELECT 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"
  ```
- Führen Sie die folgende Befehlsdatei aus, um das Dokument zu durchsuchen:

  ```
  getstart_queryCol.cmd
  ```

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

Sie haben hiermit das Lernprogramm ″Erste Schritte″ zum Speichern von XML-Dokumenten in DB2 UDB-Tabellen abgeschlossen. Beispiel:

```
SALES_PERSON
--------------------

Sriram Srinivasan
```

Zugehörige Konzepte:

- „Einführung in XML Extender" auf Seite 3
- „Lektion: XML-Dokument zusammensetzen" auf Seite 20
- „Lernprogrammübungen zu XML Extender" auf Seite 8

Lektion: XML-Dokument zusammensetzen

In dieser Lektion lernen Sie das Zusammensetzen eines XML-Dokuments aus bestehenden DB2-Daten.

Szenario:

Sie haben die Aufgabe, Informationen in einer vorhandenen Bestellungsdatenbank mit dem Namen SALES_DB zu erfassen und angeforderte Informationen 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 das XML-Dokument `getstart.xml` aus den Daten in diesen Tabellen zusammen.

Um ein XML-Dokument zusammenzusetzen, planen und erstellen Sie eine DAD-Datei, die Spalten aus den zugehörigen Tabellen einer XML-Dokumentstruktur zuordnet, die einen Bestellungsdatensatz enthält. 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 mit einem Stern 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)
TAX *	REAL

Spaltenname	Datentyp
ORDER_KEY	INTEGER

SHIP_TAB

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

Planung:

Bevor Sie XML Extender zum Zusammensetzen Ihrer Dokumente verwenden, müssen Sie die Struktur des XML-Dokuments definieren und festlegen, wie die Struktur der Daten in der Datenbank dieser entspricht. Dieser Abschnitt bietet einen Überblick zu der von der Serviceabteilung angeforderten XML-Dokumentstruktur und zur DTD, mit der Sie die Struktur des XML-Dokuments definieren. Außerdem wird in diesem Abschnitt gezeigt, wie dieses Dokument den Spalten zugeordnet wird, die die zum Füllen der Dokumente verwendeten Daten enthalten.

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 Serviceabteilung möchte, dass die Dokumentstruktur intuitiv und flexibel ist, wobei die Elemente die Daten beschreiben und nicht die Struktur des Dokuments. (Der Name des Kunden soll beispielsweise in einem Element mit dem Namen „customer" gespeichert sein statt in einem Abschnitt.)

Nachdem Sie die Dokumentstruktur entworfen haben, erstellen Sie zur Beschreibung der Struktur des XML-Dokuments eine DTD. In dieser Lerneinheit wird Ihnen ein XML-Dokument und eine DTD zur Verfügung gestellt. Mit den Regeln der DTD und der hierarchischen Struktur des XML-Dokuments können Sie eine hierarchische Zuordnung Ihrer Daten, wie in Abb. 2 auf Seite 22 gezeigt, erstellen.

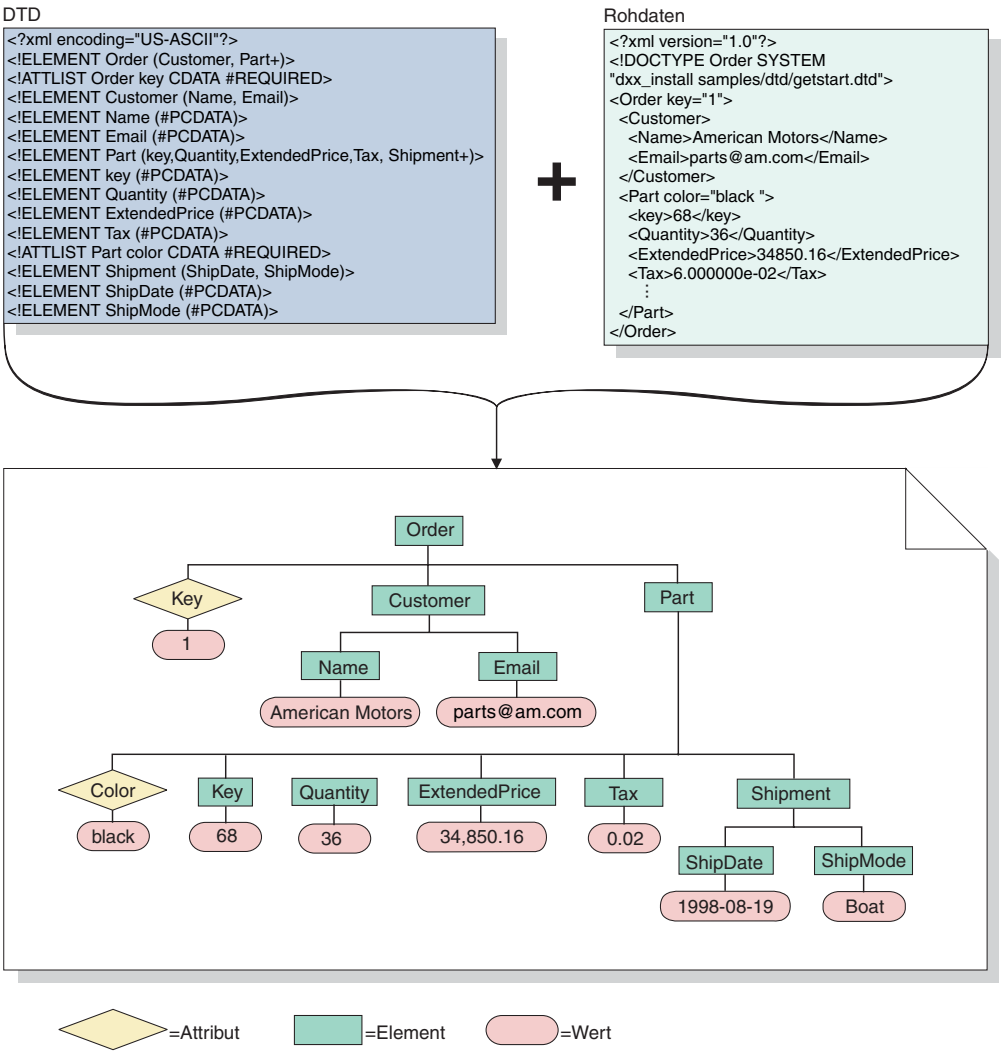


Abbildung 2. Die hierarchische Struktur der DTD und des XML-Dokuments

Beziehung XML-Dokument und Datenbank zuordnen:

Nachdem Sie die Struktur entworfen und die DTD erstellt haben, müssen Sie zeigen, wie die Struktur der Dokumente sich zu den DB2 UDB-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. 3 auf Seite 23 gezeigt.

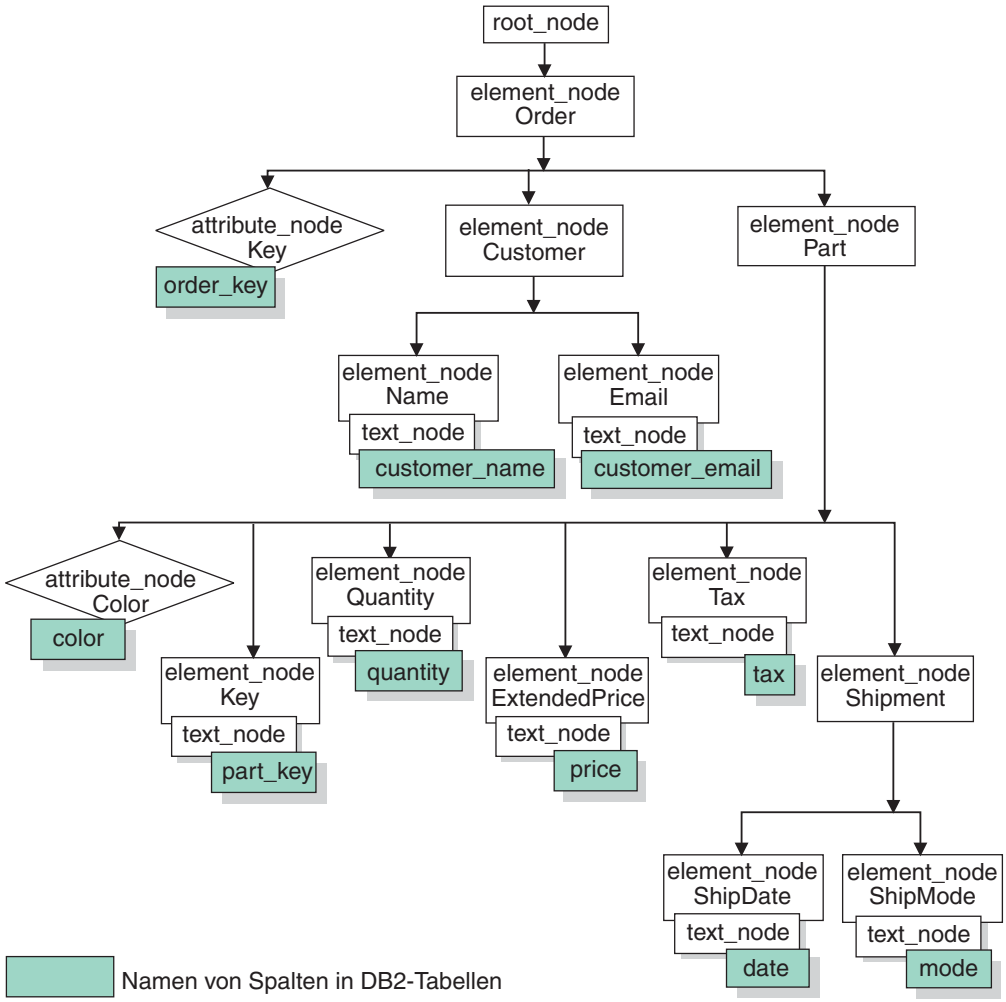


Abbildung 3. Relationalen Tabellenspalten zugeordnetes XML-Dokument

In dieser Abbildung werden Knoten verwendet, um Elemente, Attribute und Text innerhalb der XML-Dokumentstruktur anzuzeigen. Diese Knoten werden in der DAD-Datei verwendet und in späteren Schritten detailliert erläutert.

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. 3), 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.

Prozeduren und Beispiele:

In dieser Lerneinheit wird Ihnen eine Reihe von Prozeduren zur Verfügung gestellt, mit denen Sie Ihre Umgebung einrichten können. `Diese Prozeduren befinden sich im Verzeichnis dxx_install_verz/samples/db2xml/xml (dabei ist dxx_install_verz das Verzeichnis, in dem Sie die XML Extender-Dateien installiert haben)`

Die Prozeduren sind:

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

getstart_prep.cmd
Bindet die Datenbank an die gespeicherten Prozeduren von 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.

Lektionsumgebung einstellen:

In diesem Abschnitt bereiten Sie die Datenbank für die Verwendung mit 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.

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.

Gehen Sie wie folgt vor, um die Datenbank zu erstellen:

1. Wechseln Sie in das Verzeichnis *dxx_install_verz*`/samples/db2xml/xml`. Dabei ist *dxx_install_verz* das Verzeichnis, in dem Sie die XML Extender-Dateien installiert haben. Die Beispieldateien enthalten Verweise auf Dateien, die absolute Pfadnamen verwenden. Prüfen Sie die Beispieldateien und ändern Sie diese Werte in Ihren Verzeichnispfaden.
2. Öffnen Sie das DB2 UDB-Befehlsfenster:
   ```
   DB2CMD
   ```
3. Führen Sie die Befehlsdatei zum Erstellen einer Datenbank aus, indem Sie eine der folgenden Methoden verwenden:

   Geben Sie den folgenden Befehl ein:
   ```
   getstart_db.cmd
   ```

Datenbank aktivieren:

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

- Erstellen der benutzerdefinierten Typen (UDTs), der benutzerdefinierten Funktionen (UDFs) und der gespeicherten Prozeduren.
- Erstellen der Steuertabellen und Füllen dieser Tabellen mit den erforderlichen Metadaten für XML Extender.
- Erstellen des DB2XML-Schemas und Zuordnen der 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.

Verwenden Sie eine der folgenden Methoden, um die Datenbank für XML zu aktivieren:

Führen Sie die folgende Prozedur aus, um die Datenbank SALES_DB zu aktivieren:

```
getstart_prep.cmd
```

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

```
dxxadm enable_db SALES_DB
```

DAD-Datei für die XML-Objektgruppe erstellen:

Da die Daten bereits in verschiedenen Tabellen vorhanden sind, erstellen Sie eine XML-Objektgruppe, die die Tabellen dem XML-Dokument zuordnet. Sie definieren die Objektgruppe, indem Sie eine DAD-Datei erstellen.

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 Beispiel für eine DAD-Datei ähnlich der Datei, die Sie erstellen, finden Sie im Verzeichnis *dxx_install_verz*`/samples/db2xml/dad/getstart_xcollection.dad`.

Dieses Beispiel weist einige kleine Unterschiede gegenüber der in den folgenden Schritten generierten Datei auf. Wenn Sie das Beispiel für die Lektion verwenden, beachten Sie, dass die Dateipfade für Ihre Umgebung anders sein können und Sie die Beispieldatei möglicherweise aktualisieren müssen.

Gehen Sie wie folgt vor, um die DAD-Datei zum Zusammensetzen eines XML-Dokuments zu verwenden:

1. Öffnen Sie vom Verzeichnis `dxx_install_verz/samples/db2xml/xml` aus einen Texteditor, und erstellen Sie eine Datei mit dem Namen `getstart_xcollection.dad`.
2. Erstellen Sie die DAD-Kopfdaten mit dem folgenden Text:

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "dxx_install_verz/samples/db2xml/dtd/dad.dtd">
   ```

   Wechseln Sie vom Verzeichnis `dxx_install_verz` in das Verzeichnis, in dem Sie DB2 XML Extender installiert 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 XML Extender die XML-Dokumentstruktur überprüft, wenn Sie eine DTD in die DTD-Repository-Tabelle einfügen. In dieser Lektion ist keine DTD erforderlich, und der Wert ist **NO**.

   ```
   <validation>NO</validation>
   ```

   Der Wert für die Befehle <validation> kann in Groß- und Kleinbuchstaben eingegeben werden.

5. Verwenden Sie die Befehle <Xcollection> </Xcollection> zum Definieren der Zugriffs- und Speichermethode als XML-Objektgruppe. Die Zugriffs- und Speichermethoden geben an, dass die XML-Daten in einer Objektgruppe von DB2 UDB-Tabellen gespeichert werden.

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

6. Geben Sie nach dem Befehl <Xcollection> 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. Geben Sie die folgende Anweisung ein:

   ```
   <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(db2xml.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> </Xcollection>
   ```

   Diese SQL-Anweisung verwendet die folgenden Richtlinien bei Verwendung der SQL-Zuordnung. In Abb. 3 auf Seite 23 wird die Dokumentstruktur aufgeführt.

   - Spalten werden von oben nach unten angegeben entsprechend der Hierarchie der XML-Dokumentstruktur. Die Spalten für die Elemente ″order″ und ″customer″ sind beispielsweise die ersten, die für das Element ″part″ sind die zweiten und die für das Element ″shipment″ sind die dritten.
   - Die Spalten für einen sich wiederholenden Abschnitt bzw. einen sich nicht wiederholenden Abschnitt der Schablone, für die Daten aus der Datenbank erforderlich sind, werden gruppiert. Jede Gruppe hat eine Objekt-ID-Spalte: ORDER_KEY, PART_KEY, and 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 einer Anweisung ORDER BY aufgelistet. Die Spalten in ORDER BY werden nicht über ein Schema und einen Tabellennamen qualifiziert. Außerdem entsprechen Sie den Spaltennamen in der Klausel SELECT.

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
   "dxx_install_verz/samples/db2xml/dtd/getstart.dtd"</doctype>
   ```

9. Definieren Sie das Stammelement des XML-Dokuments mit den Befehlen <root_node></root_node>. Geben Sie innerhalb von root_node 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 UDB-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.

    Abb. 3 auf Seite 23 zeigt die hierarchische Struktur des XML-Dokuments und der DB2 UDB-Tabellenspalten an und gibt an, welche Arten von Knoten verwendet werden. Die schraffierten Felder kennzeichnen die Namen der DB2 UDB-Tabellenspalten, aus denen die Daten zum Zusammensetzen des XML-Dokuments extrahiert werden.

    Gehen Sie wie folgt vor, um Knotentypen einzeln nacheinander hinzuzufügen:

    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> <!-- Ende Shipment -->
         </element_node> <!-- Ende Part -->
       </element_node> <!-- Ende Order -->
       </root_node>
       ```

Das untergeordnete Element <Shipment> enthält das Attribut multi_occurrence=YES. 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 des entsprechenden element_node verschachtelt. Die hinzugefügten attribute_nodes sind in Fettdruck 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> <!-- Ende Part -->
</element_node> <!-- Ende 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 Erstellen des Dokuments aus DB2 UDB 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> <!-- Ende Shipment -->
 </element_node> <!-- Ende Part -->
</element_node> <!-- Ende 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>. Die im Befehl <column/> definierten Spalten müssen sich in der Klausel <SQL_stmt> SELECT befinden.

```
<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> <!-- Ende Shipment -->
</element_node> <!-- Ende Part -->
</element_node> <!-- Ende Order -->
</root_node>
```

11. Stellen Sie sicher, dass Sie über die erforderlichen Endebefehle verfügen:
    - Einen Endebefehl </root_node> nach dem letzten Befehl </element_node>.
    - Einen Endebefehl </Xcollection> nach dem Befehl </root_node>.
    - Einen Endebefehl </DAD> nach dem Befehl </Xcollection>.
12. Speichern Sie die Datei unter dem Namen `getstart_xcollection.dad`.

Die erstellte Datei ist mit der Beispieldatei *dxx_install_verz*`/samples/db2xml/dad/getstart_xcollection.dad` zu vergleichen. Diese Datei ist eine Arbeitskopie der zum Zusammensetzen des XML-Dokuments erforderlichen DAD-Datei. The sample file contains location paths and file path names that might need to be changed to match your environment to be run successfully.

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 Objektgruppennamens (an Stelle des Namens der DAD-Datei) bei der Ausführung gespeicherter Prozeduren kann somit die Leistung verbessert werden. In diesen Lektionen aktivieren Sie die Objektgruppe nicht.

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.

Gehen Sie wie folgt vor, um das XML-Dokument zusammenzusetzen:

1. Verwenden Sie eine der folgenden Methoden, um die gespeicherte Prozedur dxxGenXML aufzurufen:

   Geben Sie den folgenden Befehl ein:

   ```
   getstart_stp.cmd
   ```

   Die gespeicherte Prozedur setzt das XML-Dokument zusammen und speichert es in der Tabelle RESULT_TAB.

   Wenn Sie XML Extender in einer partitionierten DB2 Enterprise Server Edition-Umgebung ausführen, stellen Sie sicher, dass Sie entweder eine Ergebnistabelle mit einem qualifizierten Partitionierungsschlüssel erstellt haben oder dass Sie die Ergebnistabelle in einem Tabellenbereich erstellt haben, der sich mit einem einzelnen Knoten in einer Knotengruppe befindet.

Beispiele zu den gespeicherten Prozeduren, die in diesem Schritt verwendet werden können, finden Sie in den folgenden Dateien:

- *dxx_install_verz*/samples/db2xml/c/tests2x.sqc zeigt, wie die gespeicherte Prozedur über das eingebettete SQL aufgerufen und die ausführbare Datei tests2x generiert wird, die von getstart_stp. cmd verwendet wird. .
- *dxx_install_verz*/samples/db2xml/cli/sql2xml.c dxxsamples/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 eine der folgenden Methoden, um die die XML Extender-Abfragefunktion Content() aufzurufen:
   - Geben Sie die folgenden Befehle ein:

     ```
     DB2 CONNECT TO SALES_DB

     DB2 SELECT DB2XML.Content(DB2XML.xmlVarchar(doc),
        'dxx_install_verz/samplesdb2xml/cmd/xml/getstart.xml
          ') FROM RESULT_TAB
     ```

   - Führen Sie die folgende Befehlsdatei aus, um die Datei zu exportieren:

     ```
     getstart_exportXML.cmd
     ```

Tipp: In diesem Schritt wird gezeigt, wie ein oder mehrere zusammengesetzte XML-Dokumente über die Ergebnismengenfunktion der gespeicherten Prozeduren von DB2 UDB generieren. Über die Ergebnismengenfunktion können Sie mehrere Zeilen abrufen, um mehr als ein Dokument zu generieren. Wenn Sie die einzelnen Dokumente generieren, können Sie sie jeweils in eine Datei exportieren. Diese Methode ist die einfachste Art und Weise, die Verwendung von Ergebnismengen zu veranschaulichen. Effektivere Möglichkeiten zum Abrufen von Daten finden Sie in den CLI-
n *im Verzeichnis dxx_install_verz*/samples/db2xml/cli.

XML-Dokument in eine HTML-Datei umsetzen:

Um die Daten aus einem XML-Dokument in einem Browser anzuzeigen, müssen Sie das XML-Dokument in eine HTML-Datei umsetzen, indem Sie eine Formatvorlage und die Funktion XSLTransformToFile verwenden.

Führen Sie zum Umsetzen des XML-Dokuments in eine HTML-Datei folgende Schritte aus:

1. Generieren Sie mit einem Texteditor eine Formatvorlage, und nennen Sie sie getstart.xsl:

   ```
   <?xml version="1.0" encoding="UTF-8" ?>
    <xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="/">
     <html>
     <head/>
     <body>

     ...

       </body>
      </html>
    </xsl:template>
    </xsl:stylesheet>
   ```

   Im folgenden Verzeichnis befindet sich ein vollständiges Beispiel dieser Datei:

   %dxx_install_verz%/samples/db2xml/xslt/getstart.xsl

2. Erstellen Sie für jedes Element einen Befehl unter Verwendung des folgenden Formats:

```
<xsl:for-each select="xxxxxx">
```

Dieser Befehl wird für Umsetzungsanweisungen verwendet. Erstellen Sie einen Befehl für jedes Element der Hierarchie des XML-Dokuments. Beispiel:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:template match="/">
  <html>
  <head/>
  <body>

     <xsl:for-each select="Order">

       <xsl:for-each select="Customer">
          <xsl:for-each select="Name | Email">
           </xsl:for-each>
       </xsl:for-each>
<xsl:for-each select="Part">
        <xsl:for-each select="key | Quantity | ExtendedPrice | Tax">
           </xsl:for-each>

           <xsl:for-each select="Shipment">
        <xsl:for-each select="ShipDate | ShipMode">
        </xsl:for-each>
           </xsl:for-each>
        </xsl:for-each>
        </xsl:for-each>
    </body>
   </html>
 </xsl:template>
 </xsl:stylesheet>
```

3. Verwenden Sie zum Formatieren der HTML-Datei eine Liste, in der die Hierarchie der XML-Elemente angezeigt wird, um die Daten lesbarer zu machen. Erstellen Sie zusätzliche Textelemente zum Beschreiben der Daten. Ihre Formatvorlagedatei könnte wie folgt aussehen:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:template match="/">
  <html>
  <head/>
  <body>

     <ol style="list-style:decimal outside">
      <xsl:for-each select="Order">
       <li> Orderkey : <xsl:value-of-select="@Key"/ <br/>

       <xsl:for-each select="Customer">
        <b>Customer</b><br/>
          <xsl:for-each select="Name | Email">
      <xsl:value-of select="name()"/>
      <xsl:text> : </xsl:text>
              <xsl:value-of select="."/>
           <xsl:text>, </xsl:text>
           </xsl:for-each>
           </xsl:for-each>

           <br/><br/>
    <ol type="A">
```

```
        <xsl:for-each select="Part">
         <li><b>Parts</b><br/>
      Color : <xsl:value-of select="@color"/>
            <xsl:text>, </xsl:text>

        <xsl:for-each select="key | Quantity | ExtendedPrice | Tax">
    <xsl:value-of select="name()"/>
              <xsl:text> : </xsl:text>
              <xsl:value-of select="."/>
    <xsl:text>, </xsl:text>
           </xsl:for-each>

           <br/><br/>
      <ol type="a">
           <xsl:for-each select="Shipment">
        <li><b>Shipment</b><br/>
        <xsl:for-each select="ShipDate | ShipMode">
    <xsl:value-of select="name()"/>
              <xsl:text> : </xsl:text>
              <xsl:value-of select="."/>
    <xsl:text>, </xsl:text>
        </xsl:for-each>
       </li>
           </xsl:for-each>
          </ol><br/>
       </li>
        </xsl:for-each>
       </ol>
       </li>
        </xsl:for-each>
       </ol>
    </body>
   </html>
 </xsl:template>
 </xsl:stylesheet>
```

4. Verwenden Sie Xpath zum Editieren der Befehle <xsl:value-of select=″xxx″> mit Daten aus dem XML-Dokument.

   Die Elementbefehle sind <xsl:value-of select″.″>. Dabei wird der Punkt (″.″) zum Abrufen von normalen Elementen verwendet.

   Die Befehle <xsl:value-of select=″@attributname″> sind Attributbefehle, wobei das vor dem Attributnamen hinzugefügte Et-Zeichen (@) den Wert des Attributs extrahiert. Sie können den Befehl <xsl:value-of select=″name()″> verwenden, um den Namen des XML-Befehls abzurufen.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:template match="/">
  <html>
  <head/>
  <body>

     <ol style="list-style:decimal outside">
      <xsl:for-each select="Order">
       <li> Orderkey : <xsl:value-of-select="@Key"/ <br/>

       <xsl:for-each select="Customer">
        <b>Customer</b><br/>
          <xsl:for-each select="Name | Email">
      <xsl:value-of select="name()"/>
      <xsl:text> : </xsl:text>
              <xsl:value-of select="."/>
          <xsl:text>, </xsl:text>
          </xsl:for-each>
```

```
        </xsl:for-each>

        <br/><br/>
    <ol type="A">
        <xsl:for-each select="Part">
         <li><b>Parts</b><br/>
      Color : <xsl:value-of select="@color"/>
        <xsl:text>, </xsl:text>

        <xsl:for-each select="key | Quantity | ExtendedPrice | Tax">
    <xsl:value-of select="name()"/>
            <xsl:text> : </xsl:text>
            <xsl:value-of select="."/>
    <xsl:text>, </xsl:text>
        </xsl:for-each>

        <br/><br/>
      <ol type="a">
        <xsl:for-each select="Shipment">
        <li><b>Shipment</b><br/>
        <xsl:for-each select="ShipDate | ShipMode">
    <xsl:value-of select="name()"/>
            <xsl:text> : </xsl:text>
            <xsl:value-of select="."/>
    <xsl:text>, </xsl:text>
        </xsl:for-each>
       </li>
        </xsl:for-each>
      </ol><br/>
       </li>
        </xsl:for-each>
       </ol>
       </li>
        </xsl:for-each>
       </ol>
     </body>
    </html>
  </xsl:template>
  </xsl:stylesheet>
```

5. Speichern Sie die Formatvorlage.
6. Sie haben die folgenden Möglichkeiten, um die HTML-Datei zu erstellen:
   - Verwenden Sie XSLTransformToFile:

     ```
     SELECT XSLTransformToFile( CAST(doc AS CLOB(4k)),
         'dxx_install_verz\samples\xslt\getstart.xsl',
       'dxx_install_verz\samples\html\getstart.html')
      FROM RESULT_TAB
     ```

   - Verwenden Sie folgenden Befehl:

     ```
     Getstart_xslt.cmd
     ```

Die Ausgabedatei kann nur auf ein Dateisystem geschrieben werden, auf das der DB2 UDB-Server zugreifen kann.

Lernprogrammumgebung bereinigen:

Zum Bereinigen der Lerneinheitumgebung können Sie eine der folgenden gelieferten Prozeduren ausführen oder die Befehle von der Befehlszeile aus eingeben. Dabei werden folgende Aktionen ausgeführt:

- Inaktivieren der XML-Spalte ORDER.
- Löschen der in den Lerneinheiten erstellten Tabellen.
- Löschen der DTD aus der DTD-Repositorytabelle.

Dabei wird die Datenbank SALES_DB nicht inaktiviert oder gelöscht. Sie ist weiterhin zur Verwendung mit XML Extender verfügbar. Möglicherweise werden Fehlernachrichten angezeigt, wenn Sie nicht beide Lerneinheiten dieses Abschnitts ausgeführt haben. Sie können diese Nachrichten ignorieren.

Gehen Sie wie folgt vor, um die Lernprogrammumgebung zu bereinigen:

Führen Sie die Befehlsdatei für die Bereinigung aus, indem Sie eine der folgenden Methoden verwenden:

- Geben Sie den folgenden Befehl ein:

  ```
  getstart_clean.cmd
  ```

- Wenn Sie die Datenbank inaktivieren möchten, können Sie den folgenden XML Extender-Befehl von der Befehlszeile aus 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 von XML Extender bereitgestellt wurden.

- Wenn Sie die Datenbank freigeben wollen, können Sie den folgenden Befehl von der Befehlszeile aus ausführen:

  ```
  db2 drop database SALES_DB
  ```

  Dieser Befehl gibt SALES_DB frei.

Zugehörige Konzepte:

- „Einführung in XML Extender" auf Seite 3
- „Lektion: XML-Dokument in einer XML-Spalte speichern" auf Seite 9
- „Lernprogrammübungen zu XML Extender" auf Seite 8

Teil 2. Verwaltung

Dieser Teil beschreibt die Ausführung von Verwaltungstasks für XML Extender.

Kapitel 2. Verwaltung

Verwaltungstools für XML Extender

Die Verwaltungstools von XML Extender erleichtern das Aktivieren der Datenbank und der Tabellenspalten für XML und das Zuordnen der XML-Daten zu den relationalen DB2®-Strukturen. XML Extender verfügt über das folgende Befehlszeilentool und über Programmierschnittstellen für Verwaltungstasks.

XML Extender stellt die folgenden Tools zur Ausführung von Verwaltungstasks zur Verfügung:

- Der Verwaltungsassistent von XML Extender bietet eine grafische Benutzerschnittstelle für Verwaltungstasks.
- Der Befehl **dxxadm** bietet eine Befehlszeilenoption für Verwaltungstasks.
- Mit den gespeicherten Prozeduren der XML Extender-Verwaltung können Sie Verwaltungsbefehle von einem Programm aus aufrufen.

Verwaltung von XML Extender vorbereiten

Um XML Extender auszuführen, ist die Installation der folgenden Software erforderlich.

Erforderliche Software: Für XML Extender ist DB2® Universal Database Version 8.2 erforderlich.

Zusatzsoftware:

- DB2 Universal Database Net Search Extender Version 8.2 für die strukturelle Textsuche. Diese Software wird mit DB2 Universal Database Version 8.2 zur Verfügung gestellt.
- Folgende Komponenten für den Verwaltungsassistenten von XML Extender:
  - DB2 Universal Database Java Database Connectivity (JDBC)
  - SDK ab Version 1.1.7 oder JRE Version 1.1.1, die mit der DB2 UDB-Steuerzentrale zur Verfügung gestellt werden
  - JFC 1.1 mit Swing 1.1, die mit der DB2 UDB-Steuerzentrale zur Verfügung gestellt wird

Bevor Sie XML Extender installieren, müssen Sie die folgenden Tasks abschließen:

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

  Sie müssen XML Extender an jede Datenbank binden. Ein Beispiel finden Sie im folgenden Verzeichnis:

  dxx_install_verz/samples/db2xml/cmd/getstart_prep.cmd
- Anzeigen der Einstellungsanweisungen.
- Erstellen einer Datenbank für den XML-Zugriff.

Um Verwaltungstasks unter Verwendung von XML Extender auszuführen, müssen Sie über die Berechtigung DB2ADM verfügen.

XML Extender von früheren Versionen migrieren

Wenn Sie mit einer früheren Version von DB2® XML Extender arbeiten, müssen Sie jede Datenbank migrieren, die für XML Extender aktiviert ist, bevor diese unter XML Extender Version 8.2 eingesetzt werden kann.

Das Migrationsprogramm führt in Abhängigkeit von der installierten Basisstufe von XML Extender verschiedene Schritte aus. Das Migrationsprogramm kann folgende Schritte ausführen:

- Erstellen von benutzerdefinierten Typen (UDTs) und benutzerdefinierten Funktionen (UDFs) vom Typ XMLCLOB für die Verwendung mit Unicode- und DBCS-Datenbanken.
- Erstellen von zusätzlichen gespeicherten Prozeduren für dxxGenXML und dxxRetrieveXML, um die Verwendung von temporären Tabellen zu unterstützen.
- Ändern der gespeicherten Prozeduren zur Verwendung des Parameterstils SQL als Verbindungskonvention zur Weitergabe von Parametern.
- Erstellen neuer benutzerdefinierter Funktionen für Schemata, für die DTD-Überprüfung und für die XSLT-Funktion.
- Erstellen neuer gespeicherter Prozeduren (dxxGenXMLCLOB und dxxRetrieveXMLCLOB), die CLOBs zurückgeben.
- Löschen und erneutes Erstellen von benutzerdefinierten Funktionen (UDFs), mit denen die Parallelfähigkeit für die skalaren UDFs verwendet werden kann.

Anmerkung: dxxEnableColl wird in dxxEnableCollection umbenannt, und dxxDisableColl wird in dxxDisableCollection umbenannt. Eine gespeicherte Prozedur, db2xml.dxxDisableDB, wurde ebenfalls hinzugefügt.

Verwenden Sie beim Aufrufen von gespeicherten Prozeduren einen Punkt (.) statt eines Ausrufezeichens (!) im Prozedurnamen. Verwenden Sie z. B. *db2xml.dxxEnableColumn* statt *db2xml!dxxEnableColumn*.

Vorgehensweise

Gehen Sie wie folgt vor, um eine XML-fähige Datenbank und eine XML-fähige Spalte zu migrieren:

1. Installieren Sie DB2 UDB XML Extender Version 8.1.
2. Geben Sie in der DB2 UDB-Befehlszeile Folgendes ein:

```
db2 connect to datenbankname
db2 bind @dxxMigv.lst
dxxMigv datenbankname
```

XML Extender von früheren Releases auf Fixpacks migrieren

Das Migrationsprogramm migriert XML Extender von früheren Releases auf aktualisierte Fixpacks. Sichern Sie Ihre Datenbank, bevor Sie das Migrationsprogramm ausführen.

Gehen Sie wie folgt vor, um die Datenbank zu migrieren:

Die Migrationsdateien befinden sich im CD-Installationsimage für das Fixpack. Sie müssen diese Schritte in dem Verzeichnis ausführen, in dem Sie die Fixpackdateien extrahiert haben.

Führen Sie die folgenden Schritte aus, wenn Sie UNIX oder Windows verwenden.

1. Geben Sie in der DB2 UDB-Befehlszeile Folgendes ein:

   ```
   db2 connect to datenbankname
   db2 bind @dxxMigv.lst
   ```

2. Geben Sie in der DB2 UDB-Befehlszeile Folgendes ein:

   ```
   dxxMigv
   datenbankname
   ```

Wenn Sie die Migration nicht vornehmen, kann dies zu Problemen und unerwarteten Ergebnissen führen. So kann es z. B. zu einem Fehlschlagen bei der Inaktivierung von Datenbanken kommen, oder der Zugriff auf neue UDFs wird verweigert.

Zugehörige Konzepte:

• „XML Extender von früheren Versionen migrieren" auf Seite 40

Übersicht: XML Extender-Verwaltung

XML Extender stellt drei Verwaltungsmethoden zur Verfügung: den XML Extender-Verwaltungsassistenten, den XML Extender-Verwaltungsbefehl und die gespeicherten Prozeduren von XML Extender.

- Der Verwaltungsbefehl **dxxadm** enthält Optionen für die verschiedenen Verwaltungstasks.
- Verwaltungsbefehle können ausgeführt werden, indem von einem Programm aus für die Verwaltung gespeicherte Prozeduren aufgerufen werden.
- Der Verwaltungsassistent von XML Extender führt Sie durch die Verwaltungstasks. Sie können ihn von einer Client-Workstation aus verwenden.

Beim Planen einer Anwendung, die XML-Dokumente verwendet, müssen Sie zunächst Folgendes festlegen:

- Ob XML-Dokumente aus Daten in der Datenbank zusammengesetzt werden.
- Ob bestehende XML-Dokumente gespeichert werden. Wenn Sie XML-Dokumente speichern, müssen Sie auch entscheiden, ob diese Dokumente als intakte XML-Dokumente in einer Spalte oder als relationale Daten zerlegt gespeichert werden sollen.

Wenn Sie sich festgelegt haben, können Sie Folgendes entscheiden:

- 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® UDB-Tabellen zugeordnet werden soll.

Verwaltungsassistent konfigurieren

Die Verwaltungstasks für XML Extender umfassen das Aktivieren der Datenbankspalten für XML und das Zuordnen der XML-Daten zu relationalen DB2 UDB-Strukturen. Sie können zur Ausführung dieser Verwaltungstasks den Assistenten von XML Extender verwenden. In diesem Abschnitt wird erläutert, wie Sie den Verwaltungsassistenten einrichten und aufrufen können. Sie können den Assistenten entweder über das Windows-Startmenü oder über eine Eingabeaufforderung aufrufen.

Voraussetzungen:

Bevor Sie den Assistenten konfigurieren, müssen Sie den Verwaltungsassistenten installieren und konfigurieren, wie es in der Readme-Datei für Ihr Betriebssystem beschrieben wird. Sie müssen die erforderlichen Klassendateien in die Umgebungsvariable CLASSPATH integrieren.

Überprüfen Sie, dass das Format der Umgebungsvariable CLASSPATH (mit Ausnahme der Zeilenumbrüche) dem folgenden Beispiel entspricht:

```
C:\db2_install_verz\java\db2java.zip;C:\db2_install_verz\xml-apis.jar;
C:\db2_install_verz\dxxadmin.jar;C:\db2_installationsverzeichnis\xerces.jar;
```

Dabei ist *db2_install_verz* das Installationsverzeichnis.

Vorgehensweise:

Gehen Sie wie folgt vor, um den Verwaltungsassistenten von XML Extender zu konfigurieren:

1. Rufen Sie den Assistenten unter Verwendung des SDK (Software Developer's Kit) auf. Sie können entweder den SDK oder die Java Runtime Environment (JRE) verwenden.
   - Wenn Sie die JRE verwenden, geben Sie Folgendes ein:

     ```
     jre -classpath klassenpfad com.ibm.dxx.admin.Admin
     ```
   - Wenn Sie den SDK verwenden, geben Sie Folgendes ein:

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

   Dabei gibt *klassenpfad* die Umgebungsvariable %CLASSPATH% an, die wiederum den Standort der Klassendateien des Verwaltungsassistenten angibt. Bei Verwendung dieser Option muss Ihre Systemvariable CLASSPATH auf die Verzeichnisse `dxx_install_verz/tools` und `dxx_install_verz/java` zeigen, die die folgenden Dateien enthalten: `dxxadmin.jar`, `xerces.jar`, `xml-apis.jar` und `db2java.zip`. Beispiel:

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

   Der *klassenpfad* kann auch ein Überschreiben der Umgebungsvariablen %CLASSPATH% mit Zeigern auf Dateien im Verzeichnis `dxx_install_verz/dxxadmin` angeben, von dem aus Sie den Verwaltungsassistenten von XML Extender ausführen. Beispiel:

   ```
   java -classpath dxxadmin.jar;xml4j.jar;db2java.zip com.ibm.dxx.admin.Admin
      url=jdbc:db2:mydb2 userid=db2xml password=db2xml
      driver=COM.ibm.db2.jdbc.app.DB2Driver
   ```
2. Melden Sie sich vom Anmeldefenster aus bei der Datenbank an, die Sie zum Arbeiten mit XML-Daten verwenden wollen.
3. Geben Sie im Feld **Adresse** die vollständig qualifizierte JDBC-URL für die Datenquelle ein, zu der eine Verbindung hergestellt werden soll. Die Adresse hat die folgende Syntax:

   Für eigenständige Konfigurationen: (Standardwert und empfohlen)

   ```
   jdbc:db2:datenbankname
   ```

   Dabei ist *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:portnummer/datenbankname
```

Hierbei gilt Folgendes:

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

portnummer
Die für die Verbindung mit dem Server verwendete Portnummer. Geben Sie zum Ermitteln der Portnummer den folgenden Befehl von einer Befehlszeile auf der Servermaschine ein:

```
db2jstrt portnummer
```

Benutzer von Windows NT können die Portnummer 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
```

4. Geben Sie die DB2 UDB-Benutzer-ID und das Kennwort für die gewünschte Datenbank in den Feldern **Benutzer-ID** und **Kennwort** ein bzw. überprüfen Sie diese Angaben.
5. 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.jcc.DB2DRIVER
   ```

6. Klicken Sie den Knopf **Fertig stellen** an. Rufen Sie den Assistenten auf, und gehen Sie zum Klickstartleistenfenster.

Nachdem Sie diese Prozedur abgeschlossen haben, können Sie den Assistenten im Klickstartleistenfenster aufrufen. Mit Hilfe des Assistenten können Sie über die folgenden Funktionen verfügen:

- Aktivieren oder Inaktivieren einer Datenbank.
- Hinzufügen einer DTD zum DTD-Repository.
- Arbeiten mit XML-Spalten.
- Arbeiten mit XML-Objektgruppen.

Zugriffs- und Speichermethoden

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

XML-Spalte
Ganze XML-Dokumente werden als DB2 UDB-Spaltendaten gespeichert und abgerufen. Die XML-Daten werden in einer XML-Spalte dargestellt.

XML-Objektgruppe
XML-Dokumente werden in eine Objektgruppe relationaler Tabellen zerlegt oder aus einer Objektgruppe relationaler Tabellen zusammengesetzt.

Von den Merkmalen Ihrer Anwendung hängt es ab, welche Zugriffs- und Speichermethode am besten geeignet ist und wie die XML-Daten strukturiert werden sollten.

Sie verwenden die DAD-Datei zum Zuordnen von XML-Daten zu DB2 UDB-Tabellen über diese beiden Zugriffs- und Speichermethoden. Abb. 4 zeigt, wie die DAD die Zugriffs- und Speichermethode angibt.

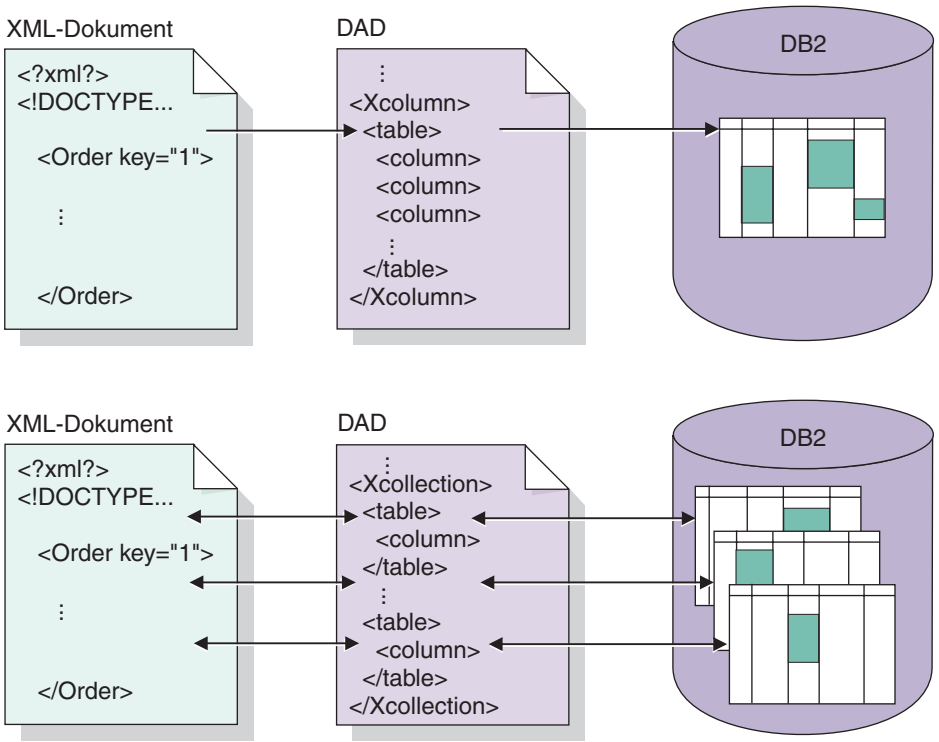


Abbildung 4. Die DAD-Datei ordnet die XML-Dokumentstruktur einer relationalen DB2 UDB-Datenstruktur zu und gibt die Zugriffs- und Speichermethode an.

Die DAD-Datei definiert den Standort der Schlüsseldateien wie der DTD und gibt die Beziehung zwischen der XML-Dokumentstruktur und den DB2 UDB-Daten an. Vor allem aber definiert sie die in der Anwendung verwendete Zugriffs- und Speichermethode.

Zugehörige Konzepte:

- „Einsatz der XML-Spaltenmethode" auf Seite 45
- „Einsatz der XML-Objektgruppenmethode" auf Seite 45

Zugehörige Referenzen:

- „Übersicht: Speicherfunktionen in XML Extender" auf Seite 146

Einsatz der XML-Spaltenmethode

Verwenden Sie in folgenden Situationen XML-Spalten:

- Die XML-Dokumente sind bereits vorhanden oder stammen aus einer 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 häufig gelesen, aber nicht aktualisiert.
- Sie wollen Dateinamendatentypen verwenden, um die XML-Dokumente außerhalb von DB2 UDB im lokalen oder fernen Dateisystem zu speichern und um DB2 UDB für Verwaltungs- und Suchoperationen verwenden zu können.
- Sie müssen Bereichssuchen auf der Basis der Werte von XML-Elementen oder -Attributen durchführen. Sie wissen, welche Elemente bzw. Attribute häufig als Suchargumente verwendet werden.
- Die Dokumente enthalten Elemente mit großen Textblöcken, und Sie wollen DB2 UDB Text Extender für eine strukturelle Textsuche verwenden und dabei die gesamten Dokumente intakt lassen.

Einsatz der XML-Objektgruppenmethode

Verwenden Sie in folgenden Situationen XML-Objektgruppen:

- 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. Und Sie wollen die Flexibilität haben, selbst zu entscheiden, ob die Daten in vorhandenen oder in neuen Tabellen gespeichert werden sollen.

XML-Spalten planen

Bevor Sie mit XML Extender Ihre Dokumente speichern, müssen Sie die Struktur des XML-Dokuments verstehen, um festlegen zu können, wie Elemente und Attribute im Dokument indexiert werden sollen. Bei der Planung, wie das Dokument indexiert werden soll, müssen Sie Folgendes festlegen:

- Den benutzerdefinierten XML-Typ, in dem das XML-Dokument gespeichert werden soll.
- Die XML-Elemente und -Attribute, die Ihre Anwendung häufig sucht, so dass deren Inhalt in Nebentabellen gespeichert und indexiert werden kann, um eine höhere Leistung zu erzielen.
- Ob XML-Dokumente in der Spalte mit einer DTD geprüft werden sollen.
- Die Struktur der Nebentabellen und wie sie indexiert werden.

XML-Datentypen für die XML-Spalten

XML Extender bietet benutzerdefinierte XML-Typen, die Sie zum Definieren einer Spalten, die XML-Dokumente halten soll, verwenden können. Diese Datentypen sind in Tabelle 4 beschrieben.

Tabelle 4. Die XML Extender-UDTs

Spalte benutzerdefinierter Typ	Quellendatentyp	Beschreibung der Verwendung
XMLVARCHAR	VARCHAR(*varchar_länge*)	Speichert ein vollständiges XML-Dokument als VARCHAR-Datentyp in DB2®. Wird für kleine Dokumente (kleiner als 3K) verwendet, die in DB2 gespeichert werden.
XMLCLOB	CLOB(*clob_länge*)	Speichert ein vollständiges XML-Dokument als CLOB-Datentyp in DB2. Wird für große Dokumente (größer als 3K) verwendet, die in DB2 gespeichert werden.
XMLFILE	VARCHAR(512)	Speichert den Dateinamen eines XML-Dokuments in DB2 und speichert das XML-Dokument in einer lokalen Datei auf dem DB2-Server. Wird für außerhalb von DB2 gespeicherte Dokumente verwendet.

Zu indexierende Elemente und Attribute für XML-Spalten

Wenn Sie die XML-Dokumentstruktur und die Anforderungen der Anwendung verstehen, können Sie festlegen, welche Elemente und Attribute am häufigsten durchsucht und extrahiert werden oder welche am aufwendigsten zu durchsuchen sind. Die DAD-Datei für eine XML-Spalte kann die Standortpfade jedes Elements und Attributs relationalen Tabellen (Nebentabellen) zuordnen, die diese Objekte enthalten. Die Nebentabellen werden anschließend indexiert.

Tabelle 5 zeigt ein Beispiel für die Typen von Daten und die Standortpfade der Elemente und Attribute des Szenarios 'Erste Schritte' für XML-Spalten. Die Daten wurden als Informationen angegeben, die häufig durchsucht werden, und die Standortpfade zeigen auf Elemente und Attribute, die die Daten enthalten. Die DAD-Datei kann Nebentabellen diese Standortpfade zuordnen.

Tabelle 5. Zu durchsuchende Elemente und Attribute

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

DAD-Datei für XML-Spalten

Für XML-Spalten gibt die DAD-Datei in erster Linie an, wie die in einer XML-Spalte gespeicherten Dokumente indexiert werden sollen. Die DAD-Datei gibt eine DTD an, die zur Überprüfung von Dokumenten, die in die XML-Spalte eingefügt werden, verwendet wird. Die Größe dieser Datei kann bis zu 2 GB betragen.

Die DAD-Datei für XML-Spalten bietet eine Übersicht über die XML-Daten, die für die Indexierung in den Nebentabellen gespeichert werden sollen.

Zum Angeben der Zugriffs- und Speichermethode über die XML-Spalte verwenden Sie in der DAD-Datei den Befehl <Xcolumn>. Der Befehl <Xcolumn> gibt an, dass die XML-Daten als vollständige XML-Dokumente in für XML-Daten aktivierten DB2 UDB-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 einschließen. Sie greifen auf die XML-Spaltendaten vor allem über SQL-Anweisungen und die UDFs von XML Extender zu.

Zugehörige Konzepte:

• „Nebentabellen planen" auf Seite 63

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 relationale 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 oder ein Schema. Die Verwendung einer DTD oder eines Schemas bei der Überprüfung stellt sicher, dass das XML-Dokument gültig ist.

Zur Prüfung mit Hilfe einer DTD muss sich im XML Extender-Repository eine DTD befinden.

Wichtig: Entscheiden Sie vor dem Einfügen von XML-Daten in DB2, ob die Daten geprüft werden sollen. XML Extender überprüft keine bereits in DB2 eingefügten Daten.

Überlegungen:

- Beim Zusammensetzen von Dokumenten können Sie nur eine DTD verwenden.
- Beim Zusammensetzen können Sie mehrere Schemata verwenden.
- Wenn Sie kein Dokument überprüfen möchten, wird die durch das XML-Dokument angegebene DTD nicht verarbeitet. Es ist wichtig, dass DTDs zur Auflösung von Einheiten- und Attributstandardwerten verarbeitet werden, auch wenn Dokumentfragmente verarbeitet werden, die nicht geprüft werden können.

DAD-Datei für XML-Objektgruppen

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

Sie geben den Namen der DAD-Datei an, wenn Sie eine Objektgruppe aktivieren oder die DAD-Datei in gespeicherten Prozeduren der XML-Objektgruppe verwenden. 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 von XML Extender verwendet wird, hat sie den Datentyp CLOB. Diese Datei kann bis zu 100 KB umfassen.

Zum Angeben der Zugriffs- und Speichermethode über die XML-Objektgruppe verwenden Sie den Befehl in der DAD-Datei. Der Befehl <Xcollection> gibt an, dass die 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 älteren unternehmensspezifischen Geschäftsdaten sein oder Tabellen, die XML Extender vor kurzem erstellt hat.

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. 5 auf Seite 49 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 "dxx_install_verz/samples/db2xml/dtd/dad.dtd">
<DAD>
  ...
  <Xcollection>
    <SQL_stmt>
       ...
    </SQL_stmt>
    <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "dxx_install_verz/samples/db2xml/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 zugordnet 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> an
          <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 5. Knotendefinitionen in einer DAD-Datei für eine XML-Objektgruppe

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

XML Extender unterstützt auch Verarbeitungsanweisungen für Formatvorlagen. Hierzu wird das Element <stylesheet> verwendet. Das Element <stylesheet> muss sich innerhalb des Stammknotens der DAD-Datei befinden, wobei der Dokumenttyp (doctype) und der Prolog (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 Websphere Studio Application Developer verwenden, um die DAD-Datei zu erstellen und zu aktualisieren. Das Element <stylesheet> wird derzeit nicht vom Verwaltungsassistenten von XML Extender 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 zugeordnet werden müssen, die in XML-Dokumenten mit einer relationalen Struktur verwendet wird, müssen Sie die Gemeinsamkeiten und Unterschiede der beiden Strukturen kennen. Abb. 6 zeigt, wie die hierarchische Struktur relationalen Tabellenspalten zugeordnet werden kann.

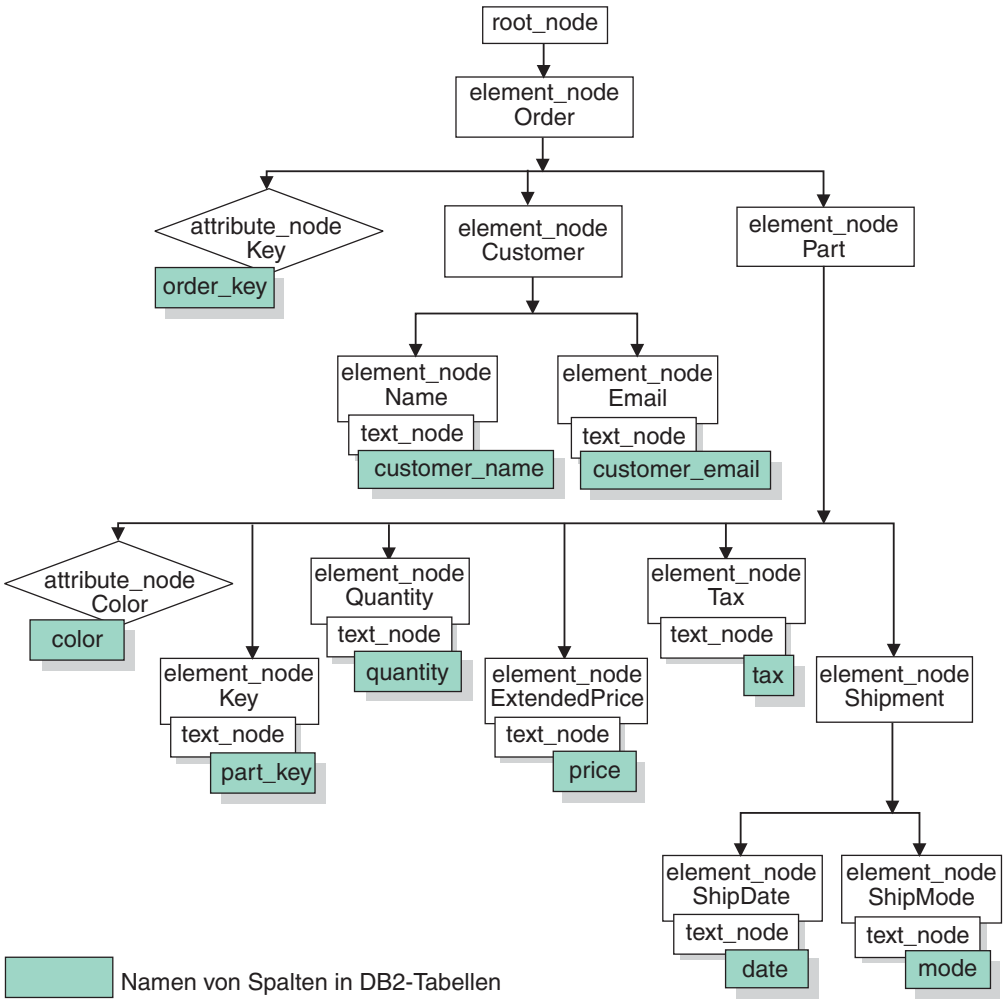


Abbildung 6. Relationalen Tabellenspalten zugeordnete XML-Dokumentstruktur

XML Extender verwendet das Zuordnungsschema beim Zusammensetzen oder Zerlegen von XML-Dokumenten mit relationalen Daten, die sich in verschiedenen relationalen Tabellen befinden. XML Extender enthält einen Assistenten, der Ihnen beim Erstellen der DAD-Datei behilflich ist. Bevor Sie jedoch die DAD-Datei erstellen, müssen Sie überlegen, wie die XML-Daten der XML-Objektgruppe zugeordnet werden sollen.

Arten von Zuordnungsschemata

Das Zuordnungsschema wird im Element <Xcollection> in der DAD-Datei angegeben. XML Extender bietet zwei Arten von Zuordnungsschemata: *SQL-Zuordnung* und *Zuordnung über die relationale Datenbank (RDB_node)*.

SQL-Zuordnung

Diese Methode ermöglicht ein direktes Zuordnen von relationalen Daten zu XML-Dokumenten über eine einzelne SQL-Anweisung. 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 des Elements SQL_stmt ist eine gültige SQL-Anweisung. Das Element SQL_stmt ordnet die Spalten in der SELECT-Klausel XML-Elementen oder -Attributen zu, die im XML-Dokument verwendet werden. Die Spaltennamen in der SELECT-Klausel der SQL-Anweisung werden zum Definieren des Werts eines attribute_node oder zum Definieren des Inhalts eines text_node verwendet. Die FROM-Klausel definiert die Tabellen mit den Daten, und die WHERE-Klausel gibt die Verknüpfungs- und Suchbedingung an.

Mit der SQL-Zuordnung können Benutzer von DB2 UDB unter Verwendung von SQL Daten zuordnen. Bei Verwendung der SQL-Zuordnung müssen Sie alle Tabellen in einer SELECT-Anweisung verbinden und damit eine Abfrage bilden können. 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 den Wert eines XML-Attributs, so dass XML Extender feststellen kann, wo die XML-Daten gespeichert bzw. von wo sie abgerufen werden sollen.

Diese Methode verwendet den von XML Extender gelieferten *RDB_node*, der eine oder mehrere Knotendefinitionen für Tabellen, wahlweise auszuführende Spalten und optionale Bedingungen enthält. 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. 7 auf Seite 52 zeigt ein Fragment einer Beispiel-DAD-Datei mit einer SQL-Zuordnung für XML-Objektgruppe, die eine Gruppe von XML-Dokumenten aus Daten in drei relationalen Tabellen zusammensetzt.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "dxx_install_verz/samples/db2xml/dtd/dad.dtd">
<DAD>
  <dtdid>dxx_install_verz/samples/db2xml/dtd/dad/
   getstart.dtd</dtdid>
  <validation>YES</validation>
  <Xcollection>
    <SQL_stmt>
       SELECT o.order_key, customer, p.part_key, quantity, price, tax, date,
              ship_id, mode, comment
         FROM order_tab o, part_tab p,
               table(select
               substr(char(timestamp(generate_unique())),16)

               as ship_id, date, node, from ship_tab)  shipid
        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 "dxx_install_verz
     /samples/db2xml/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><!- Ende Part->
    </element_node><!- Ende Order->
    </root_node>
  </Xcollection>
</DAD>
```

Abbildung 7. SQL-Zuordnungsschema

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 „Voraussetzungen für das Zuordnungsschema" definierten Regeln aufgebaut ist.

Voraussetzungen für das Zuordnungsschema

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

Voraussetzungen für das Zuordnungsschema für die 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 Abfrageprädikat verknüpfen kann. 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, dass Sie die Spalten entsprechend der Zuordnung der Tabellen zur hierarchischen Struktur des XML-Dokuments gruppieren. 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 Ebene 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 Ebene.
- Der Zusammensetzungsalgorithmus verwendet die Informationen des möglichen Schlüssels als Hinweis auf eine Hilfestellung bei der Verarbeitung. Verwenden Sie am Anfang einer Tabellenebene in der SELECT-Klausel einen einspaltigen möglichen Schlüssel. Wenn in einer Tabelle ein solcher Schlüssel nicht verfügbar ist, muss die Abfrage mit einem Tabellenausdruck und der integrierten benutzerdefinierten Funktion generate_unique() einen Schlüssel für diese Tabelle generieren. Generierte Schlüssel müssen im Ausgabedokument nicht angezeigt werden, allerdings müssen sie in der SELECT-Klausel enthalten sein. 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 über keinen Primärschlüssel verfügt, muss ein solcher generiert werden. In diesem Fall muss der Schlüssel `ship_id` generiert werden. Der Primärschlüssel 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 ein einzelner Kandidatenschlüssel für die Spalte mit der Funktion generate_unique() generiert und die Spalte 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 die Verknüpfungsbedingung an, die Tabellen in der Objektgruppe verbindet. Beispiel:

    ```
    WHERE p.price > 2500.00 AND s.date > '2003-06-01' 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 des Elements SQL_stmt.
  - Stellen Sie sicher, dass die Spaltennamen den Spaltennamen in der SELECT-Klausel entsprechen.
  - Listen Sie die möglichen einspaltigen Schlüssel in der Reihenfolge der hierarchischen Ebene der entsprechenden Tabellen auf.
  - 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 das Element SQL_stmt die oben angeführten Anforderungen gelten, ist es doch ein sehr leistungsstarkes Element, da Sie in Ihrer WHERE-Klausel ein beliebiges Prädikat angeben können, sofern der Ausdruck im Prädikat die Spalten in den Tabellen verwendet.

Voraussetzungen für das Zuordnungsschema für die RDB_node-Zuordnung

Verwenden Sie mit dieser Zuordnungsmethode nicht das Element SQL_stmt im Element <Xcollection> der DAD-Datei. Verwenden Sie stattdessen das Element RDB_node als untergeordnetes Element des Anfangselement_node und jedes attribute_node und text_node.

- **RDB_node für den Anfangs-element_node**

  Der Anfangselementknoten in der DAD-Datei steht für das Stammelement des XML-Dokuments. Geben Sie auf der Basis der folgenden Anforderungen für den Anfangs-element_node einen RDB_node an:
  - Zeilenendezeichen sind in Bedingungsanweisungen zulässig.
  - Bedingungselemente können unbegrenzt oft auf einen Spaltennamen verweisen.
  - 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>
```

  Es gibt keine Einschränkungen in Bezug auf die Reihenfolge bei Prädikaten der Stammknotenbedingung. 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, geben Sie für jede Tabelle einen

Primärschlüssel an. 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, wieder gemäß ihrer ursprünglichen Struktur 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.

• **RDB_node für jeden attribute_node und text_node**

Sie müssen für jeden attribute_node und text_node einen RDB_node angeben, wodurch der gespeicherten Prozedur mitgeteilt wird, 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 Anfangs-element_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. Mögliche Syntax für <condition>:

  - Spaltenname
  - Operator
  - Literal

  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, geben Sie den Spaltentyp für jeden attribute_node und text_node an. 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.

Anforderungen für die Tabellengröße beim Zerlegen für RDB_node-Zuordnung

Beim Zerlegen wird die RDB_node-Zuordnung verwendet, um anzugeben, wie ein XML-Dokument in DB2 UDB-Tabellen zerlegt wird, indem die Element- und Attributwerte in Tabellenzeilen extrahiert werden. Die Werte aus den einzelnen XML-Dokumenten werden in einer oder mehreren DB2-Tabellen gespeichert. Jede Tabelle kann maximal 10240 Zeilen aus jedem Dokument enthalten. Wenn ein XML-Dokument beispielsweise in fünf Tabellen zerlegt wird, kann jede dieser fünf Tabellen bis zu 10240 Zeilen für dieses bestimmte Dokument enthalten.

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, die für jedes Dokument eingefügt werden. Ein Dokument, das beispielsweise ein Element <Part> 20-mal enthält, kann als 20 Zeilen in eine Tabelle zerlegt werden. Beachten Sie bei der Verwendung von mehrfach vorkommenden Elementen, dass maximal 10240 Zeilen aus einem einzelnen Dokument in einer Tabelle zerlegt werden können.

Zugehörige Konzepte:

- „DAD-Dateien für XML-Objektgruppen" auf Seite 178

Zugehörige Tasks:

- „DTD in der Repository-Tabelle speichern" auf Seite 59

XML-Dokumente automatisch prüfen

Nachdem Sie eine Zugriffs- und Speichermethode, entweder XML-Spalte oder XML-Objektgruppe, ausgewählt haben, können Sie festlegen, ob die XML-Dokumente *geprüft* werden sollen. Sie können auch XML-Dokumente prüfen, die aus XML-Objektgruppen zusammengesetzt sind.

Sie können Ihre XML-Daten automatisch prüfen lassen, indem Sie in der DAD-Datei für den Befehl 'validation' den Wert YES angeben. Damit ein Dokument geprüft wird, wenn es in DB2® gespeichert wird, müssen Sie eine DTD innerhalb des Elements <dtdid> oder in der Spezifikation <!DOCTYPE> im Ursprungsdokument angeben. Damit ein Dokument geprüft wird, wenn es aus einer XML-Objektgruppe in DB2 zusammengesetzt wird, müssen Sie eine DTD innerhalb des Elements <dtdid> oder des Elements <doctype> in der DAD-Datei angeben.

Bei der Entscheidung, ob Ihre Dokumente geprüft werden sollen, beachten Sie folgende Aspekte:

- Die DTD-ID bzw. das Schema ist nur sinnvoll, wenn Sie das XML-Dokument überprüfen wollen.

  Fügen Sie die Schemabefehle ein, die die DAD-Datei der Schemadatei zuordnen, um DAD mit einem Schema zu prüfen. Beispiel:

  ```
  <schemabindings>
  <nonamespacelocation location="pfad/schema_name.xsd"/>
  </schemabindings>
  ```
- Sie brauchen keine DTD zum Speichern oder Archivieren von XML-Dokumenten.
- Möglicherweise ist es erforderlich, die DTD zu verarbeiten, um Entitätswerte und Standardwerte für Attribute zu setzen, unabhängig davon, ob Sie die Prüfung vornehmen möchten.
- Wenn Sie in der DAD für die Prüfung NO (keine Prüfung) angeben, wird die DTD, die durch das XML-Dokument angegeben wird, nicht verarbeitet.
- Die Überprüfung Ihrer XML-Daten hat Auswirkungen auf die Leistung.

Datenbanken für XML aktivieren

Bevor Sie XML-Dokumente speichern oder von DB2 UDB mit XML Extender abrufen können, müssen Sie die Datenbank für XML aktivieren. XML Extender aktiviert die verbundene Datenbank und verwendet das aktuelle Exemplar.

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

- Erstellen alle benutzerdefinierten Typen (UDTs), benutzerdefinierten Funktionen (UDFs) und gespeicherten Prozeduren für XML Extender.
- Erstellen der Steuertabellen und Füllen dieser Tabellen mit den erforderlichen Metadaten für XML Extender.
- Erstellen des DB2XML-Schemas in Bereichen benutzerdefinierter Tabellen und Zuordnen der erforderlichen Berechtigungen.

  Der vollständig qualifizierte Name einer XML-Funktion ist db2xml.*funktionsname*, wobei db2xml eine Kennung ist, die eine logische Gruppierung für SQL-Objekte bietet. Sie können den vollständig qualifizierten 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 UDB den Funktionspfad zum Ermitteln des Funktions- oder Datentyps.

Vorgehensweise:

Sie können mit dem Verwaltungsassistenten oder von einer Befehlszeile aus eine Datenbank aktivieren. Um diese Task von der Befehlszeile aus auszuführen, geben Sie in der Befehlszeile **dxxadm** ein. Geben Sie außerdem die zu aktivierende Datenbank an.

Mit dem folgenden Beispiel wird die bestehende Datenbank SALES_DB aktiviert.

```
dxxadm enable_db SALES_DB
```

Um mit dem Verwaltungsassistenten eine Datenbank zu aktivieren, müssen Sie die folgenden Tasks ausführen:

1. Starten Sie den Verwaltungsassistenten, und klicken Sie vom Klickstartleistenfenster aus **Datenbank aktivieren** an.

Wenn eine Datenbank bereits aktiviert wurde, wird der Knopf **Datenbank inaktivieren** angezeigt. Wenn die Datenbank inaktiviert wird, wird der Knopf **Datenbank aktivieren** angezeigt.

Wenn die Datenbank aktiviert ist, kehren Sie zum Klickstartleistenfenster zurück.

Nachdem Sie eine Datenbank aktiviert haben, können Sie die UDTs, UDFs und die gespeicherten Prozeduren von XML Extender verwenden.

Zugehörige Konzepte:

• „XML Extender von früheren Versionen migrieren" auf Seite 40

XML-Tabelle erstellen

Diese Task ist Teil der umfangreicheren Task zum Definieren und Aktivieren einer XML-Spalte.

Eine XML-Tabelle wird zum Speichern intakter XML-Dokumente verwendet. Zum Speichern ganzer Dokumente in Ihrer Datenbank mit Hilfe von DB2 UDB XML Extender müssen Sie eine Tabelle erstellen, so dass sie eine Spalte mit einem benutzerdefinierten XML-Typ (UDT) enthält. DB2 UDB XML Extender bietet drei benutzerdefinierte Typen zum Speichern Ihrer XML-Dokumente als Spaltendaten. Diese UDTs sind XMLVARCHAR, XMLCLOB und XMLFILE. Wenn eine Tabelle eine Spalte des Typs XML enthält, können Sie die Tabelle für XML aktivieren.

Sie können mit Hilfe des Verwaltungsassistenten oder über die Befehlszeile eine neue Tabelle erstellen, um eine Spalte des Typs XML hinzufügen.

Vorgehensweise:

So erstellen Sie über die Befehlszeile eine Tabelle mit einer Spalte des Typs XML:

Öffnen Sie die DB2 UDB-Befehlszeile, und geben Sie eine Anweisung zum Erstellen einer Tabelle ein (CREATE TABLE).

Sie wollen beispielsweise in einer Verkaufsanwendung eine XML-formatierte Bestellung in einer Spalte mit dem Namen ORDER einer Tabelle mit dem Namen 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 eine 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);
```

Nachdem Sie eine Tabelle erstellt haben, müssen Sie sie im nächsten Schritt für XML-Daten aktivieren.

Zugehörige Konzepte:

• „Nebentabellen planen" auf Seite 63
• Kapitel 13, „Tabellen zur Verwaltungsunterstützung von XML Extender", auf Seite 287

DTD in der Repository-Tabelle speichern

Sie können mit einer DTD die XML-Daten in einer XML-Spalte oder in einer XML-Objektgruppe überprüfen. DTDs können im DTD-Repository gespeichert werden, einer DB2 UDB-Tabelle mit dem Namen DTD_REF. Die Tabelle DTD_REF hat den Schemanamen DB2XML. Jede DTD in der Tabelle DTD_REF hat eine eindeutige ID. XML Extender erstellt die Tabelle DTD_REF, wenn Sie eine Datenbank für XML aktivieren. Sie können die DTD über die Befehlszeile einfügen oder mit Hilfe des Verwaltungsassistenten.

Vorgehensweise:

Gehen Sie wie folgt vor, um die DTD unter Verwendung des Verwaltungsassistenten einzufügen:

1. Starten Sie den Verwaltungsassistenten, und klicken Sie im Klickstartleistenfenster **DTD importieren** an, um für die aktuelle Datenbank eine vorhandene DTD-Datei in das DTD-Repository zu importieren. Das Fenster zum Importieren einer DTD wird geöffnet.
2. Geben Sie den Namen der DTD-Datei im Feld **DTD-Dateiname** an.
3. Geben Sie die DTD-ID im Feld **DTD-ID** ein.

   Die DTD-ID ist eine Kennung für die DTD. Dabei kann es sich auch um den Pfad handeln, 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.
4. Optional: Geben Sie den Namen des Autors der DTD im Feld **Autor** ein.
5. Klicken Sie **Fertig stellen** an, um die DTD in die DTD-Repository-Tabelle DB2XML.DTD_REF einzufügen, und kehren Sie zum Klickstartleistenfenster zurück.

Um eine DTD von der Befehlszeile aus einzufügen, geben Sie eine SQL-Anweisung INSERT aus Tabelle 6 ein. Beispiel:

```
DB2 INSERT into DB2XML.DTD_REF values('dxx_install_verz
   /samples/db2xml/dtd/getstart.dtd',
DB2XML.XMLClobFromFile('dxx_install_verz/dxxsamples/dtd/getstart.dtd',
0, 'user1', 'user1', 'user1');
```

Tabelle 6. Die Spaltendefinitionen für die DTD-Repositorytabelle

Spaltenname	Datentyp	Beschreibung
DTDID	VARCHAR(128)	Die ID der DTD.
CONTENT	XMLCLOB	Der Inhalt der DTD.
USAGE_COUNT	INTEGER	Die Anzahl an XML-Spalten und XML-Objektgruppen in der Datenbank, die diese DTD verwenden.
AUTHOR	VARCHAR(128)	Der Autor der DTD, optionale Information, 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 letzte Aktualisierung vorgenommen wird.

XML-Spalten aktivieren

Zum Speichern eines XML-Dokuments in einer DB2 UDB-Datenbank müssen Sie für XML die Spalte aktivieren, die das Dokument enthalten soll. Durch das Aktivieren wird die Spalte für die Indexierung vorbereitet, so dass sie schnell durchsucht werden kann. Sie können eine Spalte mit Hilfe des Verwaltungsassistenten von XML Extender oder über die Befehlszeile aktivieren. Die Spalte muss den Typ XML haben.

Wenn XML Extender eine XML-Spalte aktiviert, führt er folgende Operationen durch:

- Lesen der DAD-Datei und Ausführen der folgenden Aktionen:
  - Prüfen, ob die DTD in der Tabelle DTD_REF vorhanden ist, wenn die DTD-ID angegeben wurde.
  - Erstellen von Nebentabellen für die XML-Spalte zum Indexieren.
  - Vorbereiten der Spalte für XML-Daten.
- Optionales Erstellen einer Standardsicht der XML-Tabelle und der Nebentabellen. Die Standardsicht zeigt die Anwendungstabelle und die Nebentabellen.
- Angeben einer Spalte ROOT ID, falls noch keine angegeben ist.

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

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

Sie können XML-Spalten mit dem Verwaltungsassistenten oder von einer DB2-Befehlszeile aus aktivieren.

Vorgehensweise (mit dem Verwaltungsassistenten):

So aktivieren Sie XML-Spalten mit dem Verwaltungsassistenten:

1. Konfigurieren Sie den Verwaltungsassistenten, und starten Sie ihn.
2. Klicken Sie im Klickstartleistenfenster die Option **Mit XML-Spalten arbeiten** an, um die zugehörigen Tasks zu XML Extender-Spalten anzuzeigen. Das Fenster **Eine Task auswählen** wird geöffnet.
3. Klicken Sie **Eine Spalte aktivieren** und anschließend **Weiter** an.
4. Geben Sie die Tabelle und die Spalte an.
   - Wählen Sie im Feld **Tabellenname** die Tabelle aus, die die XML-Spalte enthält.
   - Wählen Sie die zu aktivierende Spalte im Feld **Spaltenname** aus.
5. Geben Sie den DAD-Pfad- und Dateinamen im Feld **DAD-Dateiname** ein. Beispiel:

   ```
   dxx_install_verz/samples/dad/getstart.dad
   ```
6. Optional: Geben Sie den Namen eines vorhandenen Tabellenbereichs im Feld **Tabellenbereich** ein.

   Der Standardtabellenbereich enthält Nebentabellen, die XML Extender erstellt hat. Wenn Sie einen Tabellenbereich eingeben, werden die Nebentabellen in dem angegebenen Tabellenbereich erstellt. Geben Sie keinen Tabellenbereich an, werden die Nebentabellen in dem Standardtabellenbereich erstellt.

7. Optional: Geben Sie den Namen der Standardsicht im Feld **Standardsicht** ein.

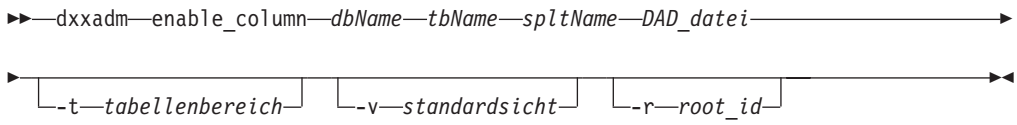
   Falls angegeben, wird die Standardsicht beim Aktivieren der Spalte automatisch erstellt. Die Standardsicht verknüpft die XML-Tabelle und alle zugehörigen Nebentabellen.
8. Empfohlen: Geben Sie den Spaltennamen des Primärschlüssels für die Tabelle im Feld **Root-ID** ein.

   XML Extender verwendet den Wert **Root-ID** als eindeutige Kennung für die Zuordnung aller Nebentabellen zur Anwendungstabelle. XML Extender fügt der Anwendungstabelle die Spalte DXXROOT_ID hinzu und generiert eine Kennung.
9. Klicken Sie **Fertig stellen** an, um die XML-Spalte zu aktivieren, die Nebentabellen zu erstellen und zum Klickstartleistenfenster zurückzukehren.
   - Wenn die Spalte erfolgreich aktiviert wurde, erhalten Sie die Nachricht `Die Spalte wurde aktiviert`.
   - Wurde die Spalte nicht erfolgreich aktiviert, wird eine Fehlernachricht angezeigt, und Sie werden aufgefordert, die Werte in den Eingabefeldern zu korrigieren, bis die Spalte erfolgreich aktiviert wird.

Vorgehensweise (über die Befehlszeile):

Um eine XML-Spalte über die Befehlszeile zu aktivieren, verwenden Sie den Befehl DXXADM enable_column.

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.

standardsicht
: Optional. Der Name der Standardsicht, die XML Extender zum Verknüpfen einer Anwendungstabelle und aller zugehörigen Nebentabellen erstellt hat.

root_id
: Optional, aber empfohlen. Der Spaltenname des Primärschlüssels in der Anwendungstabelle und eine eindeutige Kennung, die alle Nebentabellen der Anwendungstabelle zuordnet. DB2 XML Extender verwendet den Wert ROOT-ID als eindeutige Kennung für die Zuordnung aller Nebentabellen zur Anwendungstabelle. Wenn die ROOT-ID nicht angegeben ist, fügt XML Extender der Anwendungstabelle die Spalte DXXROOT_ID hinzu und generiert eine Kennung.

Einschränkung: Wenn die Anwendungstabelle einen Spaltennamen DXX-ROOT_ID enthält, müssen Sie den Parameter *root_id* angeben; andernfalls tritt ein Fehler auf.

Beispiel:

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

In diesem Beispiel wird die Spalte ORDER in der Tabelle SALES_TAB aktiviert. Die DAD-Datei ist `getstart.dad`, die Standardsicht ist sales_order_view und die ROOT-ID ist INVOICE_NUMBER.

In diesem Beispiel hat die Tabelle SALES_TAB die folgenden Spalten:

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

Die folgenden Nebentabellen werden entsprechend der DAD-Spezifikation erstellt:

ORDER_SIDE_TAB:

Spaltenname	Datentyp	Pfadausdruck
ORDER_KEY	INTEGER	`/Order/@Key`

CUSTOMER	VARCHAR(50)	`/Order`
		`/Customer`
		`/Name`

INVOICE_NUM	CHAR(6)	n/v

PART_SIDE_TAB:

Spaltenname	Datentyp	Pfadausdruck
PART_KEY	INTEGER	`/Order/Part/@Key`
PRICE	DOUBLE	`/Order/Part` `/ExtendedPrice`
INVOICE_NUM	CHAR(6)	n/v

SHIP_SIDE_TAB:

Spaltenname	Datentyp	Pfadausdruck
DATE	DATE	`/Order/Part/` `Shipment/ShipDate`
INVOICE_NUM	CHAR(6)	n/v

Alle Nebentabellen 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 der Spalte INVOICE_NUM in die Nebentabellen eingefügt, wenn eine Zeile in die Haupttabelle eingefügt wird. Wenn Sie bei der Aktivierung der XML-Spalte ORDER eine Standardsicht angeben, erstellt XML Extender die Standardsicht sales_order_view. 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_side_tab.order_key, order_side_tab.customer,
            part_side_tab.part_key, part_side_tab.price,
            ship_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
```

Wenn Sie bei der Aktivierung den Tabellenbereich angeben, werden die Nebentabellen in dem angegebenen Tabellenbereich erstellt. Ist der Tabellenbereich nicht angegeben, werden die Nebentabellen in dem Standardtabellenbereich erstellt.

Nebentabellen planen

Nebentabellen sind DB2-Tabellen, die zum Extrahieren des Inhalts eines XML-Dokuments verwendet werden, das häufig durchsucht wird. Die XML-Spalte ist den Nebentabellen zugeordnet, die den Inhalt des XML-Dokuments enthalten. Wenn das XML-Dokument in der Anwendungstabelle aktualisiert wird, werden die Werte in den Nebentabellen automatisch aktualisiert.

Abb. 8 zeigt eine XML-Spalte mit Nebentabellen.

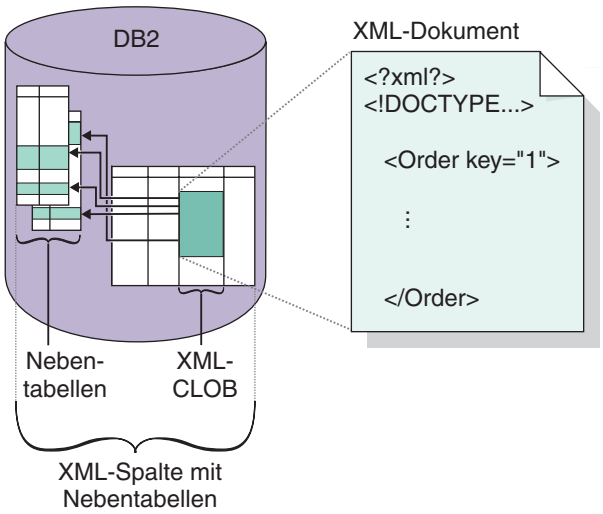


Abbildung 8. Eine XML-Spalte, deren Inhalt in Nebentabellen zugeordnet ist. In der Spalte befindet sich eine XML-Datei, die den Nebentabellen zugeordnet ist, die den Inhalt des XML-Dokuments enthalten.

Mehrfaches Vorkommen:

Wenn Elemente und Attribute mehrfach in Nebentabellen vorkommen, sollten Sie bei der Planung folgende Punkte berücksichtigen:

- 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 Nebentabelle erstellen. Das heißt, dass Elemente oder Attribute Standortpfade haben, die mehrfach auftreten, und einer Tabelle mit nur einer Spalte zugeordnet werden müssen. In der Tabelle können keine anderen Spalten enthalten sein.
- Wenn ein Dokument mehrfach auftretende Standortpfade hat, fügt XML Extender eine Spalte mit dem Namen DXX_SEQNO des Typs INTEGER zu jeder Nebentabelle hinzu, um die Reihenfolge der mehrfach auftretenden Elemente aufzuzeichnen. Mit DXX_SEQNO können Sie eine Liste der Elemente in der gleichen Reihenfolge wie im ursprünglichen XML-Dokument abrufen. Geben Sie hierzu in einer SQL-Abfrage ORDER BY DXX_SEQNO an.

Standardsichten und Abfrageleistung:

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

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

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

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

Nebentabellen indexieren

Diese Task ist Teil der umfangreicheren Task zum Definieren und Aktivieren einer XML-Spalte.

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

Vorgehensweise:

Um einen Index für Ihre Nebentabellen von einer DB2 UDB-Befehlszeile aus zu erstellen, verwenden Sie die SQL-Anweisung DB2 CREATE INDEX.

Das folgende Beispiel erstellt Indizes für vier Nebentabellen mit Hilfe der DB2-Eingabeaufforderung.

```
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-Dokumente mit Hilfe der SQL-Zuordnung zusammensetzen

Sie können XML-Dokumente mit Hilfe der SQL-Zuordnung von der Befehlszeile aus oder über den Verwaltungsassistenten zusammensetzen.

Sie sollten die SQL-Zuordnung verwenden, wenn Sie ein XML-Dokument zusammensetzen und eine SQL-Anweisung zum Definieren der Tabelle und der Spalten verwenden wollen, aus dem die Daten in dem XML-Dokument abgeleitet werden. Die SQL-Zuordnung kann ausschließlich zum Zusammensetzen von XML-Dokumenten verwendet werden. Sie erstellen eine DAD-Datei zum Zusammensetzen des XML-Dokuments mit der SQL-Zuordnung.

Voraussetzungen:

Bevor Sie Ihre Dokumente zusammensetzen, müssen Sie zunächst die Beziehung zwischen Ihren DB2 UDB-Tabellen und dem XML-Dokument zuordnen. Dieser Schritt umfasst das Zuordnen der Hierarchie des XML-Dokuments und die Angabe, wie die Daten im Dokument einer DB2 UDB-Tabelle zugeordnet sind.

Vorgehensweise:

Um XML-Dokumente von der Befehlszeile aus zusammenzusetzen, führen Sie die folgenden Schritte aus:

1. Erstellen Sie ein neues Dokument in einem Texteditor, und verwenden Sie folgende Syntax:

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

2. Fügen Sie die Befehle <DAD> </DAD> ein.

   Das DAD-Element enthält alle anderen Elemente.

3. Fügen Sie die Befehle ein, die zum Prüfen der DAD mit DTD oder einem Schema verwendet werden.

   - Fügen Sie die DTDID-Befehle ein, die die DAD-Datei dem XML-Dokument DTD zuordnen, um das zusammengesetzte XML-Dokument mit DTD zu prüfen. Beispiel:

     ```
     <dtdid>pfad/dtd_name.dtd>
     ```

   - Fügen Sie die Schemabefehle ein, die die DAD-Datei der Schemadatei zuordnen, um das zusammengesetzte XML-Dokument mit einem Schema zu prüfen. Beispiel:

     ```
     <schemabindings>
     <nonamespacelocation location="pfad/schema_name.xsd"/>
     </schemabindings>
     ```

Die DTD oder das Schema ist nur sinnvoll, wenn Sie das XML-Dokument überprüfen wollen. Verwenden Sie den Prüfbefehl um anzugeben, ob DB2 UDB XML Extender das XML-Dokument prüft.

- Wenn das XML-Dokument überprüft werden soll, geben Sie Folgendes ein:

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

- Wenn das XML-Dokument nicht überprüft werden soll, geben Sie Folgendes ein:

  ```
  <validation>NO</validation>
  ```

4. Geben Sie die Befehle `<Xcollection> </XCollection>` ein, um anzugeben, dass Sie XML-Objektgruppen als Zugriffs- und Speichermethode für Ihre XML-Daten verwenden.
5. Fügen Sie innerhalb der Befehle <Xcollection> </Xcollection> die Befehle <SQL_stmt> </SQL_stmt> ein, um die SQL anzugeben, die die Zuordnung von relationalen Daten zu XML-Dokumenten vornimmt. Diese Anweisung wird für die Abfrage von Daten aus DB2 UDB-Tabellen verwendet. Das folgende Beispiel zeigt eine Beispiel-SQL-Abfrage:

   ```
   <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>
   ```

   Die Beispiel-SQL-Anweisung für die Zuordnung der relationalen Daten zum XML-Dokument erfüllt die folgenden Anforderungen:

   - Spalten werden von oben nach unten angegeben entsprechend der Hierarchie der XML-Dokumentstruktur.
   - Die Spalten für eine Entität werden gruppiert.
   - Die Objekt-ID-Spalte ist in jeder Gruppe die erste Spalte.
   - Die Tabelle Order_tab hat keine einzelne Schlüsselspalte; daher wird die integrierte DB2 UDB-Funktion generate_unique zum Generieren der Spalte ship_id verwendet.
   - In der Klausel ORDER BY werden die Spalten mit den Objekt-IDs in der Reihenfolge aufgelistet, die der Top-down-Hierarchie der Tabelle entspricht, der sie angehören. Die Spalte in ORDER BY darf nicht über ein Schema qualifiziert werden, und die Spaltennamen müssen den Spaltennamen in der Klausel SELECT entsprechen.

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

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

7. Geben Sie den Befehl <doctype> </doctype> ein. Dieser Befehl enthält die Deklaration DOCTYPE, die in jedes erstellte Dokument eingefügt werden soll. Beispiel:

   ```
   <doctype>! DOCTYPE Order SYSTEM "dxx_install_verz
      /samples/db2xml/dtd/getstart.dtd"</doctype>
   ```

8. Geben Sie das Stammelement und die Elemente und Attribute an, aus denen sich das XML-Dokument zusammensetzt.
   a. Fügen Sie die Befehle <root_node></root_node> hinzu, um das Stammelement zu definieren. Alle Elemente und Attribute, aus denen das XML-Dokument besteht, werden innerhalb des root_node angegeben.
   b. Verwenden Sie die Befehle `<element_node>`, `<attribute_node>`, `<text_node>` und `<column>`, um die Namen von Elementen und Attributen im zusammengesetzten Dokument anzugeben und sie den in der SQL-Anweisung angegebenen Spalten zuzuordnen.

      Befehl <column>
      (Spalte) Gibt die Spalte an, aus der die Daten für den Element- oder Attributwert abgerufen werden.

      Befehl <element_node>
      (Elementknoten) Gibt die Elemente im XML-Dokument an. Setzen Sie das Namensattribut des Befehls 'element_node' auf den Namen des Elements fest. Jeder element_node kann untergeordnete Elementknoten enthalten.

      Befehl <attribute_node>
      (Attributknoten) Gibt die Attribute eines Elements im XML-Dokument an. Die Attribute werden in ihren Elementknoten verschachtelt. Setzen Sie das Namensattribut des Befehls 'attribute_node' auf den Namen des Attributs fest.

      Befehl <text_node>
      (Textknoten) Gibt den Textinhalt des Elements und die Spaltendaten in einer relationalen Tabelle für die Elementknoten der untersten Ebene an. Geben Sie für jedes Element auf der untersten Ebene die Befehle <text_node> an, um zu kennzeichnen, dass das Element Zeichendaten enthält, die beim Zusammensetzen des Dokuments aus DB2 extrahiert werden. Verwenden Sie für jeden 'element_node' der untersten Ebene einen Befehl <column>, um anzugeben, aus welcher Spalte Daten extrahiert werden sollen, wenn das XML-Dokument erstellt wird. Spaltenbefehle befinden sich üblicherweise in den Befehlen <attribute_node> oder <text_node>. Alle definierten Spaltennamen müssen in der SELECT-Klausel von <SQL_stmt> am Anfang der DAD-Datei stehen.
9. Stellen Sie sicher, dass die DTL-Endbefehle sich in den korrekten Positionen befinden:
   a. Stellen Sie sicher, dass der Endbefehl </root_node> sich hinter dem letzten Befehl </element_node> befindet.
   b. Stellen Sie sicher, dass der Endbefehl </Xcollection> sich hinter dem Befehl </root_node> befindet.
   c. Stellen Sie sicher, dass der Endbefehl </DAD> sich hinter dem Befehl </Xcollection> befindet.
10. Speichern Sie die Datei unter dem Namen *datei.dad*. Dabei ist *datei* der Name Ihrer Datei.

Das folgende Beispiel für Windows zeigt eine vollständige DAD:

```
<?xml version'"1.0"?>
<!DOCTYPE DAD SYSTEM "C:\dxx_xml\test\dtd\dad.dtd">
<DAD>
<validation>NO</validation>   <Xcollection>
<SQL_stmt> select o.order_key, customer_name, customer_email,
 p.part_key, color, qty, price, tax, ship_id, date, mode from order_tab o,
```

```
 part_tab p, (select db2xml.generate_unique() 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_install_verz\samples\db2xml\dtd/Order.dtd"
 </doctype>
 <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>
    <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="qty"/></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>
      </element_node>
      </element_node>
     </root_node>
     </Xcollection>
     </DAD>
```

XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen

Die RDB_node-Zuordnung verwendet die Befehle <RDB_node> zur Angabe von DB2 UDB-Tabellen, Spalten und Bedingungen für einen Element- oder Attributknoten. Verwenden Sie diese Methode, wenn Sie XML-Dokumente durch Verwendung einer XML-ähnlichen Struktur zusammensetzen wollen. 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 Gibt optional eine Bedingung für die Spalte an.

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

Für die Knotenart:	**Folgende untergeordnete RDB-Elemente werden verwendet:**		
	table (Tabelle)	**column (Spalte)**	**condition (Bedingung)**
Stammelement	Ja	Nein	Ja[1]
Attribut	Ja	Ja	Optional
Text	Ja	Ja	Optional

[1] Erforderlich bei mehreren Tabellen

Sie können den Verwaltungsassistenten oder eine Befehlszeile verwenden, um mit Hilfe der RDB_node-Zuordnung XML-Dokumente zusammenzusetzen.

Einschränkungen:

Wenn Sie Ihre XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen, müssen alle Anweisungen zu einem bestimmten Element mit den Spalten in der gleichen Tabelle übereinstimmen.

Vorgehensweise:

Führen Sie die folgenden Schritte aus, um ein XML-Dokument mit Hilfe der RDB_node-Zuordnung von der Befehlszeile aus zusammenzusetzen:

1. Öffnen Sie einen Texteditor, und erstellen Sie DAD-Kopfdaten, indem Sie folgende Syntax verwenden:

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "pfad/dad.dtd">
   ```

   Dabei ist *pfad*/dad.dtd der Pfad und Dateiname der DTD für die DAD.
2. Fügen Sie die Befehle <DAD></DAD> ein. Dieses Element enthält alle anderen Elemente.
3. Fügen Sie die Befehle ein, die zum Prüfen der DAD mit DTD oder einem Schema verwendet werden.
   - Fügen Sie die DTDID-Befehle ein, die die DAD-Datei dem XML-Dokument DTD zuordnen, um DAD mit DTD zu prüfen. Beispiel:

     ```
     <dtdid>pfad/dtd_name.dtd</dtdid>
     ```
   - Fügen Sie die Schemabefehle ein, die die DAD-Datei der Schemadatei zuordnen, um DAD mit einem Schema zu prüfen. Beispiel:

     ```
     <schemabindings>
     <nonamespacelocation location="pfad/schema_name.xsd"/>
     </schemabindings>
     ```

   Die DTD-ID oder das Schema ist nur sinnvoll, wenn Sie das XML-Dokument überprüfen wollen. Verwenden Sie den Prüfbefehl um anzugeben, ob DB2 UDB XML Extender das XML-Dokument prüft:
   - Wenn das XML-Dokument überprüft werden soll, geben Sie Folgendes ein:

     ```
     <validation>YES</validation>
     ```
   - Wenn das XML-Dokument nicht überprüft werden soll, geben Sie Folgendes ein:

     ```
     <validation>NO</validation>
     ```

4. Fügen Sie die Befehle <Xcollection></XCollection> ein, um anzugeben, dass Sie XML-Objektgruppen als Zugriffs- und Speichermethode für Ihre XML-Daten verwenden.
5. Fügen Sie die folgenden Prologinformationen hinzu:

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

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

   ```
   <doctype>! DOCTYPE Order SYSTEM "dxx_install_verz
   /samples/db2xml/dtd/getstart.dtd"</doctype>
   ```

7. Fügen Sie die Befehle <root_node></root_node> ein. Innerhalb der root_node-Befehle geben Sie die Elemente und Attribute an, aus denen das XML-Dokument besteht.
8. Ordnen Sie im Befehl <root_node> die Elemente und Attribute im XML-Dokument den Element- und Attributknoten zu, die den DB2 UDB-Daten entsprechen. Verwenden Sie das RDB_node-Element für den element_node, text_node und attribute_node. Diese Knoten stellen einen Pfad von den XML-Daten zu den DB2 UDB-Daten zur Verfügung. Gehen Sie wie folgt vor, um die Elemente und Attribute in Ihrem XML-Dokument zuzuordnen:
   a. Geben Sie einen RDB_node für den Anfangs-element_node an. Dieses Element gibt alle Tabellen an, die dem XML-Dokument zugeordnet sind. Um einen RDB_node für den Anfangs-element_node anzugeben, fügen Sie die Befehle <RDB_node> nach dem Befehl root_node ein.
      - Geben Sie einen RDB_node für den attribute_node an.
      - Geben Sie einen RDB_node für den text_node an.

   b. Definieren Sie einen Tabellenknoten 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>
      ```

      Wenn Sie ein XML-Dokument mit Hilfe einer DAD-Datei zerlegen, 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. Sie müssen außerdem einen Primärschlüssel für jede Tabelle angeben, wenn Sie eine Objektgruppe aktivieren. Das nachfolgende Beispiel zeigt, wie für jede Tabelle, die im element_node angegeben ist, eine Schlüsselspalte angegeben wird.

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

Zugehörige Konzepte:

- „Schemata für die Zuordnung von XML-Objektgruppen" auf Seite 110
- „Standortpfade" auf Seite 118
- „DAD-Dateien für XML-Objektgruppen" auf Seite 178
- „Voraussetzungen für die Verwendung der RDB_Node-Zuordnung" auf Seite 114
- „Übersicht: Gespeicherte Zusammensetzungsprozeduren von XML Extender" auf Seite 214

Zugehörige Tasks:

- „XML-Objektgruppe mit Hilfe der RDB_node-Zuordnung zerlegen" auf Seite 71
- „Daten in XML-Objektgruppen verwalten" auf Seite 98
- „Daten in XML-Objektgruppen aktualisieren und löschen" auf Seite 106

XML-Objektgruppe mit Hilfe der 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 UDB-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 Gibt optional eine Bedingung für die Spalte an.

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

Für die Knotenart:	**Wird folgendes untergeordnete RDB-Element verwendet:**		
	table (Tabelle)	**column (Spalte)**	**condition (Bedingung)**
Stammelement	Ja	Nein	Ja[1]
Attribut	Ja	Ja	wahlfrei
Text	Ja	Ja	wahlfrei

(1) Erforderlich bei mehreren Tabellen

Prozedur unter Verwendung einer Befehlszeile::

Gehen Sie wie folgt vor, um XML-Dokumente unter Verwendung einer Befehlszeile zu zerlegen:

1. Erstellen Sie in einem beliebigen Texteditor eine Datei. Erstellen Sie einen DAD-Header, indem Sie die folgende Syntax eingeben:

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "pfad/dad.dtd">
   ```

   Dabei ist *pfad*/dad.dtd der Pfad und Dateiname der DTD für die DAD.
2. Fügen Sie die Befehle <DAD></DAD> ein.

3. Fügen Sie die Befehle ein, die zum Prüfen der DAD mit DTD oder einem Schema verwendet werden.
   - Fügen Sie die DTDID-Befehle ein, die die DAD-Datei dem XML-Dokument DTD zuordnen, um DAD mit DTD zu prüfen. Beispiel:

     ```
     <dtdid>pfad/dtd_name.dtd</dtdid>
     ```

   - Fügen Sie die Schemabefehle ein, die die DAD-Datei der Schemadatei zuordnen, um DAD mit einem Schema zu prüfen. Beispiel:

     ```
     <schemabindings>
     <nonamespacelocation location="pfad/schema_name.xsd"/>
     </schemabindings>
     ```

   Die DTD-ID oder das Schema ist nur sinnvoll, wenn Sie das XML-Dokument überprüfen wollen. Verwenden Sie den Prüfbefehl um anzugeben, ob DB2 UDB XML Extender das XML-Dokument prüft:
   - Wenn das XML-Dokument überprüft werden soll, geben Sie Folgendes ein:

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

   - Wenn das XML-Dokument nicht überprüft werden soll, geben Sie Folgendes ein:

     ```
     <validation>NO</validation>
     ```

4. Fügen Sie die Befehle <Xcollection></XCollection> ein, um anzugeben, dass Sie XML-Objektgruppen als Zugriffs- und Speichermethode für Ihre XML-Daten verwenden.
5. Fügen Sie die folgenden Prologinformationen hinzu:

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

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

   ```
   <doctype>! DOCTYPE Order SYSTEM "dxx_install_verz
      /samples/db2xml/dtd/getstart.dtd"</doctype>
   ```

7. Definieren Sie den Root-Knoten mit den Befehlen <root_node></root_node>.
8. Ordnen Sie im root_node die Elemente und Attribute im XML-Dokument den Element- und Attributknoten zu, die den DB2 UDB-Daten entsprechen. Diese Knoten bieten einen Pfad von den XML-Daten zu den DB2 UDB-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 innerhalb eines element_node-Elements ein RDB_node-Element. Beispiel:

         ```
         <element_node name="name">
           <RDB_node>
             </RDB_node>
         </element_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:

```
tabellenname.tabellenspalte = tabellenname.tabellenspalte AND
tabellenname.tabellenspalte = tabellenname.tabellenspalte ...
```

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-element_node '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>
```

Das Schlüsselattribut ist zum Zerlegen und Aktivieren einer Objektgruppe erforderlich.

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

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

Ein Elementknoten kann einen der folgenden Elementtypen enthalten:

text_node Zur Angabe, dass das Element über Inhalt für eine DB2 UDB-Tabelle verfügt. In diesem Fall enthält das Element keine untergeordneten Elemente.

attribute_node
Zur Angabe eines Attributs.

child elements
Untergeordnete Elemente des element_node.

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

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

table Gibt die Tabelle an, der das Element oder Attribut zugeordnet wird.

column Gibt die Spalte an, der das Element oder Attribut zugeordnet wird.

condition Gibt optional eine Bedingung an, bei der eine Einfügeoperation für die Spalte vorgenommen wird.

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 den Befehl <RDB_node> ein, um das XML-Element der DB2 UDB-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 Befehl <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 für jedes Attribut in Ihrem XML-Dokument, das einer Spalte in einer DB2 UDB-Tabelle zugeordnet ist, einen Befehl <attribute_node>. Beispiel:

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

Der attribute_node enthält einen RDB_node für die Zuordnung des Attributwerts zu einer DB2 UDB-Tabelle und -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>
```

9. 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>
```

10. Stellen Sie sicher, dass die DTL-Endbefehle sich in den korrekten Positionen befinden:
    a. Stellen Sie sicher, dass der Endbefehl </root_node> sich hinter dem letzten Befehl </element_node> befindet.
    b. Stellen Sie sicher, dass der Endbefehl </Xcollection> sich hinter dem Befehl </root_node> befindet.
    c. Stellen Sie sicher, dass der Endbefehl </DAD> sich hinter dem Befehl </Xcollection> befindet.

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zerlegungsprozeduren von XML Extender" auf Seite 226

Zugehörige Tasks:

- „XML-Dokumente in DB2 UDB-Daten zerlegen" auf Seite 102
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Teil 3. Programmierung

Dieser Teil des Handbuchs beschreibt Programmiertechniken zur Verwaltung Ihrer XML-Daten.

Kapitel 3. XML-Spalten

In diesem Abschnitt wird beschrieben, wie Sie Daten in XML-Spalten mit DB2 verwalten.

Daten in XML-Spalten verwalten

Wenn Sie XML-Spalten zum Speichern von Daten verwenden, speichern Sie ein vollständiges XML-Dokument in seinem nativen Format als Spaltendaten in DB2. 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.

Nach dem Aktivieren einer Datenbank für XML stehen die folgenden benutzerdefinierten Typen (UDTs) von XML Extender zur Verfügung:

XMLCLOB
Verwenden Sie diesen UDT für XML-Dokumentinhalte, die als großes Zeichenobjekt (Character Large Object, CLOB) in DB2 gespeichert sind.

XMLVARCHAR
Verwenden Sie diesen UDT für XML-Dokumentinhalte, die als VARCHAR in DB2 gespeichert sind.

XMLFILE
Verwenden Sie diesen UDT für XML-Dokumente, die in einer Datei auf einem lokalen Dateisystem gespeichert sind.

Sie können Anwendungstabellen erstellen oder ändern, um Spalten vom XML-UDT-Datentyp zu erhalten. Diese Tabellen werden als XML-Tabellen bezeichnet.

Nachdem Sie eine Spalte in einer Tabelle für XML aktiviert haben, können Sie die XML-Spalte erstellen und die folgenden Verwaltungstasks 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

Um alle diese Tasks auszuführen, verwenden Sie die benutzerdefinierten Funktionen (UDFs), die von XML Extender geliefert werden. Verwenden Sie die Standardumsetzungsfunktionen zum Speichern von XML-Dokumenten in DB2. Standardumsetzungsfunktionen setzen den SQL-Basistyp in die benutzerdefinierten Typen von XML Extender um und wandeln Exemplare eines Datentyps (Ursprung) in Exemplare eines anderen Datentyps (Ziel) um.

Zugehörige Konzepte:

- „XML-Spalten als Speicher- und Zugriffsmethode" auf Seite 80
- „Indizes für XML-Spaltendaten verwenden" auf Seite 81

XML-Spalten als Speicher- und Zugriffsmethode

Sie können die Dokumentstruktur in der jeweiligen Form speichern oder verwalten, in der sie momentan vorhanden ist. XML enthält alle Informationen, die für die Erstellung einer Gruppe von Dokumenten notwendig sind.

Wenn Sie beispielsweise in einer Publishing-Firma arbeiten, die Artikel im Web bereitstellt, wollen Sie vielleicht ein Archiv der veröffentlichten Artikel verwalten. In einem solchen Szenario gibt Ihnen XML Extender die Möglichkeit, XML-Artikel vollständig oder teilweise in einer Spalte einer DB2®-Tabelle zu speichern, der *XML-Spalte*, wie in Abb. 9 dargestellt.

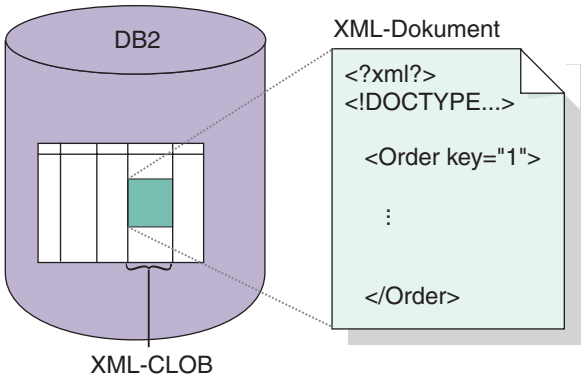


Abbildung 9. Speichern strukturierter XML-Dokumente in einer DB2 UDB-Tabellenspalte

Mit der Speicher- und Zugriffsmethode über XML-Spalten können die XML-Dokumente unter Verwendung von DB2 verwaltet werden. Sie können XML-Dokumente in einer Spalte vom Typ XML speichern, und Sie können den Inhalt des Dokuments abfragen, um ein bestimmtes Element oder Attribut zu finden. Sie können eine DTD in DB2 UDB für ein oder mehrere Dokumente zuordnen und speichern. Darüber hinaus können Sie den Inhalt von Elementen und Attributen zu DB2 UDB-Tabellen, den so genannten *Nebentabellen*, zuordnen. Diese Nebentabellen können zur Erzielung einer besseren Abfrageleistung indexiert werden; sie werden jedoch nicht automatisch indexiert. Die Spalte, die zum Speichern des Dokuments verwendet wird, wird XML-Spalte genannt. Dieser Begriff gibt an, dass die Spalte für die XML-Spaltenzugriffs- und -speichermethode verwendet wird.

In der DAD-Datei können Sie die Befehle <Xcolumn> und </Xcolumn> eingeben, um anzugeben, dass die Speicher- und Zugriffsmethode, die Sie verwenden, die XML-Spalte ist. Die DAD ordnet dann den XML-Element- und -Attributinhalt zu, der in den Nebentabellen gespeichert werden soll.

Bevor Sie mit XML Extender Ihre Dokumente speichern, müssen Sie die Struktur des XML-Dokuments verstehen, um festlegen zu können, wie Elemente und Attribute im Dokument indexiert werden sollen. Bei der Planung, wie das Dokument indexiert werden soll, müssen Sie Folgendes festlegen:

- Den benutzerdefinierten XML-Typ, in dem das XML-Dokument gespeichert werden soll.
- Die XML-Elemente und -Attribute, die Ihre Anwendung häufig sucht, so dass deren Inhalt in Nebentabellen gespeichert und indexiert werden kann, um eine höhere Leistung zu erzielen.
- Ob XML-Dokumente in der Spalte mit einer DTD geprüft werden sollen.

XML-Spalte definieren und aktivieren

Sie verwenden XML-Spalten zum Speichern von und Zugreifen auf vollständige XML-Dokumente in der Datenbank. Mit dieser Speichermethode können Sie Dokumente mit den XML-Dateitypen speichern, die Spalten in Nebentabellen indexieren und XML-Dokumente abfragen oder durchsuchen.

Verwenden Sie XML-Spalten, wenn Sie vollständige XML-Dokumente in einer DB2-Tabellenspalte speichern wollen, wenn das Dokument nicht häufig aktualisiert wird oder wenn Sie intakte XML-Dokumente speichern wollen.

Wenn Sie XML-Dokumentstrukturen zu DB2 UDB-Tabellen zuordnen wollen, so dass Sie XML-Dokumente aus vorhandenen DB2 UDB-Daten zusammensetzen oder XML-Dokumente in DB2-Daten zerlegen können, verwenden Sie XML-Objektgruppen an Stelle von XML-Spalten.

Vorgehensweise:

So definieren und aktivieren Sie eine XML-Spalte über die Befehlszeile:

1. Erstellen Sie eine DAD-Datei (Dokumentzugriffsdefinitionsdatei).
2. Erstellen Sie eine Tabelle, in der die XML-Dokumente gespeichert werden.
3. Aktivieren Sie die Spalte für XML-Daten.
4. Indexieren Sie Nebentabellen.

Die XML-Spalte wird als XML-Benutzerdatentyp erstellt. Wenn diese Tasks abgeschlossen sind, können Sie XML-Dokumente in der Spalte speichern. Diese Dokumente können anschließend aktualisiert, durchsucht und extrahiert werden.

Zugehörige Konzepte:

- „XML-Spalten als Speicher- und Zugriffsmethode" auf Seite 80
- „Indizes für XML-Spaltendaten verwenden" auf Seite 81
- „XML-Dokumente automatisch prüfen" auf Seite 56
- „Lektion: XML-Dokument in einer XML-Spalte speichern" auf Seite 9

Zugehörige Tasks:

- „DAD-Datei für XML-Spalten erstellen" auf Seite 175
- „XML-Tabelle erstellen" auf Seite 58
- „XML-Spalten aktivieren" auf Seite 60
- „Nebentabellen indexieren" auf Seite 64
- „Daten in XML-Spalten verwalten" auf Seite 79

Indizes für XML-Spaltendaten verwenden

Eine wichtige Planungsentscheidung bei der Verwendung von XML-Spalten ist, ob Nebentabellen für XML-Spaltendokumente indexiert werden sollen. 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 Nebentabellen 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. XML Extender extrahiert die Werte der XML-Elemente oder Attribute aus XML-Dokumenten und speichert sie in den Nebentabellen, so dass Sie Indizes zu diesen Nebentabellen erstellen können. Sie können jede Spalte einer Nebentabelle mit einem Standortpfad, der ein XML-Element oder -Attribut und einen SQL-Datentyp kennzeichnet, angeben.

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

Erstellen Sie für eine Schnellsuche mit Hilfe der DB2 UDB-Technologie *B-Baumstruktur-Indexierung* Indizes dieser Spalten. Weitere Informationen zur *B-Baumstruktur-Indexierung* finden Sie in der Dokumentation zu DB2 UDB.

Beachten Sie beim Erstellen eines Indexes folgende Aspekte:

- 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 Nebentabelle erstellen.
- Sie können mehrere Indizes zu einer XML-Spalte erstellen.
- Sie können Nebentabellen über die ROOT-ID zuordnen. ROOT-ID ist der Spaltenname des Primärschlüssels in der Anwendungstabelle und eine eindeutige Kennung, die alle Nebentabellen 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 XML Extender die Anwendungstabelle und fügt eine Spalte DXXROOT_ID hinzu, in der eine eindeutige ID gespeichert wird, die beim Einfügen erstellt wird. Alle Nebentabellen enthalten eine Spalte DXXROOT_ID mit der eindeutigen ID. Wenn der Primärschlüssel als ROOT-ID verwendet wird, haben alle Nebentabellen 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 DB2 UDB Net Search Extender aktivieren, können Sie auch die Funktion für strukturellen Text von Net Search Extender verwenden. Net Search Extender unterstützt die *Abschnittsuche* ("section search"), die die Möglichkeit einer konventionellen Volltextsuche erweitert und den Vergleich von Suchbegriffen in einem spezifischen Dokumentkontext ermöglicht, der über Standortpfade angegeben wird. Der *Index für den strukturellen Text* kann mit der Indexierung von XML Extender bei allgemeinen SQL-Datentypen verwendet werden.

XML-Daten speichern

Mit XML Extender können Sie intakte XML-Dokumente in eine XML-Spalte einfügen. Wenn Sie Nebentabellen definieren, aktualisiert XML Extender diese Tabellen automatisch. Wenn Sie ein XML-Dokument direkt speichern, speichert XML Extender den Basistyp als XML-Typ.

Voraussetzungen:

- Stellen Sie sicher, dass Sie die DAD-Datei erstellt bzw. aktualisiert haben.
- Stellen Sie fest, welcher Datentyp beim Speichern des Dokuments verwendet werden soll.
- Wählen Sie eine Methode (Umsetzungsfunktionen oder UDFs) zum Speichern der Daten in der DB2-Tabelle aus.

Geben Sie eine SQL-Anweisung INSERT an, die die XML-Tabelle und die die Spalte angibt, die das XML-Dokument enthalten soll.

XML Extender bietet zwei Methoden zum Speichern von XML-Dokumenten: Standardumsetzungsfunktionen und Speicher-UDFs.

Tabelle 7 zeigt, in welchen Fällen diese Methoden verwendet werden sollten.

Tabelle 7. Die XML Extender-Speicherfunktionen

DB2 UDB-Basistyp ist...	**Speichern in DB2 UDB als...**			
	XMLVARCHAR	**XMLCLOB**	**XMLDBCLOB**	**XMLFILE**
VARCHAR	XMLVARCHAR()	n/v	n/v	XMLFile FromVarchar()
CLOB	n/v	XMLCLOB()	XMLDB CLOB, Umsetzungs-funktion	XMLFile FromCLOB()
FILE	XMLVarchar FromFile()	XMLCLOB FromFile()	XMLDB CLOBFrom File, UDF	XMLFILE

Standardumsetzungsfunktionen für das Speichern von XML-Daten

Für jeden UDT gibt es eine Standardumsetzungsfunktion zur Umsetzung des SQL-Basistyps in den UDT. Sie können die von XML Extender bereitgestellten Umsetzungsfunktionen in Ihrer Klausel VALUES verwenden, um Daten einzufügen. Tabelle 8 zeigt die bereitgestellten Umsetzungsfunktionen:

Tabelle 8. Standardumsetzungsfunktionen von XML Extender

Umsetzungsfunktion	**Rückgabetyp**	**Beschreibung**
XMLVARCHAR(VARCHAR)	XMLVARCHAR	Eingabe aus dem Speicher-puffer als VARCHAR
XMLCLOB(CLOB)	XMLCLOB	Eingabe aus dem Speicher-puffer als CLOB oder CLOB-Zeigereinheit
XMLFILE(VARCHAR)	XMLFILE	Nur Speicherung des Datei-namens

Beispielsweise fügt die folgende Anweisung einen Umsetzungsausdruck vom Typ VARCHAR in den Typ XMLVARCHAR ein:

```
INSERT INTO sales_tab
VALUES('123456', 'Sriram Srinivasan', DB2XML.XMLVarchar(:xml_buff))
```

Speicher-UDFs für das Speichern von XML-Daten

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 ein XML-Dateidokument als XMLCLOB-Datentyp in DB2 UDB importieren wollen, können Sie die Funktion XMLCLOBFromFile() verwenden.

Tabelle 9 zeigt die von XML Extender bereitgestellten Speicherfunktionen.

Tabelle 9. Die XML Extender-Speicher-UDFs

Benutzerdefinierte Speicherfunktion	Rückgabetyp	Beschreibung
XMLVarcharFromFile()	XMLVARCHAR	Liest ein XML-Dokument aus einer Datei auf dem Server ein und gibt den Wert des Datentyps XMLVARCHAR zurück. Optional: Geben Sie die Codierung der Datei an.
XMLCLOBFromFile()	XMLCLOB	Liest ein XML-Dokument aus einer Datei auf dem Server ein und gibt den Wert des Datentyps XMLCLOB zurück. Optional: Geben Sie die Codierung der Datei an.
XMLFileFromVarchar()	XMLFILE	Liest ein XML-Dokument aus dem Speicher als VARCHAR-Daten ein, schreibt es in eine externe Datei und gibt den Wert des Datentyps XMLFILE zurück; dieser Wert ist der Dateiname. Optional: Geben Sie die Codierung der externen Datei an.
XMLFileFromCLOB()	XMLFILE	Liest ein XML-Dokument aus dem Speicher als CLOB-Daten oder als CLOB-Querverweis ein, schreibt es in eine externe Datei und gibt den Wert des Datentyps XMLFILE zurück; dieser Wert ist der Dateiname. Optional: Geben Sie die Codierung der externen Datei an.

Beispielsweise speichert die folgende Anweisung unter Verwendung der Funktion XMLCLOBFromFile() einen Datensatz in einer XML-Tabelle als XMLCLOB:

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
    VALUES('1234', 'MyName',
      XMLCLOBFromFile('dxx_install_verz/samples/db2xml/xml/getstart.xml'))
```

In diesem Beispiel wird das XML-Dokument aus der Datei mit dem Namen `dxx_install_verz/samples/db2xml/xml/getstart.xml` in die Spalte ORDER in der Tabelle SALES_TAB importiert.

Methode zum Abrufen eines XML-Dokuments

Mit XML Extender können Sie entweder ein vollständiges Dokument oder den Inhalt von Elementen und Attributen abrufen. Wenn Sie eine XML-Spalte direkt abrufen, gibt XML Extender den UDT als Spaltentyp zurück. Ausführliche Informationen zum Abrufen von Daten finden Sie in den folgenden Abschnitten:

- „Vollständiges XML-Dokument abrufen"
- „Elementinhalte und Attributwerte aus XML-Dokumenten abrufen" auf Seite 87

XML Extender bietet zwei Methoden zum Abrufen von Daten: Standardumsetzungsfunktionen und die überladene UDF Content(). Tabelle 10 zeigt, in welchen Fällen die beiden Methoden verwendet werden sollten.

Tabelle 10. Die Abruffunktionen von XML Extender

Der XML-Typ ist...	**Aus DB2 UDB abrufen als ...**			
	VARCHAR	**CLOB**	**DBCLOB**	**FILE**
XMLVARCHAR	**VARCHAR**	n/v	n/v	Content(), UDF
XMLCLOB	n/v	**XMLCLOB**	n/v	Content(), UDF
XMLFILE	n/v	Content(), UDF	n/v	FILE

Vollständiges XML-Dokument abrufen

Vorgehensweise

So rufen Sie ein vollständiges XML-Dokument ab:

1. Stellen Sie sicher, dass Sie das XML-Dokument in einer XML-Tabelle gespeichert haben, und legen Sie fest, welche Daten abgerufen werden sollen.
2. Wählen Sie eine Methode (Umsetzungsfunktionen oder UDFs) zum Abrufen der Daten in der DB2 UDB-Tabelle aus.
3. Wenn Sie die überladene UDF Content() verwenden, stellen Sie den Typ der Daten fest, die abgerufen werden sollen, und welcher Datentyp exportiert werden soll.
4. Die XML-Spalte, aus der das Element oder Attribut extrahiert werden soll, muss vom Datentyp XMLVARCHAR, XMLCLOB as LOCATOR oder XMLFILE sein.

Geben Sie eine SQL-Abfrage an, die die XML-Tabelle und -Spalte angibt, aus der das XML-Dokument abgerufen werden soll.

Standardumsetzungsfunktionen für das Abrufen von XML-Daten

Die Standardumsetzungsfunktion, die von DB2 UDB für UDTs bereitgestellt wird, setzt einen XML-UDT in einen SQL-Basistyp um und arbeitet anschließend damit. In Ihrer Anweisung SELECT können Sie die Umsetzungsfunktionen, die von XML Extender bereitgestellt werden, zum Abrufen von Daten verwenden. Tabelle 11 auf Seite 86 zeigt die bereitgestellten Umsetzungsfunktionen.

Tabelle 11. Die Standardumsetzungsfunktionen von XML Extender

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

Beispielsweise ruft die folgende Anweisung XMLVARCHAR ab und speichert es als Datentyp VARCHAR:

```
EXEC SQL SELECT DB2XML.XMLVarchar(order) from SALES_TAB
```

UDF Content() zum Abrufen von XML-Daten 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, d. h. eine Datei, die sich außerhalb von DB2 UDB auf dem DB2 UDB-Server befindet.

Beispielsweise haben Sie Ihr XML-Dokument als Datentyp XMLFILE gespeichert. Wenn Sie damit im Hauptspeicher arbeiten wollen, können Sie die UDF Content() verwenden, die den Datentyp XMLFILE als Eingabe nimmt und CLOB zurückgibt.

Die UDF Content() führt je nach dem angegebenen Datentyp zwei verschiedene Abruffunktionen aus:

- Die UDF Content() ruft ein Dokument aus dem Zusatzspeicher ab und legt es im Hauptspeicher ab.

  Sie können mit der UDF Content() das XML-Dokument in einen Speicherpuffer oder einen CLOB-*Querverweis* (eine Host-Variable mit einem Wert, der einen einzelnen LOB-Wert im Datenbankserver darstellt) abrufen, wenn das Dokument als externe Datei gespeichert ist.

  Verwenden Sie die folgende Funktionssyntax, wobei *xmlobj* die abgefragte XML-Spalte ist:

  XMLFILE in CLOB:

  ```
  Content(xmlobj XMLFile)
  ```

- Die UDF Content() ruft ein Dokument aus dem internen Speicher ab und exportiert es in eine externe Datei.

  Sie können mit der UDF Content() ein XML-Dokument abrufen, das als Datentyp XMLCLOB in DB2 UDB gespeichert ist, und es in eine Datei auf dem Dateisystem des Datenbankservers exportieren. Die UDF Content() gibt den Namen der Datei als Datentyp VARCHAR zurück.

  Verwenden Sie die folgende Syntax:

  XML-Typ in externe Datei:

  ```
  Content(xmlobj XML type, dateiname varchar(512), zielcodierung varchar(100))
  ```

  Hierbei gilt Folgendes:

  xmlobj Der Name der XML-Spalte, aus der der XML-Inhalt abgerufen werden soll. *xmlobj* kann vom Typ XMLVARCHAR oder XMLCLOB sein.

  dateiname
  Der Name der externen Datei, in der die XML-Daten gespeichert werden sollen.

| *zielcodierung*
| Optional: Gibt die Codierung der Ausgabedatei an.

Im nachfolgenden Beispiel zeigt ein kleines C-Programmsegment mit eingebetteten SQL-Anweisungen (SQL-Anweisungen, die innerhalb eines Anwendungsprogramms codiert sind), wie ein XML-Dokument aus einer Datei in den Hauptspeicher abgerufen wird. In diesem Beispiel wird davon ausgegangen, dass XMLFILE der Datentyp der Spalte ORDER ist.

```
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 { /* gewünschte Aktion mit XML-Dokument im Puffer */}
       }
    EXEC SQL CLOSE c1;
    EXEC SQL CONNECT RESET;
```

Elementinhalte und Attributwerte aus XML-Dokumenten abrufen

Sie können den Inhalt eines Elements oder den Wert eines Attributs aus einem oder mehreren XML-Dokumenten (Suche in einem einzelnen Dokument oder in einer Dokumentgruppe) abrufen (extrahieren). XML Extender bietet benutzerdefinierte Funktionen zum Extrahieren, die Sie in der SELECT-Klausel für jeden SQL-Datentyp angeben können.

Das Abrufen von Elementinhalten und Attributwerten 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. Um die Namen aller Kunden abzurufen, die Artikel zu einem Preis von mehr als 2500 EURO bestellt haben, verwenden Sie die folgende SQL-Anweisung mit der Extraktions-UDF in der SELECT-Klausel:

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
        WHERE price > 2500.00
```

In diesem Beispiel ruft die Extraktions-UDF den Inhalt des Elements <customer> aus der Spalte ORDER ab und speichert es als Datentyp VARCHAR. Der Standortpfad ist `/Order/Customer/Name`. Darüber hinaus wird die Anzahl der zurückgegebenen Werte mit Hilfe einer WHERE-Klausel verringert. Diese Klausel gibt an, dass nur der Inhalt des Elements <customer> abgerufen werden soll, dessen Unterelement <ExtendedPrice> einen größeren Wert als 2500.00 hat.

Tabelle 12 auf Seite 88 zeigt die UDFs, die Sie zum Extrahieren von Elementinhalten und Attributwerten verwenden können, wobei die folgende Syntax entweder für Tabellenfunktionen oder Skalarfunktionen verwendet wird.

Syntax:

`extract` *`abgerufener_datentyp`*`(`*`xmlobj`*`,` *`pfad`*`)`

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

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`).

Einschränkung: Extraktions-UDFs können Standortpfade unterstützen, die Prädikate mit Attributen haben, nicht aber 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 12 zeigt die Extraktionsfunktionen im skalaren und tabellarischen Format:

Tabelle 12. Die Extraktionsfunktionen von XML Extender

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

Beispiel für Skalarfunktion: 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 umgesetzt.

```
CREATE TABLE t1(key decimal(3,2));
INSERT into t1 values
SELECT * from table(DB2XML.extractInteger(DB2XML.XMLFile
   ('c:\dxx_install_verz\samples\db2xml\xml\getstart.xml'), '/Order/@Key="1"]'));
SELECT * from t1;
```

XML-Daten aktualisieren

Mit 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.

Vorgehensweise

So aktualisieren Sie XML-Daten:

1. Das XML-Dokument muss in einer XML-Tabelle gespeichert werden.
2. Der Benutzer muss wissen, welche Daten abgerufen werden sollen.
3. Der Benutzer muss eine Methode zum Aktualisieren der Daten in der DB2 UDB-Tabelle auswählen (Umsetzungsfunktionen oder UDFs).
4. Geben Sie eine SQL-Abfrage an, die die zu aktualisierende XML-Tabelle und -Spalte angibt.

Vollständiges XML-Dokument aktualisieren

Sie können ein XML-Dokument aktualisieren, indem Sie die Standardumsetzungsfunktion oder eine Speicher-UDF verwenden.

Mit einer Standardumsetzungsfunktion aktualisieren

Für jeden benutzerdefinierten Typ (UDT) gibt es eine Standardumsetzungsfunktion zur Umsetzung des SQL-Basistyps in den UDT. Sie können die von XML Extender bereitgestellten Umsetzungsfunktionen zum Aktualisieren des XML-Dokuments verwenden.

Beispielsweise aktualisiert die folgende Anweisung den Typ XMLVARCHAR aus dem umgesetzten Typ VARCHAR, wobei davon ausgegangen wird, dass `xml_buff` eine Host-Variable ist, die als Typ VARCHAR definiert ist.

```
UPDATE sales_tab SET=DB2XML.XMLVarchar(:xml_buff)
```

XML-Dokumente mit einer Speicher-UDF aktualisieren

Für jeden XML Extender-UDT gibt es eine Speicher-UDF zum Importieren von Daten in DB2 UDB aus einer Ressource, die sich vom Basistyp unterscheidet. Sie können mit einer Speicher-UDF das gesamte XML-Dokument aktualisieren, indem Sie es ersetzen.

Das folgende Beispiel aktualisiert das XML-Objekt aus der Datei mit dem Namen `dxx_install_verz/samples/db2xml/xml/getstart.xml` in die Spalte ORDER in der Tabelle SALES_TAB.

```
UPDATE sales_tab
       set order = XMLVarcharFromFile('dxx_install_verz/samples/db2xml
       /xml/getstart.xml) WHERE sales_person = 'MyName'
```

Spezifische Elemente und Attribute eines XML-Dokuments aktualisieren

Verwenden Sie die UDF Update, um spezifische Änderungen vorzunehmen, statt das gesamte Dokument zu aktualisieren. Bei Verwendung dieser UDF geben Sie den Standortpfad des Elements oder Attributs an, dessen Wert ersetzt wird. Sie müssen das XML-Dokument nicht editieren; XML Extender nimmt die Änderungen für Sie vor.

Syntax:

`Update(`*`xmlobj`*`, `*`pfad`*`, `*`wert`*`)`

Hierbei gilt Folgendes:

xmlobj Der Name der XML-Spalte, für die der Wert des Elements bzw. Attributs aktualisiert werden soll.

pfad Der Standortpfad des zu aktualisierenden Elements bzw. Attributs.

wert Der neue Wert des zu aktualisierenden Werts.

Beispielsweise ersetzt die folgende Anweisung den Wert des Elements <Customer> durch 'IBM':

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
   WHERE sales_person = 'Sriram Srinivasan'
```

Mehrfaches Vorkommen: Wenn Sie einen Standortpfad in der UDF Update angeben, wird der Inhalt jedes Elements oder Attributs mit einem entsprechenden Pfad durch den angegebenen Wert ersetzt. Wenn ein Standortpfad mehr als einmal in einem Dokument vorkommt, ersetzt die UDF Update alle vorhandenen Werte durch den Wert in dem Parameter *wert*.

Methoden zum Durchsuchen von XML-Dokumenten

Das Durchsuchen von XML-Daten ähnelt dem Abrufen von XML-Daten: mit beiden Techniken werden Daten zur weiteren Bearbeitung abgerufen, aber die Suche erfolgt, indem der Inhalt der WHERE-Klausel als Abrufkriterium verwendet wird.

XML Extender bietet verschiedene Methoden zum Durchsuchen der XML-Dokumente, die in einer XML-Spalte gespeichert sind. Folgendes ist möglich:

- Durchsuchen der Dokumentstruktur und Rückgabe der Ergebnisse entsprechend dem Elementinhalt oder den Attributwerten.
- Durchsuchen einer Sicht der XML-Spalte und deren Nebentabellen
- Direktes Durchsuchen der Nebentabellen aus Gründen der Leistung
- Durchsuchen mit Hilfe von Extraktions-UDFs mit WHERE-Klauseln
- Verwenden von DB2® Net Search Extender zum Durchsuchen von Spaltendaten im strukturellen Inhalt nach einer Zeichenfolge

Mit XML Extender können Sie Indizes verwenden, um Spalten in Nebentabellen schnell zu durchsuchen. Diese Spalten enthalten XML-Elementinhalte oder Attributwerte, die aus XML-Dokumenten extrahiert wurden. Durch die Angabe des Datentyps eines Elements oder Attributs können Sie nach einem SQL-Datentyp suchen oder Bereiche durchsuchen. Im Beispiel zu den Bestellungen könnten Sie etwa nach allen Bestellungen suchen, deren Preis höher ist als 2500,00.

Darüber hinaus können Sie mit Net Search Extender eine strukturelle Textsuche oder eine Volltextsuche durchführen. Sie können beispielsweise eine Spalte mit dem Namen RESUME haben, die Bewerbungen zur Wiedervorlage im XML-Format enthält. Wenn Sie die Namen aller Bewerber mit Java™-Kenntnissen suchen wollen, können Sie mit DB2 Net Search Extender die XML-Dokumente nach allen Lebensläufen durchsuchen, in denen das Element <skill> die Zeichenfolge „JAVA" enthält.

Die folgenden Abschnitte beschreiben die Suchmethoden:

- „XML-Dokument nach Struktur durchsuchen"
- „DB2 UDB Net Search Extender für strukturelle Textsuchvorgänge in XML-Dokumenten verwenden" auf Seite 93

XML-Dokument nach Struktur durchsuchen

Mit den Suchfunktionen von XML Extender können Sie XML-Daten in einer Spalte auf der Basis der Dokumentstruktur (d. h. nach Elementen und Attributen des Dokuments) durchsuchen.

Prozeduren:

Zum Durchsuchen der Daten haben Sie folgende Möglichkeiten:

- Direkte Abfrage der Nebentabellen
- Verwenden einer *verknüpften Sicht*
- Verwenden von Extraktions-UDFs

Diese Suchmethoden werden in den folgenden Beispielen beschrieben. Als Basis dient das folgende Szenario. Die Tabelle SALES_TAB enthält eine XML-Spalte mit dem Namen ORDER. Diese Spalte enthält drei Nebentabellen ORDER_SIDE_TAB, PART_SIDE_TAB und SHIP_SIDE_TAB. Eine Standardsicht, sales_order_view, wurde beim Aktivieren der Spalte ORDER angelegt. Diese Sicht 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
```

Beispiel: Nebentabellen mit direkter Abfrage durchsuchen

Eine Direktabfrage mit einer Unterabfragensuche bietet den besten Durchsatz für eine strukturelle Suche, wenn die Nebentabellen indexiert sind.

Vorgehensweise

Für ein korrektes Durchsuchen der Nebentabellen können Sie eine Abfrage oder Unterabfrage verwenden.

Beispielsweise verwendet die folgende Anweisung eine Abfrage und eine Unterabfrage, um eine Nebentabelle 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.

Beispiel: Über eine verknüpfte Sicht suchen

XML Extender kann eine Standardsicht erstellen, die die Anwendungstabelle und die Nebentabellen mit einer eindeutigen ID verknüpft. Mit dieser Standardsicht oder jeder anderen Sicht, die eine Anwendungstabelle und Nebentabellen verknüpft, können Sie Spaltendaten durchsuchen und die Nebentabellen abfragen. Diese Methode bietet eine einzelne virtuelle Sicht der Anwendungstabelle und ihrer Nebentabellen. Je mehr Nebentabellen erstellt werden, desto zeitaufwendiger ist allerdings die Abfrage.

Tipp: Sie können die Root-ID oder DXXROOT_ID, die von XML Extender erstellt wurde, verwenden, um die Tabellen beim Erstellen Ihrer eigenen Sicht zu verknüpfen.

Beispielsweise durchsucht die folgende Anweisung die Sicht mit dem Namen SALES_ORDER_VIEW und gibt die Werte aus der Spalte SALES_PERSON zurück, bei denen der Preis der Bestellposition höher ist als 2500,00.

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

Beispiel: Mit Extraktions-UDFs suchen

Sie können auch die Extraktions-UDFs von XML Extender zum Suchen nach Elementen und Attributen verwenden, wenn Sie keine Indizes oder Nebentabellen für die Anwendungstabelle erstellt haben. Die Verwendung der Extraktions-UDFs zum Durchsuchen der XML-Daten ist aufwendig und sollte nur in Verbindung mit WHERE-Klauseln verwendet werden, die die Anzahl der bei der Suche berücksichtigten XML-Dokumente begrenzen.

Die folgende Anweisung sucht mit einer Extraktions-UDF von 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 Extraktions-UDF die Elemente </Order/Customer/Name>, die die Unterzeichenfolge IBM® enthalten.

Beispiel: 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.

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 die Nebentabelle 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.

DB2 UDB Net Search Extender für strukturelle Textsuchvorgänge in XML-Dokumenten verwenden

Wenn DB2 UDB Net Search Extender installiert ist, können Sie ihn für eine strukturelle Textsuche verwenden.

Vorgehensweise

Gehen Sie wie folgt vor, um DB2 UDB Net Search Extender zu verwenden:

1. Entscheiden Sie, ob Sie die strukturelle Textsuche oder die Volltextsuche verwenden wollen.
2. Aktivieren Sie eine XML-Spalte für DB2 UDB Net Search Extender.
3. Erstellen Sie eine Abfrage, um die Suche durchzuführen.

Um die Verwendung der Suche mit DB2 UDB Net Search Extender zu lernen, siehe DB2 Universal Database Extender: Net Search Extender Verwaltung und Programmierung, Version 7.

Strukturelle Textsuch- und Volltextsuchvorgänge verwenden

Beim Durchsuchen der XML-Dokumentstruktur durchsucht XML Extender Elemente, die in allgemeine Datentypen umgesetzt sind; er sucht jedoch nicht nach Textelementen. Sie können Net Search Extender für eine strukturelle Textsuche oder eine Volltextsuche in einer für XML aktivierten Spalte verwenden. DB2 UDB Net Search Extender unterstützt die XML-Dokumentsuche ab DB2 UDB Version 6.1. Net Search Extender ist für die Betriebssysteme AIX®, Windows®, iSeries sowie für die Solaris™-Betriebsumgebung verfügbar.

Strukturelle Textsuche
: Sucht nach Textzeichenfolgen, die auf der Baumstruktur des XML-Dokuments aufbauen. Sie können beispielsweise in einer Dokumentstruktur von /Order/Customer/Name eine strukturelle Textsuche verwenden, um die Zeichenfolge "IBM" innerhalb des Unterelements <Customer> zu suchen. Das Dokument kann jedoch die Zeichenfolge "IBM" in einem Unterelement <Comment> oder als Teil des Namens eines Produkts enthalten. Eine strukturelle Textsuche sucht nur in dem angegebenen Element nach der Zeichenfolge. In diesem Beispiel werden nur die Dokumente gefunden, die "IBM" im Unterelement </Order/Customer/Name> enthalten; ein Dokument, das "IBM" in anderen Elementen enthält, aber nicht im Unterelement </Order/Customer/Name>, wird 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.

XML-Spalte für DB2 UDB Net Search Extender aktivieren

In einer für XML aktivierten Datenbank aktivieren Sie DB2 UDB Net Search Extender, um den Inhalt einer für XML aktivierten Spalte zu durchsuchen.

Für dieses Beispiel erhält die Datenbank den Namen SALES_DB, die Tabelle den Namen ORDER, und die XML-Spaltennamen lauten XVARCHAR und XCLOB.

1. Die Datei `install.txt` auf der DB2 UDB Extender-CD enthält Informationen zur Installation von Net Search Extender.
2. Führen Sie den Befehl **txstart** aus:
   - Bei UNIX-Betriebssystemen geben Sie den Befehl über die Eingabeaufforderung des Exemplareigners ein.
   - Geben Sie für Windows NT® den Befehl in dem Befehlsfenster ein, in dem DB2INSTANCE angegeben wurde.
3. Öffnen Sie das Befehlszeilenfenster von Net Search Extender, und stellen Sie eine Verbindung zur Datenbank her. Geben Sie in der Eingabeaufforderung **db2tx** Folgendes ein:

   ```
   connect to SALES_DB
   ```
4. Aktivieren Sie die Datenbank für DB2 UDB Net Search Extender.

   Geben Sie in der Eingabeaufforderung **db2tx** Folgendes ein:

   ```
   enable database
   ```
5. Aktivieren Sie die Spalten in der XML-Tabelle für DB2 UDB Net Search Extender. Definieren Sie die Datentypen des XML-Dokuments, die Sprache, die Codepages und andere Informationen zu der Spalte.
   - Geben Sie für die VARCHAR-Spalte XVARCHAR Folgendes ein:

     ```
     enable text column order xvarchar function
     db2xml.varchartovarchar handle varcharhandle ccsid 1252
     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 1252
     language us_english indextype precise updateindex update
     ```
6. Überprüfen Sie den Status des Indexes.
   - Geben Sie für die XVARCHAR-Spalte Folgendes ein:

     ```
     get index status order handle varcharhandle
     ```
   - Geben Sie für die XCLOB-Spalte Folgendes ein:

     ```
     get index status order handle clobhandle
     ```
7. Definieren Sie das XML-Dokumentmodell in einer Initialisierungsdatei für das Dokumentmodell mit dem Namen `desmodel.ini`. Diese Datei befindet sich unter UNIX im Verzeichnis `/db2tx/txins000` und unter Windows NT im Verzeichnis `/instance/db2tx/txins000`. 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/Shipment/ShipMode = /Order/Part/Shipment/ShipMode
   ```

Mit dem DB2 UDB Net Search Extender nach Text suchen

Um mit DB2 UDB Net Search Extender nach Text zu suchen, erstellen Sie eine Abfrage, die das Element oder das Attribut angibt, nach dem gesucht werden soll. DB2 UDB Net Search Extender verwendet diese Abfrage anschließend, um nach dem Elementinhalt oder den Attributwerten zu suchen.

Geben Sie beispielsweise die folgenden Anweisungen in ein DB2 UDB-Befehlsfenster ein, um mit DB2 UDB Net Search Extender den Text eines XML-Dokuments zu durchsuchen:

```
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() von Net Search Extender sucht im Text eines XML-Dokuments nach dem Suchbegriff.

Dieses Beispiel enthält nicht alle Schritte, die zum Durchsuchen von Spaltendaten mit DB2 UDB Net Search Extender erforderlich sind. Im Handbuch *IBM DB2 Universal Database Net Search Extender Verwaltung und Benutzerhandbuch* finden Sie weitere Informationen über die Suchfunktion und das Konzept von Net Search Extender.

XML-Dokumente löschen

Verwenden Sie die SQL-Anweisung DELETE, um die Zeile mit einem XML-Dokument aus einer XML-Spalte zu löschen. Sie können eine WHERE-Klausel angeben, um bestimmte Dokumente zu löschen.

Beispielsweise löscht die folgende Anweisung alle Dokumente, die für <Extended-Price> 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)
```

Die entsprechenden Zeilen in den Nebentabellen werden automatisch gelöscht.

Zugehörige Konzepte:

• „XML-Spalten als Speicher- und Zugriffsmethode” auf Seite 80

Zugehörige Tasks:

• „Daten in XML-Spalten verwalten” auf Seite 79

Einschränkungen beim Aufruf von Funktionen von JDBC (Java Database Connectivity) 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.

Beispielsweise 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))
```

Beispiele:

In den folgenden Beispielen weist die Tabelle `Sales_Tab` drei Spalten auf. Die Spalte `invoice_num` weist den Datentyp Char(6) auf, die Spalte `sales_person` den Datentyp Varchar(20) und die Spalte `order` den Datentyp XMLVarchar.

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 4. Daten in XML-Objektgruppen verwalten

XML-Objektgruppen als Speicher- und Zugriffsmethode

Relationale Daten werden entweder aus ankommenden XML-Dokumenten *zerlegt* oder für ausgehende XML-Dokumente *zusammengesetzt*. Zerlegte Daten sind der nicht markierte Inhalt eines XML-Dokuments, die in einer oder mehreren Datenbanktabellen gespeichert sind. Bzw. XML-Dokumente sind aus bestehenden Daten in einer oder mehreren Datenbanktabellen zusammengesetzt. Wenn Ihre Daten mit anderen Anwendungen gemeinsam verwendet werden sollen, wollen Sie eingehende und abgehende XML-Dokumente 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. 10.

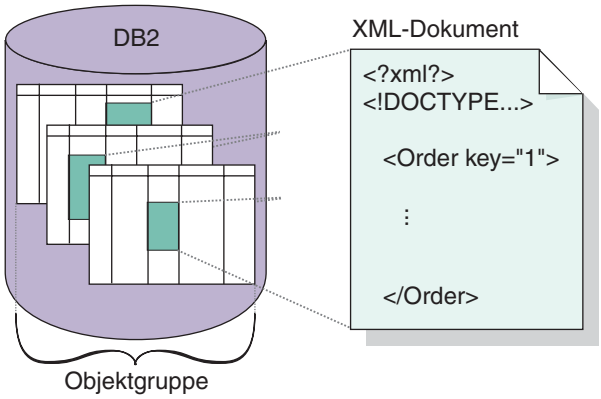


Abbildung 10. Speichern von Dokumenten als nicht markierte Daten in DB2 UDB-Tabellen

Die XML-Objektgruppe ist in einer DAD-Datei definiert; diese Datei gibt an, wie Elemente und Attribute einer oder mehreren relationalen Tabellen zugeordnet werden. Die Objektgruppe ist eine Gruppe von Spalten, die einer DAD-Datei zugeordnet ist, die die Daten in einem bestimmten XML-Dokument oder einer Gruppe von XML-Dokumenten enthält. Sie können einen Objektgruppennamen durch Aktivieren definieren und anschließend diesen Namen verwenden, wenn Sie eine gespeicherte Prozedur zum Zusammensetzen oder Zerlegen von XML-Dokumenten aufrufen. Dabei handelt es sich um eine aktivierte XML-Objektgruppe. Der Objektgruppe wird ein Name gegeben, so dass sie leicht mit gespeicherten Prozeduren ausgeführt werden kann, die die XML-Dokumente zusammensetzen oder zerlegen.

Wenn Sie eine Objektgruppe in der DAD-Datei definieren, verwenden Sie eine von zwei Arten von Zuordnungsschemata, *SQL-Zuordnung* oder *RDB-Knotenzuordnung*, mit denen die Tabellen, Spalten und Bedingungen definiert werden, die zur Zuordnung von XML-Daten zu DB2 UDB-Tabellen verwendet werden. Die SQL-Zuordnung verwendet SQL SELECT-Anweisungen zum Definieren der DB2 UDB-Tabellen und Bedingungen für die Objektgruppe. Die RDB-Knotenzuordnung verwendet einen XPath-basierten relationalen Datenbankknoten oder einen RDB-Knoten, der untergeordnete Elemente besitzt.

Gespeicherte Prozeduren stehen zum Zusammensetzen oder Zerlegen von XML-Dokumenten zur Verfügung. Die Namen von gespeicherten Prozeduren werden durch DB2XML qualifiziert, dem *Schemanamen* von XML Extender.

Daten in XML-Objektgruppen 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, die die Objektgruppe bilden, können neue Tabellen sein oder vorhandene Tabellen mit Daten, die mit XML Extender verwendet werden, um XML-Dokumente für Ihre Anwendungen zusammenzusetzen. Spaltendaten in diesen Tabellen enthalten keine XML-Befehle, sondern nur den Inhalt und die Werte, die den Elementen und Attributen zugeordnet sind. Sie verwenden gespeicherte Prozeduren zum Speichern, Abrufen, Aktualisieren, Löschen und zum Durchsuchen von XML-Objektgruppendaten.

Vorbereitungen zum Zusammensetzen von XML-Dokumenten aus DB2-Daten

"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. Um diese gespeicherten Prozeduren verwenden zu können, erstellen Sie eine DAD-Datei (Dokumentzugriffsdefinitionsdatei). Eine DAD-Datei gibt die Zuordnung zwischen dem XML-Dokument und der DB2-Tabellenstruktur an. Die gespeicherten Prozeduren verwenden die DAD-Datei zum Zusammensetzen des XML-Dokuments.

Vorgehensweise:

Führen Sie die folgenden Aktionen aus, bevor Sie mit dem Zusammensetzen von XML-Dokumenten beginnen:

1. Ordnen Sie die Struktur des XML-Dokuments den relationalen Tabellen zu, die den Inhalt der Element- und Attributwerte enthalten.
2. Wählen Sie eine Zuordnungsmethode aus: SQL-Zuordnung oder RDB_node-Zuordnung.
3. Bereiten Sie die DAD-Datei vor.

XML Extender bietet vier gespeicherte Prozeduren, dxxGenXML(), dxxGenXMLCLOB(), dxxRetrieveXML() und dxxRetrieveXMLCLOB, zum Zusammensetzen von XML-Dokumenten. Die Häufigkeit, in der das XML-Dokument aktualisiert werden soll, ist ein Hauptfaktor bei der Auswahl der zu verwendenden gespeicherten Prozedur.

XML-Dokumente zusammensetzen, die gelegentlich aktualisiert werden

Wenn Ihr Dokument nur gelegentlich aktualisiert wird, verwenden Sie die gespeicherte Prozedur dxxGenXML zum Zusammensetzen des Dokuments. Sie müssen keine Objektgruppe aktivieren, um diese gespeicherte Prozedur verwenden zu können. Die gespeicherte Prozedur verwendet stattdessen eine DAD-Datei.

Die gespeicherte Prozedur dxxGenXML() erstellt 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 eine 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 und verfügt immer über eine Spalte vom Typ VARCHAR, CLOB, XMLVARCHAR oder XMLCLOB, der zum Speichern der XML-Daten verwendet wird.

Darüber hinaus fügt XML Extender die Spalte DXX_VALID vom Typ INTEGER in die Ergebnistabelle ein, sofern diese Spalte noch nicht in der Tabelle enthalten ist. Dies ist nur der Fall, wenn der Wert des Elements für die Gültigkeitsprüfung in der DAD-Datei auf YES gesetzt ist. XML Extender fügt den Wert 1 für ein gültiges XML-Dokument und den Wert 0 für ein ungültiges Dokument ein.

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 sowie alle eventuellen Rückkehrcodes und Nachrichten zurück.

Die entsprechende gespeicherte Prozedur für die Zerlegung ist dxxShredXML; sie verwendet ebenfalls die DAD-Datei als Eingabeparameter. Für diese Prozedur muss die XML-Objektgruppe nicht aktiviert sein.

Vorgehensweise

Um eine XML-Objektgruppe mit der gespeicherten Prozedur dxxGenXML zusammenzusetzen, integrieren Sie den Aufruf einer gespeicherten Prozedur in Ihrer Anwendung mit der folgenden Deklaration einer gespeicherten Prozedur:

```
dxxGenXML(CLOB(100K)     DAD,               /* Eingabe */
          char(32 resultTabName) resultTabName,     /* input */

               integer        overrideType,      /* Eingabe */
               varchar(1024)  override,          /* Eingabe */
               integer        maxRows,           /* Eingabe */
               integer        numRows,           /* Ausgabe */
               long           returnCode,        /* Ausgabe */
               varchar(1024)  returnMsg)         /* Ausgabe */
```

Beispiel: Im 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;            /* tatsächliche Anzahl Zeilen */
long  returnCode;         /* Rückkehrfehlercode */
char  returnMsg[1024];    /* Fehlernachrichtentext */
short dad_ind;
short rtab_ind;
short ovtype_ind;
short ov_ind;
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);
```

```
     /* Daten aus einer Datei in ein CLOB einlesen */
     strcpy(dadfile.name,"dxx_install_verz
      /samples/dad/getstart_xcollection.dad");
     dadfile.name_length = strlen("dxx_install_verz
     /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';
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 db2xml.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);
```

Nachdem die gespeicherte Prozedur aufgerufen wurde, enthält die Ergebnistabelle 250 Zeilen, da die in der DAD-Datei angegebene SQL-Abfrage 250 XML-Dokumente generiert hat.

XML-Dokumente zusammensetzen, die häufig aktualisiert werden

Wenn Ihr Dokument häufig aktualisiert wird, verwenden Sie die gespeicherte Prozedur dxxRetrieveXML zum Zusammensetzen des Dokuments. Da dieselben Tasks wiederholt ausgeführt werden, ist die Verbesserung der Leistung von großer Bedeutung.

Die gespeicherte Prozedur dxxRetrieveXML funktioniert wie die gespeicherte Prozedur dxxGenXML mit dem Unterschied, dass sie den Namen einer aktivierten XML-Objektgruppe an Stelle einer DAD-Datei verwendet. Wenn eine XML-Objektgruppe aktiviert ist, wird eine DAD-Datei in der Tabelle XML_USAGE gespeichert. Daher ruft XML Extender die DAD-Datei ab und verwendet sie genauso zum Zusammensetzen des Dokuments wie die gespeicherte Prozedur dxxGenXML.

Die gespeicherte Prozedur dxxRetrieveXML ermöglicht die Verwendung derselben DAD-Datei für die Zusammensetzung und Zerlegung.

Die entsprechende gespeicherte Prozedur für das Zerlegen ist dxxInsertXML; sie verwendet ebenfalls den Namen einer aktivierten XML-Objektgruppe.

Vorgehensweise

Um eine XML-Objektgruppe mit der gespeicherten Prozedur dxxRetrieveXML zusammenzusetzen, 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 */
```

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 die Tabelle eine Spalte vom Typ XMLVARCHAR enthält.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
             char    collection;   /* DAD-Puffer */
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;            /* tatsächliche Anzahl Zeilen */
long  returnCode;         /* Rückkehrfehlercode */
char  returnMsg[1024];    /* Fehlernachrichtentext */
             short   collection_ind;
short rtab_ind;
short ovtype_ind;
short ov_ind;
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 db2xml.dxxRetrieveXML(: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);
```

Zugehörige Konzepte:

- „XML-Objektgruppen als Speicher- und Zugriffsmethode" auf Seite 97
- „Schemata für die Zuordnung von XML-Objektgruppen" auf Seite 110
- „Standortpfade" auf Seite 118
- „DAD-Dateien für XML-Objektgruppen" auf Seite 178
- „Übersicht: Gespeicherte Zusammensetzungsprozeduren von XML Extender" auf Seite 214

Zugehörige Tasks:

- „XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen" auf Seite 68
- „Formatvorlagen für eine XML-Objektgruppe" auf Seite 117
- „XML-Objektgruppe mit Hilfe der RDB_node-Zuordnung zerlegen" auf Seite 71
- „Daten in XML-Objektgruppen aktualisieren und löschen" auf Seite 106
- „XML-Objektgruppen durchsuchen" auf Seite 108

XML-Dokumente in DB2 UDB-Daten zerlegen

Das Zerlegen eines XML-Dokuments bedeutet, dass die Daten in einem XML-Dokument syntaktisch analysiert und in relationalen Tabellen gespeichert werden. 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 UDB-Tabellenstruktur angibt. Die gespeicherten Prozeduren verwenden die DAD-Datei zum Zerlegen des XML-Dokuments.

XML-Objektgruppe zum Zerlegen aktivieren

In den meisten Fällen müssen Sie eine XML-Objektgruppe aktivieren, bevor Sie die gespeicherte Prozedur verwenden können. In folgenden Fällen müssen Sie die Objektgruppen aktivieren:

- Beim Zerlegen von XML-Dokumenten in neue Tabellen muss eine XML-Objektgruppe aktiviert sein, da alle Tabellen in der XML-Objektgruppe von XML Extender beim Aktivieren der Objektgruppe erstellt werden.
- Wenn die Beibehaltung der Reihenfolge von Elementen und Attributen mit mehrfachem Vorkommen wichtig ist. XML Extender behält die Reihenfolge von Elementen oder Attributen mit mehrfachem Vorkommen nur für Tabellen bei, die beim Aktivieren einer Objektgruppe erstellt werden. Wenn XML-Dokumente in vorhandene relationale Tabellen zerlegt werden, kann die Beibehaltung der Reihenfolge nicht garantiert werden.

Im Abschnitt zum Verwaltungsbefehl 'dxxadm' finden Sie Informationen zur Option 'enable_collection'.

Wenn Sie die DAD-Datei übergeben wollen, wenn die Tabellen bereits in der Datenbank vorhanden sind, brauchen Sie keine XML-Objektgruppe zu aktivieren.

Bevor Sie ein XML-Dokument in DB2 UDB-Daten zerlegen, führen Sie die folgenden Schritte aus:

1. Ordnen Sie die Struktur des XML-Dokuments den relationalen Tabellen zu, die den Inhalt der Element- und Attributwerte enthalten.
2. Bereiten Sie die DAD-Datei mit der RDB_node-Zuordnung vor.
3. Optional: Aktivieren Sie die XML-Objektgruppe.

Vorgehensweise:

Verwenden Sie eine der beiden gespeicherten Prozeduren (dxxShredXML() bzw. dxxInsertXML), die DB2 UDB ML Extender zum Zerlegen von XML-Dokumenten zur Verfügung stellt.

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 eingespart werden soll. Die gespeicherte Prozedur dxxShredXML() erfordert keine aktivierte Objektgruppe und verwendet stattdessen 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. Sie fügt entsprechend der Angabe <Xcollection> in der DAD-Datei Daten aus einem XML-Dokument in eine XML-Objektgruppe ein. Die gespeicherte Prozedur dxxShredXML() zerlegt anschließend das XML-Dokument und fügt unmarkierte XML-Daten in die in der DAD-Datei angegebenen Tabellen 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 entsprechende gespeicherte Prozedur für das Zusammensetzen ist dxxGenXML(); sie verwendet ebenfalls die DAD-Datei als Eingabeparameter. Für diese Prozedur muss die XML-Objektgruppe nicht aktiviert sein.

So zerlegen Sie eine XML-Objektgruppe mit 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 */
```

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(100K) dad;       /* DAD*/
      SQL TYPE is CLOB_FILE dadFile;         /* DAD-Datei */
      SQL TYPE is CLOB(1M)  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,
"dxx_install_verz/samples/db2xml/dad/
       getstart_xcollection.dad");
        dadFile.name_length=strlen("dxx_install_verz
       /samples/db2xml/dad/getstart_xcollection.dad");
       dadFile.file_option=SQL_FILE_READ;
       strcpy(xmlFile.name,"dxx_install_verz
       /samples/db2xml/xml/getstart_xcollection.xml");
       xmlFile.name_length=strlen
      ("dxx_install_verz/samples/db2xml/xml
      /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. 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 Daten aus einem 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 nicht markierte 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.

Vorgehensweise

Gehen Sie wie folgt vor, um eine XML-Objektgruppe zu zerlegen: dxxInsertXML():

Integrieren Sie den Aufruf einer gespeicherten Prozedur in Ihrer Anwendung mit der folgenden Deklaration einer gespeicherten Prozedur:

```
dxxInsertXML(char(
             ) collectionName,   /* Eingabe */
             CLOB(1M)       xmlobj,          /* Eingabe */
             long           returnCode,      /* Ausgabe */
             varchar(1024)  returnMsg)       /* Ausgabe */
```

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 der XML-Objektgruppe */
   SQL TYPE is CLOB_FILE xmlFile;          /* Eingabe-XML-Datei */
   SQL TYPE is CLOB(1M)   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,"dxx_install_verz/samples/db2xml
   /xml/getstart_xcollection.xml");
   xmlobj.name_length=strlen("dxx_install_verz/samples/db2xml
   /xml/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);
```

Einschränkungen für die Tabellengröße beim Zerlegen

Beim Zerlegen wird die RDB_node-Zuordnung verwendet, um anzugeben, wie ein XML-Dokument in DB2 UDB-Tabellen zerlegt wird, indem die Element- und Attributwerte extrahiert und in Tabellenzeilen gespeichert werden. Die Werte aus den einzelnen XML-Dokumenten werden in einer oder mehreren DB2 UDB-Tabellen gespeichert. Jede Tabelle kann maximal 10240 Zeilen aus jedem Dokument enthalten.

Wenn ein XML-Dokument beispielsweise in fünf Tabellen zerlegt wird, kann jede dieser fünf Tabellen bis zu 10240 Zeilen für dieses bestimmte Dokument enthalten. Wenn die Tabelle Zeilen für mehrere Dokumente enthält, kann sie bis zu 10240 Zeilen für jedes Dokument enthalten.

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 der Verwendung von mehrfach vorkommenden Elementen, dass maximal 1024 Zeilen aus einem einzelnen Dokument in eine Tabelle zerlegt werden können.

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zerlegungsprozeduren von XML Extender" auf Seite 226

Zugehörige Tasks:

- „XML-Objektgruppe mit Hilfe der RDB_node-Zuordnung zerlegen" auf Seite 71
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Gespeicherte Prozedur dxxInsertXML()" auf Seite 228
- „Gespeicherte Prozedur dxxShredXML()" auf Seite 226

Daten in XML-Objektgruppen aktualisieren und löschen

Sie können XML-Objektgruppen aktualisieren, löschen, durchsuchen und abrufen. Allerdings ist der Zweck der Verwendung einer XML-Objektgruppe das Speichern oder Abrufen nicht markierter Daten in Datenbanktabellen. Die Daten in den vorhandenen Datenbanktabellen haben nichts mit den ankommenden XML-Dokumenten zu tun; die Aktionen zum Aktualisieren, Löschen und Durchsuchen bestehen aus normalen SQL-Zugriffen auf diese Tabellen.

XML Extender bietet die Möglichkeit, von der XML-Objektgruppensicht aus Operationen mit den Daten vorzunehmen. Mit SQL UPDATE- und DELETE-Anweisungen können Sie die für das Zusammensetzen von XML-Dokumenten verwendeten Daten ändern und somit die XML-Objektgruppe aktualisieren. Das Ausführen von SQL-Operationen für die Objektgruppentabellen wirkt sich auf die generierten Dokumente aus.

- 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

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. Bei mehrfachem Vorkommen der Elemente oder Attribute kann durch Aktualisierungen 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 noch vorhanden, wenn der Elementtext bzw. der Attributwert existiert; somit handelt es sich aus dieser Sicht um ein Aktualisieren. SQL-Operationen für Objektgruppentabellen wirken sich auf Dokumente aus, die aus den Tabellen generiert werden.

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:
  - Schließen Sie für die SQL-Zuordnung die Verknüpfungsbedingung in das Element <SQL_stmt> ein.
  - Schließen Sie für die RDB_node-Zuordnung die Verknüpfungsbedingung in das Anfangselement <condition> des Stammelementknotens ein.

Elemente und Attributknoten aktualisieren

In einer XML-Objektgruppe sind 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. Wenn Sie die Zeile nicht löschen, die die Werte für den Anfangs-element_node enthält, wird das Löschen von Elementwerten als eine Aktualisierung des XML-Dokuments betrachtet.

In der folgenden Anweisung DELETE 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. Das Ausführen von SQL-Operationen für die Objektgruppentabellen wirkt sich auf die generierten Dokumente aus.

Vorgehensweise

Zum Löschen des Dokuments löschen Sie eine Zeile in der Tabelle, die den in der 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. Das Löschen des Dokuments mit dieser Methode funktioniert nur, wenn die Integritätsbedingungen von Primärschlüssel und Fremdschlüssel vollständig in SQL angegeben sind und wenn die Beziehung der in der DAD gezeigten Tabellen exakt mit diesen Integritätsbedingungen übereinstimmt.

Beispiel:

Die folgende Anweisung DELETE gibt den Wert der Primärschlüsselspalte an.

```
DELETE from order_tab
   WHERE order_key=1
```

ORDER_KEY ist der Primärschlüssel in der Tabelle ORDER_TAB, die, wie in der DAD angegeben, die Tabelle der obersten Ebene ist. 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-Objektgruppen durchsuchen

In diesem Abschnitt wird das Durchsuchen einer XML-Objektgruppe in Bezug auf das Generieren von XML-Dokumenten mit Hilfe von Suchkriterien und das Suchen nach zerlegten XML-Daten beschrieben.

XML-Dokumente mit Suchkriterien zusammensetzen

Diese Aufgabe entspricht dem Zusammensetzen mit einer Bedingung.

Vorgehensweise

Sie können die Suchkriterien wie folgt angeben:

- Angeben der Bedingung im text_node und attribute_node der DAD-Datei.
- Angeben des Parameters *override* 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)  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. Wenden Sie die Ergebnisse der strukturellen Textsuche an, um das angegebene XML-Dokument abzurufen oder zu generieren.

Schemata für die Zuordnung von XML-Objektgruppen

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

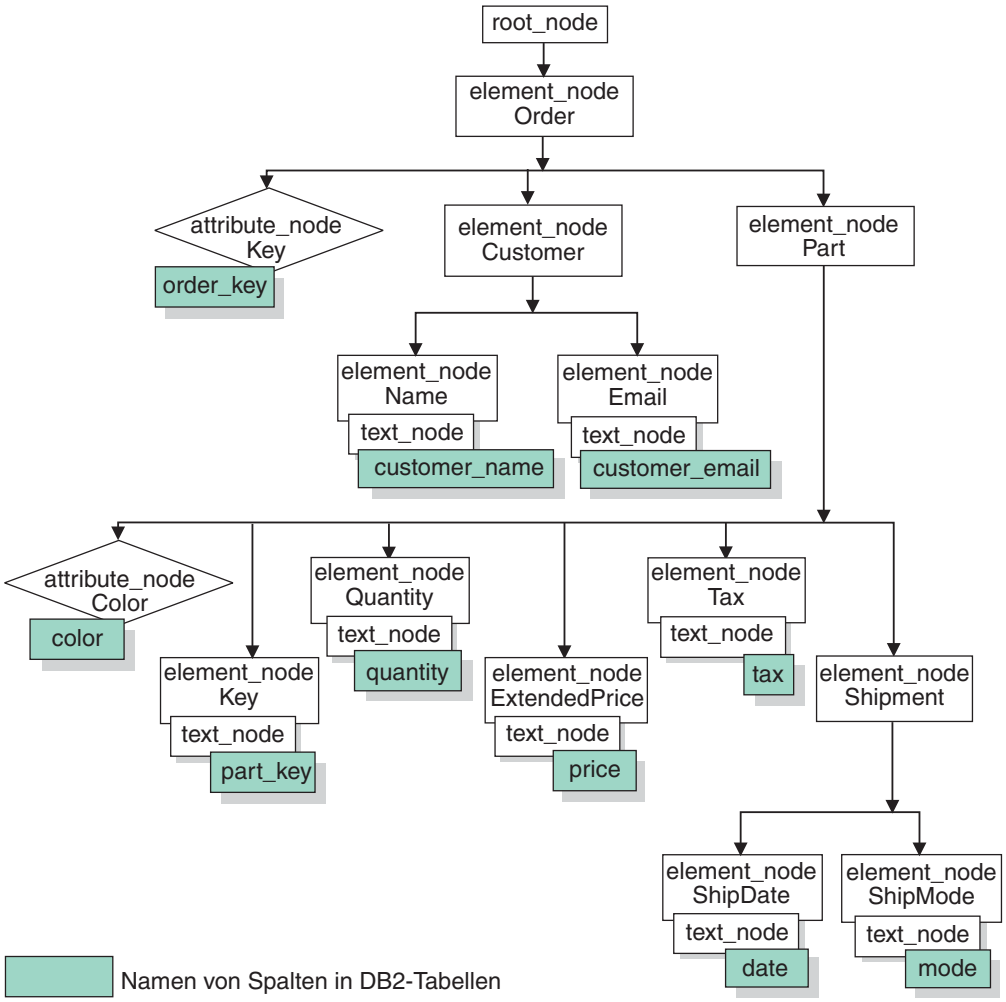


Abbildung 11. XML-Dokumentstruktur auf relationale Tabellenspalten abgebildet

XML Extender verwendet ein Zuordnungsschema beim Zusammensetzen oder Zerlegen von XML-Dokumenten in verschiedenen relationalen Tabellen. 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:

Verwenden Sie <Xcollection> zum Angeben des Zuordnungsschemas in der DAD-Datei. XML Extender bietet zwei Arten von Zuordnungsschemata: *SQL-Zuordnung* und *Zuordnung über die relationale Datenbank (RDB_node-Zuordnung)*.

SQL-Zuordnung

Diese Methode ermöglicht ein direktes Zuordnen von relationalen Daten zu XML-Dokumenten über eine einzelne SQL-Anweisung. Die SQL-Zuordnung wird nur für das Zusammensetzen verwendet. Der Inhalt des Elements <SQL_stmt> muss eine gültige SQL-Anweisung sein. Das Element <SQL_stmt> gibt Spalten in der SELECT-Klausel an, die später in der DAD den XML-Elementen oder -Attributen zugeordnet werden. Wenn 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 den Inhalt von *text_node* den Spalten zuzuordnen, die das gleiche *name_attribute* haben.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 den Wert eines XML-Attributs, so dass XML Extender feststellen kann, wo die XML-Daten gespeichert bzw. von wo sie abgerufen werden sollen.

Diese Methode verwendet den von XML Extender gelieferten *RDB_node*, der eine oder mehrere Knotendefinitionen für Tabellen, für optionale Spalten und für optionale Bedingungen enthält. Die Elemente <table> und <column> in der DAD definieren, 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 müssen Sie eine DAD-Datei mit einem Element <Xcollection> erstellen. Abb. 12 auf Seite 112 zeigt ein Fragment einer Muster-DAD-Datei mit der SQL-Zuordnung für eine XML-Objektgruppe, die eine Gruppe von XML-Dokumenten aus Daten in drei relationalen Tabellen zusammensetzt.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "dxx_install_verz/samples/db2xml/dtd/dad.dtd">
<DAD>
  <dtdid>dxx_install_verz/samples/dad/getstart.dtd</dtdid>
  <validation>YES</validation>
  <Xcollection>
    <SQL_stmt>
       SELECT o.order_key, customer, p.part_key, quantity, price, tax, date,
              ship_id, mode, comment
         FROM order_tab o, part_tab p,
               table(select substr(char(timestamp
                (generate_unique())),16)
                as ship_id, date, mode, from ship_tab)
         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
    "dxx_install_verz/samples/db2xml/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

XML Extender bietet verschiedene gespeicherte Prozeduren zur Verwaltung von Daten in einer XML-Objektgruppe. Diese gespeicherten Prozeduren unterstützen beide Arten der Zuordnung.

Zugehörige Konzepte:

- „DAD-Dateien für XML-Objektgruppen" auf Seite 178
- „Voraussetzungen für die Verwendung der SQL-Zuordnung" auf Seite 113
- „Voraussetzungen für die Verwendung der RDB_Node-Zuordnung" auf Seite 114

Zugehörige Tasks:

- „XML-Dokumente mit Hilfe der SQL-Zuordnung zusammensetzen" auf Seite 65
- „XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen" auf Seite 68
- „XML-Objektgruppe mit Hilfe der RDB_node-Zuordnung zerlegen" auf Seite 71

Voraussetzungen für die Verwendung der SQL-Zuordnung

Voraussetzungen für die Verwendung der SQL-Zuordnung

In diesem Zuordnungsschema müssen Sie das Element <SQL_stmt> innerhalb des DAD-Elements <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 derselben Tabelle und ordnen Sie die Tabellen entsprechend der Baumstrukturebene, da sie der hierarchischen Struktur Ihres XML-Dokuments zugeordnet werden. Die erste Spalte in jeder Spaltengruppierung ist eine Objekt-ID. 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 die Spalten `order_key` und `customer` aus der Tabelle ORDER_TAB die höchste relationale Ebene auf, da sie im hierarchischen Baum des XML-Dokuments höher angesiedelt sind. Die Spalten `ship_id`, `date` und `mode` aus der Tabelle SHIP_TAB befinden sich auf der niedrigsten relationalen Ebene.
  - 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 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. `ship_id` wird als Primärschlüssel generiert, da die Tabelle SHIP_TAB keinen Primärschlüssel hat. `ship_id` 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 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, part key from ship_tab) as s
    ```

    In diesem Beispiel wird ein einzelner Kandidatenschlüssel für die Spalte mit der Funktion generate_unique() generiert und die Spalte erhält den Aliasnamen `ship_id`.

- Verwenden Sie einen Aliasnamen, wenn erforderlich ist, dass eine Spalte eindeutig ist. Sie können beispielsweise `o` für Spalten in der Tabelle ORDER_TAB, `p` für Spalten in der Tabelle PART_TAB und `s` für Spalten in der Tabelle SHIP_TAB verwenden.

- **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 nach den Spaltennamen keine Angabe, wie etwa ASC oder DESC, folgt.
  - Stellen Sie sicher, dass die Spaltennamen den Spaltennamen in der SELECT-Klausel entsprechen.
  - Listen Sie alle Objekt-IDs in der gleichen relativen Reihenfolge auf, in der sie in der SELECT-Klausel erscheinen.
  - Eine Kennung kann mit Hilfe eines Tabellenausdrucks und der Funktion generate_unique() oder einer benutzerdefinierten Funktion generiert werden.
  - Behalten Sie die Top-Down-Reihenfolge der Entitäten-Hierarchie bei. Die erste 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 der ORDER BY-Klausel nicht durch ein Schema oder einen Tabellennamen.

Das Element <SQL_stmt> ist sehr leistungsstark, da Sie in Ihrer WHERE-Klausel ein beliebiges Prädikat angeben können, sofern der Ausdruck im Prädikat die Spalten in den Tabellen verwendet.

Zugehörige Referenzen:

- Anhang A, „Beispiele", auf Seite 313

Voraussetzungen für die Verwendung der RDB_Node-Zuordnung

Verwenden Sie zusammen mit der RDB_Node-Zuordnung nicht das Element <SQL_stmt> im Element <Xcollection> der DAD-Datei. Verwenden Sie stattdessen das Element RDB_node als untergeordnetes Element des Anfangs-element_node und jedes attribute_node und text_node.

- **RDB_node für den Anfangs-element_node**

  Der Anfangselementknoten 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 der XML-Objektgruppe zugeordnet sind. Die folgende Zuordnung gibt beispielsweise drei Tabellen im <RDB_node> des Elementknotens <Order> (des Anfangselementknotens) 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.

– Bedingungselemente können unbegrenzt oft auf einen Spaltennamen verweisen.
– Wenn Sie eine Objektgruppe aktivieren, müssen Sie für jede Tabelle einen Primärschlüssel angeben. Der Primärschlüssel kann aus einer einzelnen Spalte oder mehreren Spalten, dem zusammengesetzten Schlüssel, bestehen. Geben Sie den Primärschlüssel an, indem Sie einen *Attributschlüssel* zum Tabellenelement des RDB_node hinzufügen. Wenn Sie einen zusammengesetzten Schlüssel liefern, wird das *Schlüsselattribut* über die Namen der Schlüsselspalten, durch Leerzeichen getrennt, angegeben. Beispiel:
  `<table name="part_tab" key="part_key price"/>`
  Die für das Zerlegen angegebenen Informationen werden ignoriert, wenn beim Zusammensetzen dieselbe DAD verwendet wird.
– 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. Wenn Sie XML-Dokumente in eine XML-Objektgruppe zerlegen, kann die Reihenfolge mehrfach vorkommender Elemente und Attributwerte verloren gehen, es sei denn, Sie geben diese Reihenfolge in der DAD-Datei ein. Um diese Reihenfolge beizubehalten, verwenden Sie das Zuordnungsschema von RDB_node, und geben Sie das Attribut orderBy für die Tabelle mit dem Stammelement in seinem RDB_node an.

Scheiben Sie den Tabellennamen und den Spaltennamen im Befehl <table> explizit aus.

- **RDB_node für jeden attribute_node und text_node**

  XML Extender muss wissen, von wo in der Datenbank die Daten abgerufen werden sollen. XML Extender muss außerdem wissen, wo in der Datenbank der Inhalt aus einem XML-Dokument abgelegt werden soll. Sie müssen einen RDB_node für jeden Attributknoten und Textknoten angeben. Sie müssen außerdem Namen für die Tabelle und die Spalte angeben; der Bedingungswert ist optional.

  1. Geben Sie den Namen der Tabelle an, die die Spaltendaten enthält. Der Tabellenname muss im RDB_node des Anfangs-element_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>
```

2. Geben Sie den Namen der Spalte ein, die die Daten für den Elementtext enthält. Im vorigen Beispiel wurde die Spalte als PRICE angegeben.
3. Geben Sie eine Abfragebedingung an, wenn Sie die XML-Dokumente über die Bedingung erstellen wollen. Nur die Daten, die diese Bedingung erfüllen, werden in die generierten XML-Dokumente einbezogen. Die Bedingung muss eine gültige WHERE-Klausel sein. Im obigen Beispiel ist die Bedingung als `price > 2500.00` angegeben, so dass nur Zeilen, in denen der Preis über 2500 liegt, in die XML-Dokumente einbezogen werden.
4. Wenn Sie ein Dokument zerlegen oder die über die DAD-Datei angegebene XML-Objektgruppe aktivieren, müssen Sie den Spaltentyp für jeden Attributknoten und Textknoten angeben. Durch die Angabe des Spaltentyps für jeden Attributknoten und Textknoten stellen Sie den richtigen Datentyp für jede Spalte sicher, 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"/>
   ```

   Der für das Zerlegen eines Dokuments angegebene Spaltentyp wird beim Zusammensetzen ignoriert.

- Behalten Sie die Top-Down-Reihenfolge der Entitäten-Hierarchie bei. Stellen Sie sicher, dass die Elementknoten korrekt verschachtelt sind, damit XML Extender beim Zusammensetzen oder Zerlegen von Dokumenten die Beziehung zwischen den Elementen richtig versteht. Beispielsweise verschachtelt die folgende DAD-Datei 'Shipment' nicht korrekt in 'Part':

  ```
  <element_node name="Part">
      ...
   <element_node name="ExtendedPrice">
            ...
        </element_node>
            ...
  </element_node> <!-- Ende des Elements 'Part' -->

    <element_node name="Shipment" multi_occurrence="YES">
        <element_node name="ShipDate">
            ...
          </element_node>
        <element_node name="ShipMode">
            ...
          </element_node>

     </element_node> <!-- Ende des Elements 'Shipment'-->
  ```

  Diese DAD-Datei erzeugt XML-Dokumente, in denen die Elemente 'Part' und 'Shipment' Geschwisterelemente sind.

  ```
  <Part color="black ">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-2</Tax>
  </Part>

    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
  ```

Der folgende Code zeigt das Element 'Shipment' verschachtelt in das Element 'Part' in der DAD-Datei.

```
<element_node name="Part">
    ...
    <element_node name="ExtendedPrice">
          ...
            </element_node>
    ...
   <element_node name="Shipment" multi_occurrence="YES">
         <element_node name="ShipDate">
             ...
         </element_node>
         <element_node name="ShipMode">
             ...
           </element_node>

    </element_node> <!-- Ende des Elements 'Shipment'-->
</element_node> <!-- Ende des Elements 'Part' -->
```

Durch das Verschachteln des Elements 'Shipment' in das Element 'Part' wird eine XML-Datei erzeugt, bei der 'Shipment' ein untergeordnetes Element (Kindelement) des Elements 'Part' ist:

```
<Part color="black ">
  <key>68</key>
  <Quantity>36</Quantity>
  <ExtendedPrice>34850.16</ExtendedPrice>
  <Tax>6.000000e-2</Tax>
  <Shipment>
    <ShipDate>1998-08-19</ShipDate>
    <ShipMode>BOAT  </ShipMode>
  </Shipment>
</Part>
```

Es gibt keine Einschränkungen in Bezug auf die Reihenfolge bei Prädikaten der Stammknotenbedingung. 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.

Für eine Unterverzeichnisstruktur der DAD-Datei mit element_nodes und attribute_nodes, die derselben Tabelle zugeordnet werden, gilt Folgendes:

- Attributknoten müssen nicht das erste untergeordnete Element des niedrigsten allgemeinen Vorfahren der Elementknoten sein, die denselben Tabellen zugeordnet werden.
- Attributknoten können sich an beliebigen Stellen in einer Unterverzeichnisstruktur befinden, solange sie nicht in eine Verknüpfungsbedingung involviert sind.

Einschränkungen: Die Anzahl der in einer RDB_node-Zuordnungs-DAD erlaubten Tabellen beträgt 30. Die Anzahl der zulässigen Spalten pro Tabelle beträgt 500. Es gibt keine Einschränkung, wie oft jede Tabelle oder Spalte in den Verknüpfungsprädikaten der Bedingungsanweisung angegeben werden kann.

Formatvorlagen für eine XML-Objektgruppe

Beim Zusammensetzen von Dokumenten unterstützt XML Extender auch Verarbeitungsanweisungen für Formatvorlagen über das Element <stylesheet>. Die Verarbeitungsanweisungen müssen sich innerhalb des Stammelements <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>
```

Standortpfade

Ein *Standortpfad* definiert den Standort eines XML-Elements oder -Attributs innerhalb der Struktur eines XML-Dokuments. XML Extender verwendet den Standortpfad für die folgenden Zwecke:

- Zum Lokalisieren der zu extrahierenden Elemente und Attribute bei der Verwendung von Extraktions-UDFs wie z. B. dxxRetrieveXML.
- Zum Angeben der Zuordnung zwischen einem XML-Element bzw. -attribut und einer DB2-Spalte beim Definieren des Indexierungsschemas in der DAD für XML-Spalten.
- Für die strukturelle Textsuche unter Verwendung von Net Search Extender.
- Zum Überschreiben der DAD-Dateiwerte für eine XML-Objektgruppe in einer gespeicherten Prozedur.

Abb. 13 zeigt ein Beispiel eines Standortpfads und seiner Beziehung zur Struktur des XML-Dokuments.

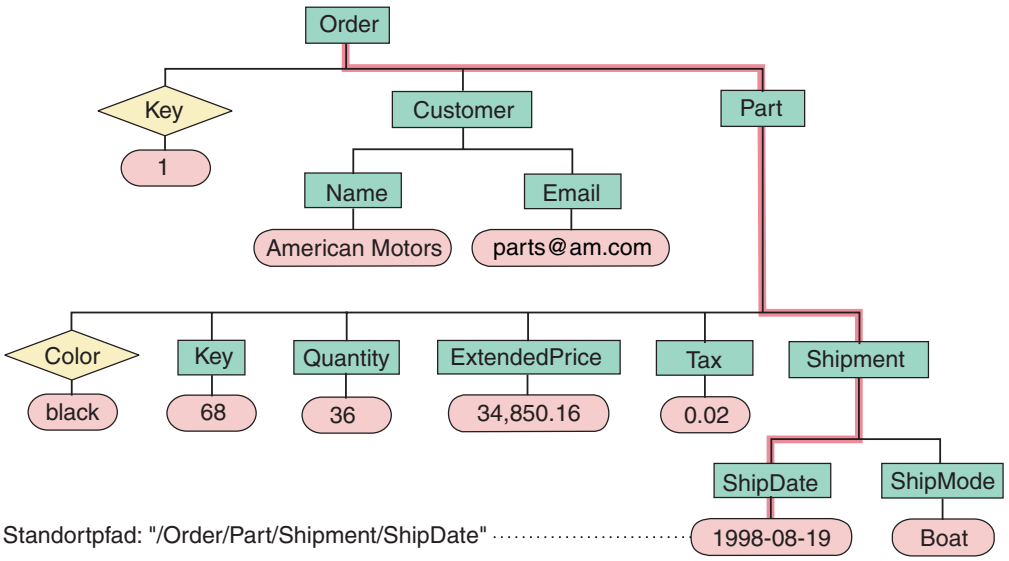


Abbildung 13. Speichern von Dokumenten als strukturierte XML-Dokumente in einer DB2 UDB-Tabelle

Zugehörige Referenzen:

- „Syntax des Standortpfads" auf Seite 119

Syntax des Standortpfads

XML Extender verwendet den Standortpfad zum Navigieren in der XML-Dokumentstruktur. Die folgende Liste beschreibt die Syntax für den Standortpfad, die von XML Extender unterstützt wird. Ein einfacher Schrägstrich (/) als Pfad gibt an, dass der Kontext das gesamte Dokument ist.

1. / Steht für das XML-Stammelement. Dies ist das Element, das alle anderen Elemente im Dokument enthält.
2. */tag1*
   Steht für das Element *tag1* unter dem Stammelement.
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 das Element mit dem Namen *tag2*, dessen Attribut *attr1* den Wert 5 hat. Das Element *tag2* ist ein untergeordnetes Element des Elements *tag1* unter dem Stammelement.
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.

Einfacher Standortpfad

Einfacher Standortpfad ist eine Art von Standortpfad, der in der DAD-Datei für XML-Spalten verwendet wird. Ein einfacher Standortpfad ist eine Folge von Elementartnamen, die durch einen einzelnen Schrägstrich (/) verbunden sind. Die Werte der einzelnen Attribute sind hinter der Elementart in eckigen Klammern angegeben. Tabelle 13 fasst die Syntax für den einfachen Standortpfad zusammen.

Tabelle 13. 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.

Tabelle 13. Syntax zum einfachen Standortpfad (Forts.)

Element	Standortpfad	Beschreibung
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.

Syntax des Standortpfads

Die Syntax des Standortpfads ist vom Kontext abhängig, in dem Sie auf den Standort eines Elements oder Attributs zugreifen. Da XML Extender eine Eins-zu-Eins-Zuordnung zwischen einem Element bzw. Attribut und einer DB2-Spalte verwendet, schränkt er die Syntaxregeln für die DAD-Datei und die Funktionen ein. Tabelle 14 beschreibt, in welchen Kontexten die Syntaxoptionen verwendet werden.

Tabelle 14. Einschränkungen von XML Extender bei der Verwendung des Standortpfads

Verwendung des Standortpfads	Unterstützter Standortpfad
Wert des Pfadattributs in der XML-Spalten-DAD-Zuordnung für Nebentabellen	*/tag1/tag2/..../tagn* und */tag1/tag2/@attr1* (einfacher Standortpfad, siehe Beschreibung in Tabelle 13 auf Seite 119)
Extraktions-UDFs	Alle Standortpfade[1]
Aktualisierungs-UDF	Alle Standortpfade[1]
Such-UDF Net Search Extender	*/tag1/tag2/..../tagn* — Ausnahme: Die Root-Markierung ist ohne Schrägstrich angegeben. Beispiel: `tag1/tag2/..../tagn`

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

Zugehörige Konzepte:

• „Standortpfade" auf Seite 118

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 UDB-Tabellen
- Zusammensetzen eines XML-Dokuments aus vorhandenen Daten in verschiedenen DB2 UDB-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 Verwaltungsassistenten von XML Extender, über den Befehl **dxxadm** mit der Option enable_collection oder über die gespeicherte Prozedur dxxEnableCollection() von XML Extender aktivieren.

Verwendung des Verwaltungsassistenten:

Gehen Sie wie folgt vor, um eine XML-Objektgruppe mit dem Assistenten zu aktivieren:

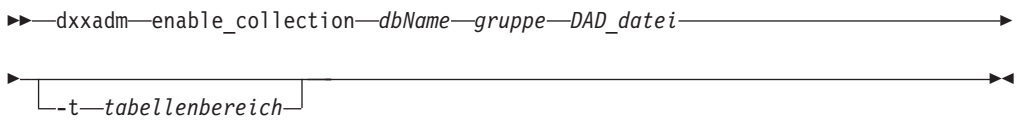
1. Konfigurieren Sie den Verwaltungsassistenten, und starten Sie ihn.
2. Klicken Sie im Klickstartleistenfenster **Mit XML-Objektgruppen arbeiten** an. Das Fenster **Eine Task auswählen** wird geöffnet.
3. Klicken Sie **Eine Objektgruppe aktivieren** und anschließend **Weiter** an. Das Fenster zum Aktivieren einer Objektgruppe wird geöffnet.
4. Wählen Sie den Namen der zu aktivierenden Objektgruppe im Feld **Name der Objektgruppe** aus.
5. Geben Sie den Namen der DAD-Datei im Feld **DAD-Dateiname** an.
6. Optional: Geben Sie den Namen eines zuvor erstellten Tabellenbereichs im Feld **Tabellenbereich** ein.

   Der Tabellenbereich enthält neue, zum Zerlegen generierte DB2 UDB-Tabellen.
7. Klicken Sie **Fertig stellen** an, um die Objektgruppe zu aktivieren und zum Klickstartleistenfenster zurückzukehren.
   - Wenn die Objektgruppe erfolgreich aktiviert wurde, erscheint die Nachricht `Objektgruppe erfolgreich aktiviert`.
   - Wurde die Objektgruppe nicht erfolgreich aktiviert, wird eine Fehlernachricht angezeigt. Wiederholen Sie diese Schritte, bis die Objektgruppe erfolgreich aktiviert wurde.

Objektgruppen mit dem Befehl dxxadm aktivieren:

Geben Sie zum Aktivieren einer XML-Objektgruppe in einer DB2 UDB-Befehlszeile 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 UDB-Tabellen enthält, die zum Zerlegen generiert wurden. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.

Beispiel: Das folgende Beispiel aktiviert über die Befehlszeile in der Datenbank SALES_DB eine Objektgruppe ″sales_ord″. Die DAD-Datei verwendet die SQL-Zuordnung.

```
dxxadm enable_collection SALES_DB sales_ord getstart_collection.dad
```

Nachdem Sie die XML-Objektgruppe aktiviert haben, können Sie XML-Dokumente mit den gespeicherten Prozeduren von XML Extender zusammensetzen oder zerlegen.

Zugehörige Konzepte:

- „XML-Objektgruppen als Speicher- und Zugriffsmethode” auf Seite 97

Zugehörige Tasks:

- „XML-Objektgruppen inaktivieren” auf Seite 122
- „Daten in XML-Objektgruppen verwalten” auf Seite 98

XML-Objektgruppen inaktivieren

Durch das Inaktivieren einer XML-Objektgruppe wird der Datensatz, der Tabellen und Spalten als Teil einer Objektgruppe kennzeichnet, aus der Tabelle XML_USAGE entfernt. Es werden keine Datentabellen freigegeben. Sie inaktivieren eine Objektgruppe, wenn Sie die DAD aktualisieren möchten und eine Objektgruppe wieder aktivieren müssen, oder wenn Sie eine Objektgruppe löschen möchten.

Sie können eine XML-Objektgruppe über den Verwaltungsassistenten von XML Extender, über den Befehl **dxxadm** mit der Option disable_collection oder über die gespeicherte Prozedur dxxDisableCollection() von XML Extender inaktivieren.

Vorgehensweise:

Gehen Sie wie folgt vor, um eine XML-Objektgruppe mit dem Verwaltungsassistenten zu inaktivieren:

1. Starten Sie den Verwaltungsassistenten.
2. Klicken Sie im Klickstartleistenfenster **Mit XML-Objektgruppen arbeiten** an, um die Tasks zu XML Extender-Objektgruppen anzuzeigen. Das Fenster **Eine Task auswählen** wird geöffnet.
3. Klicken Sie **Eine XML-Objektgruppe inaktivieren** und anschließend **Weiter** an, um eine XML-Objektgruppe zu inaktivieren. Das Fenster zum Inaktivieren einer Objektgruppe wird geöffnet.
4. Geben Sie den Namen der zu inaktivierenden Objektgruppe im Feld **Name der Objektgruppe** ein.
5. Klicken Sie **Fertig stellen** an, um die Objektgruppe zu inaktivieren und zum Klickstartleistenfenster zurückzukehren.
   - Wenn die Objektgruppe erfolgreich inaktiviert wurde, wird die Nachricht `Objektgruppe erfolgreich inaktiviert` angezeigt.
   - Wurde die Objektgruppe nicht erfolgreich inaktiviert, wird eine Fehlernachricht angezeigt. Wiederholen Sie diese Schritte, bis die Objektgruppe erfolgreich inaktiviert wurde.

Geben Sie zum Inaktivieren einer XML-Objektgruppe von der Befehlszeile aus 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.

Beispiel:

```
dxxadm disable_collection SALES_DB sales_ord
```

Zugehörige Konzepte:

- „XML-Objektgruppen als Speicher- und Zugriffsmethode" auf Seite 97
- „Übersicht: Gespeicherte Verwaltungsprozeduren von XML Extender" auf Seite 208

Zugehörige Tasks:

- „Daten in XML-Objektgruppen verwalten" auf Seite 98

Kapitel 5. XML-Schemata

Das XML-Schema kann an Stelle einer DTD verwendet werden, um die Spezifikationen für den Inhalt von XML-Dokumenten zu definieren. Das XML-Schema verwendet XML-Format oder XML-Syntax, um die Elemente und Attributnamen eines XML-Dokuments zu definieren, und definiert die Art des Inhalts, den Elemente und Attribute enthalten dürfen.

Vorteile von XML-Schemata gegenüber DTDs

DTDs sind einfacher zu codieren und zu prüfen als ein XML-Schema. Aber es gibt bei der Verwendung eines XML-Schemas einige Vorteile, wie die folgende Liste zeigt:

- XML-Schemata sind gültige XML-Dokumente, die von Tools, wie etwa dem XSD Editor im WebSphere® Studio Application Developer, XML Spy oder XML Authority, verarbeitet werden können.
- XML-Schemata sind leistungsfähiger als DTDs. Alle Definitionen, die mit Hilfe von DTD vorgenommen werden, können Sie auch mit Hilfe von Schemata vornehmen, aber nicht umgekehrt.
- XML-Schemata unterstützen eine Reihe von Datentypen, die denen ähnlich sind, die in den meisten gebräuchlichen Programmiersprachen verwendet werden. Sie bieten außerdem die Möglichkeit der Erstellung zusätzlicher Typen. Sie können den Dokumentinhalt auf den passenden Typ beschränken. Beispielsweise können Sie die Eigenschaften von Feldern replizieren, die Sie in DB2® finden.
- XML-Schemata unterstützen reguläre Ausdrücke, um Integritätsbedingungen für Zeichendaten zu setzen. Dies ist bei der Verwendung einer DTD nicht möglich.
- XML-Schemata bieten eine bessere Unterstützung für XML-Namensbereiche, über die Sie Dokumente prüfen können, die mehrere Namensbereiche verwenden. Außerdem können Sie Konstrukte von Schemata erneut verwenden, die bereits in anderen Namensbereichen definiert sind.
- XML-Schemata bieten eine bessere Unterstützung für die Modularität und Wiederverwendbarkeit mit dem Einschließen und Importieren von Elementen.
- XML-Schemata unterstützen die Vererbung für Element-, Attribut- und Datentypdefinitionen.

Zugehörige Tasks:

- „Datentypen, Elemente und Attribute in Schemata” auf Seite 126

Zugehörige Referenzen:

- „Beispiele eines XML-Schemas” auf Seite 128

XML-Schemaelement complexType

Das XML-Schemaelement complexType wird zur Definition einer Elementart verwendet, die aus Unterelementen bestehen kann. Beispielsweise zeigen die folgenden Befehle die Projektion einer Adresse in einem XML-Dokument:

```
<billTo country="US">
    <name>Dan Jones</name>
    <street>My Street</street>
```

```
    <city>My Town</city>
    <state>CA</state>
    <zip>99999</zip>
</billTo>
```

Die Struktur dieses Elements kann wie folgt im XML-Schema definiert werden:

```
1 <xsd:element name="billTo" type="USAddress"/>
2 < xsd:complexType name="USAddress">
 3  <xsd:sequence>
 4   < xsd:element name="name" type="xsd:string"/>
 5   < xsd:element name="street" type="xsd:string"/>
 6   < xsd:element name="city" type="xsd:string"/>
 7   < xsd:element name="state" type="xsd:string"/>
 8   < xsd:element name="zip" type="xsd:decimal"/>
 9  </xsd:sequence>
 10  < xsd:attribute name="country"
                          type="xsd:NMTOKEN" use="fixed"
                          value="US"/>
 12</xsd:complexType>
```

Im obigen Beispiel wird davon ausgegangen, dass das Präfix `xsd` an den Namensbereich des XML-Schemas gebunden wurde. Die Zeilen 2 bis 12 definieren den 'complexType' 'USAddress' als eine Folge von fünf Elementen und einem Attribut. Die Reihenfolge der Elemente ist durch die Reihenfolge festgelegt, in der sie im Befehl 'sequence' erscheinen.

Die inneren Elemente sind vom Datentyp 'xsd:string' oder 'xsd:decimal'. Bei beiden handelt es sich um vordefinierte einfache Datentypen.

An Stelle des Befehls <sequence> können Sie alternativ den Befehl <all> bzw. <choice> verwenden. Mit dem Befehl 'all' müssen alle Unterelemente angezeigt werden, jedoch nicht in einer bestimmten Reihenfolge. Mit dem Befehl 'choice' muss genau eines der Unterelemente im XML-Dokument angezeigt werden.

Sie können außerdem einen benutzerdefinierten Datentyp verwenden, um andere Elemente zu definieren.

Datentypen, Elemente und Attribute in Schemata

Einfache Datentypen in XML-Schemata

XML-Schemata bieten eine Reihe von einfachen integrierten Datentypen. Sie können andere Datentypen davon ableiten, indem Sie Integritätsbedingungen anwenden.

In Beispiel 1 kann der Bereich des Basistyps 'xsd:positiveInteger' zwischen 0 und 100 liegen.

Beispiel 1

```
< xsd:element name="quantity">
   < xsd:simpleType>
    < xsd:restriction base="xsd:positiveInteger">
     < xsd:maxExclusive value="100"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
```

In Beispiel 2 ist der Basistyp 'xsd:string' durch einen regulären Ausdruck beschränkt.

Beispiel 2

```
<xsd:simpleType name="SKU">
   < xsd:restriction base="xsd:string">
    < xsd:pattern value="\d{3}-[A-Z]{2}"/>
   </xsd:restriction>
  </xsd:simpleType>
```

Beispiel 3 zeigt einen Aufzählungstyp auf der Basis des integrierten Typs 'string'.

Beispiel 3

```
<xsd:simpleType name="SchoolClass">
   < xsd:restriction base="xsd:string">
    < xsd:enumeration value="WI"/>
    < xsd:enumeration value="MI"/>
    < xsd:enumeration value="II"/>
    < xsd:enumeration value="DI"/>
    < xsd:enumeration value="AI"/>
   </xsd:restriction>
  </xsd:simpleType>
```

Elemente in XML-Schemata

Um ein Element in einem XML-Schema zu deklarieren, müssen Sie den Namen und den Typ als Attribut des Elements 'element' angeben. Beispiel:

```
<xsd:element name="street" type="xsd:string"/>
```

Darüber hinaus können Sie die Attribute 'minOccurs' und 'maxOccurs' verwenden, um die maximale oder minimale Anzahl der Vorkommen des Elements im XML-Dokument festzulegen. Der Standardwert für 'minOccurs' und 'maxOccurs' ist 1.

Attribute in XML-Schemata

Attributdeklarationen müssen am Ende einer Elementdefinition stehen. Beispiel:

```
<xsd:complexType name="PurchaseOrderType">
< xsd:sequence>
  < xsd:element name="billTo" type="USAddress"/>
< xsd:sequence>
  < xsd:attribute name="orderDate" type="xsd:date"/>
 </xsd:complexType>
```

Zugehörige Konzepte:

- „Vorteile von XML-Schemata gegenüber DTDs" auf Seite 125

Zugehörige Tasks:

- „Überprüfungsfunktionen" auf Seite 172

Zugehörige Referenzen:

- „Beispiele eines XML-Schemas" auf Seite 128
- „XML-Schemaelement complexType" auf Seite 125

Beispiele eines XML-Schemas

Eine gute Strategie zum Schreiben von XML-Schemata ist, zunächst die Datenstruktur Ihres XML-Dokuments mit Hilfe eines UML-Tools zu entwerfen. Nach dem Design der Struktur können Sie die Struktur Ihrem Schemadokument zuordnen. Das folgende Beispiel zeigt ein XML-Schema.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
 3
 4   <xs:element name="personnel">
 5    <xs:complexType>
 6     <xs:sequence>
 7       <xs:element ref="person" minOccurs='1' maxOccurs='unbounded'/>
 8    </xs:sequence>
 9   </xs:complexType>
10  </xs:element>
11
12   <xs:element name="person">
13    <xs:complexType>
14     <xs:sequence>
15       <xs:element ref="name"/>
16       <xs:element ref="email" minOccurs='0' maxOccurs='4'/>
17    </xs:sequence>
18     <xs:attribute name="id"  type="xs:ID" use='required'/>
19   </xs:complexType>
20  </xs:element>
21
22   <xs:element name="name">
23    <xs:complexType>
24     <xs:sequence>
25      <xs:element ref="family"/>
26      <xs:element ref="given"/>
27    </xs:sequence>
28   </xs:complexType>
29  </xs:element>
30
31   <xs:element name="family" type='xs:string'/>
32   <xs:element name="given" type='xs:string'/>
33   <xs:element name="email" type='xs:string'/>
34 </xs:schema>
```

Die ersten beiden Zeilen bedeuten, dass dieses XML-Schema mit XML 1.0 kompatibel und mit Unicode 8 decodiert ist. Sie geben die Verwendung des Standardnamensbereichs für XML-Schemata an, wodurch der Zugriff auf die grundlegenden Datentypen und Strukturen für XML-Schemata möglich ist.

Die Zeilen 4 bis 10 definieren 'personnel' als 'complexType', der aus einer Folge von 1 bis n Personen besteht. Der 'complexType' wird anschließend in den Zeilen 12 bis 20 definiert. Er besteht aus dem Elementnamen 'complexType' und dem Element 'email'. Das Element 'email' ist optional (minOcccurs = '0') und kann bis zu vier Mal (maxOccurs = '4') auftreten. Je höher die Zahl für das Vorkommen (die Häufigkeit) eines Elements ist, desto länger dauert die Prüfung des Schemas. Im Gegensatz dazu können Sie in einer DTD nur 0, 1 oder unbegrenzt für das Vorkommen eines Elements wählen.

Die Zeilen 22 bis 29 definieren den Typ 'name', den Sie für den Typ 'person' verwendet haben. Der Typ 'name' besteht aus einer Folge von einem Element 'family' und einem Element 'given'.

Die Zeilen 31 bis 33 definieren die einzelnen Elemente *'family'*, *'given'* und *'email'*, die den Typ 'string' enthalten, der deklariert wurde.

XML-Dokumentexemplar, das das Schema verwendet

Das folgende Beispiel ist ein XML-Dokument, das ein Exemplar des Schemas *personalnr.xsd* ist.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:noNamespaceSchemaLocation='personsnr.xsd'>
4
5   <person id="Big.Boss" >
6      <name><family>Boss</family> <given>Big</given></name>
7     <email>chief@foo.com</email>
8   </person>
9
10   <person id="one.worker">
11   <name><family>Worker</family><given>One</given></name>
12      <email>one@foo.com</email>
13   </person>
14
15    <person id="two.worker">
16   <name><family>Worker</family><given>Two</given></name>
17      <email>two@foo.com</email>
18   </person>
19 </personnel>
```

XML-Dokumentexemplar, das eine DTD verwendet

Dieses Beispiel zeigt, wie dieses XML-Schema als DTD realisiert werden könnte.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!ELEMENT email (#PCDATA)>
3   <!ELEMENT family (#PCDATA)>
4   <!ELEMENT given (#PCDATA)>
5   <!ELEMENT name (family, given)>
6   <!ELEMENT person (name, email*)>
7
8   <!ATTLIST person
9 id ID #REQUIRED>
10 <!ELEMENT personnel (person+)>
```

Mit einer DTD können Sie das maximale Vorkommen von 'email' auf lediglich 1 oder unbegrenzt setzen.

Unter Verwendung dieser DTD würde das XML-Dokumentexemplar dem im oberen Beispiel gezeigten entsprechen, mit Ausnahme von Zeile 2, die wie folgt geändert werden müsste:

```
<!DOCTYPE personnel SYSTEM "personsnr.dtd">
```

Zugehörige Konzepte:

- „Vorteile von XML-Schemata gegenüber DTDs" auf Seite 125

Zugehörige Tasks:

- „Datentypen, Elemente und Attribute in Schemata" auf Seite 126
- „Überprüfungsfunktionen" auf Seite 172

Zugehörige Referenzen:

- „XML-Schemaelement complexType" auf Seite 125

Teil 4. Referenz

Dieser Teil 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 6. Verwaltungsbefehl dxxadm

Übersicht: Befehl 'dxxadm'

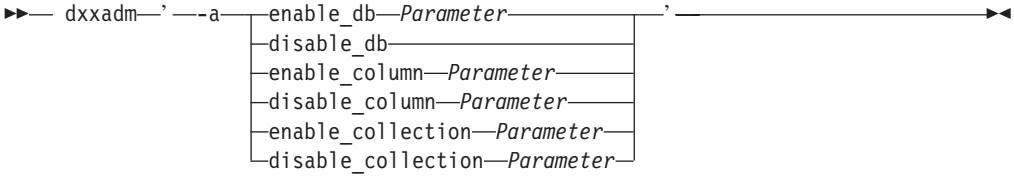
XML Extender bietet einen Verwaltungsbefehl, **dxxadm**, zur Ausführung der folgenden Verwaltungstasks:

- enable_column
- enable collection
- enable_db
- disable_column
- disable_collection
- disable_db

Zugehörige Konzepte:

- „Verwaltungstools für XML Extender" auf Seite 39
- „Übersicht: XML Extender-Verwaltung" auf Seite 41

Syntax des Verwaltungsbefehls dxxadm



Parameter:

Tabelle 15. Parameter für dxxadm

Parameter	Beschreibung
enable_db	Aktiviert die XML Extender-Funktionen für eine Datenbank.
disable_db	Inaktiviert die XML Extender-Funktionen für eine Datenbank.
enable_column	Aktiviert eine XML-Spalte, so dass XML-Dokumente in der Spalte gespeichert werden können.
disable_column	Inaktiviert die für XML aktivierte Spalte.
enable_collection	Aktiviert eine XML-Objektgruppe entsprechend der angegebenen DAD.
enable_collection	Inaktiviert eine für XML aktivierte Objektgruppe.

Optionen für den Verwaltungsbefehl

Die folgenden Befehlsoptionen des Befehls **dxxadm** stehen für Systemprogrammierer zur Verfügung:

- enable_column
- enable_collection
- enable_db
- disable_column
- disable_collection
- disable_db

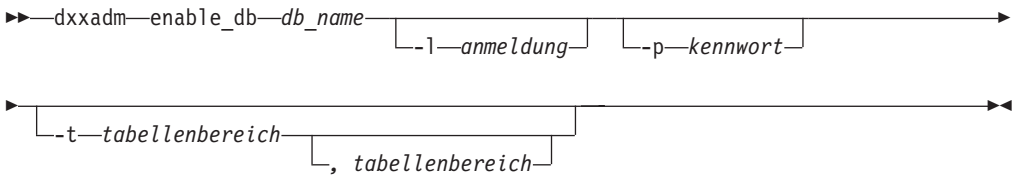
Option enable_db des Befehls dxxadm

Zweck:

Aktiviert die XML Extender-Funktionen für eine Datenbank. Wenn die Datenbank aktiviert ist, erstellt XML Extender die folgenden Objekte:

- Die benutzerdefinierten Typen (UDTs) von XML Extender
- Die benutzerdefinierten Funktionen (UDFs) von XML Extender
- Die DTD-Repositorytabelle von XML Extender, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden.
- Die XML Extender-Nutzungstabelle XML_USAGE, in der allgemeine Informationen für die einzelnen Spalten, die für XML aktiviert sind, und für die einzelnen Objektgruppen gespeichert werden

Syntax:



Parameter:

Tabelle 16. Parameter für enable_db

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der die XML-Daten gespeichert sind.
-l *anmeldung*	Optionale Benutzer-ID, die zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Optionales Kennwort, das zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.
-t *tabellenbereich*	Optionaler Name des bestehenden Tabellenbereichs zur Aufnahme der Tabellen db2xml.XML_USAGE und db2xml.DTD_REF. Auch ein zweiter Tabellenbereich kann angegeben werden.

Wenn Sie die partitionierte DB2 UDB Enterprise Server Edition verwenden und bei der Aktivierung der Datenbank einen Tabellenbereich angeben wollen, müssen Sie bei der Erstellung des Tabellenbereichs eine Knotengruppe angegeben haben. Beispiel:

```
db2 "create database partition group mygroup on node (0,1)"
db2 "create regular tablespace mytb in database partition group mygroup
   managed by system using ('mytb')"
```

Im obigen Beispiel würden Sie dann bei der Aktivierung der Datenbank den Tabellenbereich `mytb` angeben.

Wenn bei der Aktivierung der Datenbank keine Tabellenbereichsoption angegeben wird, prüft XML Extender, ob die Tabellenbereiche DXXDTDRF und DXXXMLUS vorhanden sind. Die Tabelle `db2xml.dtd_ref` wird im Tabellenbereich DXXDTDRF erstellt, sofern dieser Tabellenbereich vorhanden ist. Die Tabelle `db2xml.xml_usage` wird im Tabellenbereich DXXXMLUS erstellt. Wenn einer der Tabellenbereiche DXXDTDRF oder DXXXMLUS nicht vorhanden ist, wird die entsprechende Tabelle (`db2xml.dtd_ref` oder `db2xml.xml_usage`) in dem Tabellenbereich erstellt, der sich am besten eignet.

Wenn bei der Aktivierung der Datenbank nur ein DXXDTDRF-Tabellenbereich angegeben wird, werden beide Tabellen im angegebenen Tabellenbereich erstellt. Wenn bei der Aktivierung der Datenbank zwei Tabellenbereiche angegeben werden, wird die Tabelle `db2xml.dtd_ref` im ersten und die Tabelle `db2xml.xml_usage` im zweiten aufgeführten Tabellenbereich erstellt.

Beispiel::

Mit dem folgenden Beispiel wird die Datenbank SALES_DB aktiviert.

```
dxxadm enable_db SALES_DB
```

Zugehörige Referenzen:

- „Übersicht: Befehl 'dxxadm'" auf Seite 133

Option disable_db des Befehls dxxadm

Zweck:

Inaktiviert die XML Extender-Funktionen für eine Datenbank; diese Aktion wird „Inaktivieren einer Datenbank" genannt. Wenn die Datenbank inaktiviert ist, kann sie von XML Extender nicht mehr verwendet werden. Wenn XML Extender die Datenbank inaktiviert, werden die folgenden Objekte freigegeben:

- Die benutzerdefinierten Typen (UDTs) von XML Extender.
- Die benutzerdefinierten Funktionen (UDFs) von XML Extender.
- Die DTD-Repositorytabelle von XML Extender, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden.
- Die XML Extender-Nutzungstabelle, XML_USAGE, in der allgemeine Informationen für die einzelnen Spalten, die für XML aktiviert sind, und für die einzelnen Objektgruppen gespeichert werden.

Wichtig: Sie müssen alle XML-Spalten inaktivieren, bevor Sie versuchen, eine Datenbank zu inaktivieren. XML Extender kann eine Datenbank nicht inaktivieren, wenn diese Spalten oder Objektgruppen enthält, die für XML aktiviert wurden. Sie müssen außerdem alle Tabellen freigeben, die Spalten enthalten, die mit UDTs von XML Extender, z. B. XMLCLOB, definiert wurden.

Syntax:

Parameter:

Tabelle 17. Parameter für disable_db

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der die XML-Daten gespeichert sind.
-l *anmeldung*	Die Benutzer-ID, die zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das Kennwort, das zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiel::
Mit dem folgenden Beispiel wird die Datenbank SALES_DB inaktiviert.

```
dxxadm disable_db SALES_DB
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Verwaltungsprozeduren von XML Extender" auf Seite 208
- Kapitel 13, „Tabellen zur Verwaltungsunterstützung von XML Extender", auf Seite 287

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix

Option enable_column des Befehls dxxadm

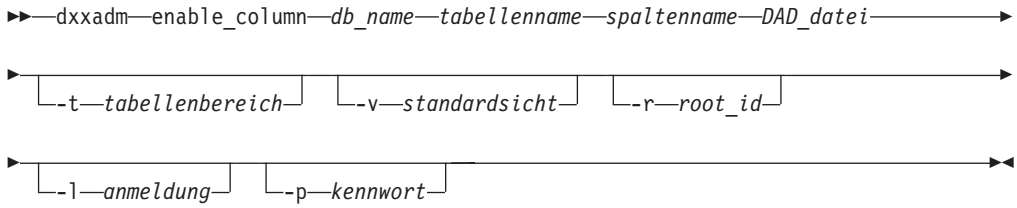
Zweck:

Stellt eine Verbindung zu einer Datenbank her und aktiviert eine XML-Spalte, so dass sie die UDTs von XML Extender enthalten kann. Beim Aktivieren einer Spalte führt XML Extender die folgenden Tasks aus:

- Feststellen, ob die XML-Tabelle einen Primärschlüssel hat. Falls nicht, ändert XML Extender die XML-Tabelle und fügt eine Spalte mit dem Namen DXXROOT_ID hinzu.
- Erstellen der in der DAD-Datei angegebenen Nebentabellen 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 von XML Extender benannt wurde.

- Optionales Erstellen einer Standardsicht für die XML-Tabelle und ihre Nebentabellen. Dabei wird optional ein von Ihnen angegebener Name verwendet.

Syntax:



Parameter:

Tabelle 18. Parameter für enable_column

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der die XML-Daten gespeichert sind.
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 Nebentabellen zuordnet.
-t *tabellenbereich*	Der Tabellenbereich, der die Nebentabellen enthält, die der XML-Spalte zugeordnet sind. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.
-v *standardsicht*	Der Name der Standardsicht, die die XML-Spalte und die Nebentabellen verknüpft.
-r *root_id*	Der Name des Primärschlüssels in der XML-Spaltentabelle, der als Root-ID für Nebentabellen verwendet werden soll. Die Root-ID ist wahlfrei.
-l *anmeldung*	Die Benutzer-ID, die zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das Kennwort, das zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Wenn Sie die partitionierte DB2 UDB Enterprise Server Edition verwenden und bei der Aktivierung einer Spalte einen Tabellenbereich angeben wollen, müssen Sie bei der Erstellung des Tabellenbereichs eine Knotengruppe angegeben haben. Zum Beispiel:

```
db2 "create database partition group mygroup on node (0,1)"
db2 "create regular tablespace mytb in database partition group mygroup
   managed by system using ('mytb')"
```

Im obigen Beispiel würden Sie dann bei der Aktivierung der Spalte den Tabellenbereich `mytb` angeben:

```
dxxadm enable_column mydatabase mytable mycolumn "dad/mydad.dad" -t mytb
```

Beispiel::

Mit dem folgenden Beispiel wird eine XML-Spalte aktiviert.

```
dxxadm enable_column SALES_DB SALES_TAB ORDER getstart.dad
   -v sales_order_view -r INVOICE_NUMBER
```

Zugehörige Beispiele:

- „dxx_xml -- s-getstart_enableCol_NT-cmd.htm"
- „dxx_xml -- s-getstart_enableCol-cmd.htm"

Option disable_column des Befehls dxxadm

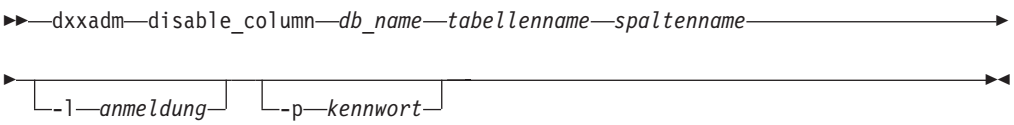
Zweck:

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 Nebentabellen werden freigegeben.

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 XML Extender sowohl die erstellten Nebentabellen als auch den XML-Spalteneintrag in der Tabelle XML_USAGE bei.

Syntax:



Parameter:

Tabelle 19. Parameter für disable_column

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der die Daten gespeichert sind.
tabellenname	Der Name der Tabelle, in der sich die XML-Spalte befindet.
spaltenname	Der Name der XML-Spalte.
-l *anmeldung*	Die Benutzer-ID, die zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das Kennwort, das zum Herstellen einer Verbindung zur Datenbank verwendet wird. 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
```

Zugehörige Referenzen:

- „Option enable_collection des Befehls dxxadm" auf Seite 139

Zugehörige Beispiele:

- „dxx_xml -- s-getstart_clean_NT-cmd.htm"
- „dxx_xml -- s-getstart_clean-cmd.htm"

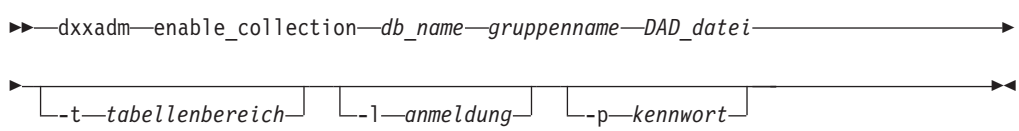
Option enable_collection des Befehls dxxadm

Zweck:

Stellt die Verbindung zu einer Datenbank her und aktiviert eine XML-Objektgruppe entsprechend der angegebenen DAD. Wenn Sie XML Extender in einer partitionierten Enterprise Server Edition-Umgebung ausführen, stellen Sie sicher, dass alle in Ihrer DAD-Datei angegebenen Tabellen mindestens eine Spalte enthalten, die als Partitionierungsschlüssel qualifiziert ist. Beim Aktivieren einer Objektgruppe führt 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.

Syntax:



Parameter:

Tabelle 20. Parameter für enable_collection

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der die Daten gespeichert sind.
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 Name des Tabellenbereichs, der der Objektgruppe zugeordnet ist. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.
-l *anmeldung*	Die Benutzer-ID, die zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das Kennwort, das zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Wenn Sie die partitionierte DB2 UDB Enterprise Server Edition verwenden und bei der Aktivierung einer Objektgruppe einen Tabellenbereich angeben wollen, müssen Sie bei der Erstellung des Tabellenbereichs eine Knotengruppe angegeben haben. Zum Beispiel:

```
db2 "create database partition group mygroup on node (0,1)"
db2 "create regular tablespace mytb in database partition group mygroup
   managed by system using ('mytb')"
```

Im obigen Beispiel würden Sie dann bei der Aktivierung der Objektgruppe den Tabellenbereich `mytb` angeben.

Beispiele:

Mit dem folgenden Beispiel wird eine XML-Objektgruppe aktiviert.

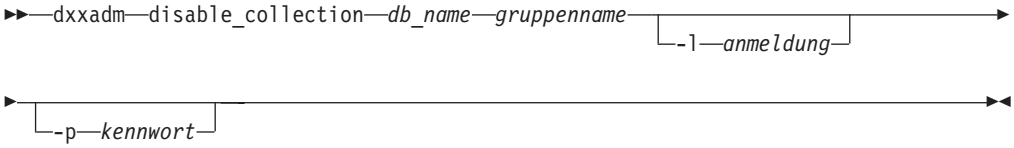
```
dxxadm enable_collection SALES_DB sales_ord
getstart_xcollection.dad -t orderspace
```

Option disable_collection des Befehls dxxadm

Zweck:

Stellt die Verbindung zu einer Datenbank her und inaktiviert eine für XML aktivierte Objektgruppe. Der Objektgruppenname kann in den gespeicherten Prozeduren für Zusammensetzung (dxxRetrieveXML) und für Zerlegung (dxxInsertXML) nicht mehr verwendet werden. Wenn eine XML-Objektgruppe inaktiviert ist, wird der entsprechende Gruppeneintrag aus der Tabelle XML_USAGE gelöscht. Durch das Inaktivieren der Objektgruppe werden die Objektgruppentabellen, die während der Verwendung der Option enable_collection erstellt wurden, nicht gelöscht.

Syntax:



Parameter:

Tabelle 21. Parameter für disable_collection

Parameter	Beschreibung
db_name	Der Name der Datenbank, in der die Daten gespeichert sind.
gruppenname	Der Name der XML-Objektgruppe.
-l *anmeldung*	Die Benutzer-ID, die zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist sie nicht angegeben, wird die aktuelle Benutzer-ID verwendet.
-p *kennwort*	Das Kennwort, das zum Herstellen einer Verbindung zur Datenbank verwendet wird. Ist es nicht angegeben, wird das aktuelle Kennwort verwendet.

Beispiel::
Mit dem folgenden Beispiel wird eine XML-Objektgruppe inaktiviert.

```
dxxadm disable_collection SALES_DB sales_ord
```

Kapitel 7. Benutzerdefinierte Datentypen zu XML Extender

Benutzerdefinierte Typen (UDT = User-defined type) sind Datentypen, die von einer DB2®-Anwendung oder einem DB2-Tool erstellt wurden. XML Extender erstellt die folgenden benutzerdefinierten Datentypen zur Verwendung mit XML-Spalten:

- XMLVARCHAR
- XMLCLOB
- XMLFILE

Anhand der Datentypen wird die Spalte in der Anwendungstabelle definiert, in der das XML-Dokument gespeichert wird. Sie können XML-Dokumente auch als Dateien in einem Dateisystem speichern, indem Sie einen Dateinamen angeben.

Alle benutzerdefinierten Typen (UDTs) von XML Extender haben das Qualifikationsmerkmal **DB2XML**, das dem *Schemanamen* der benutzerdefinierten Typen von DB2 UDB XML Extender entspricht. Beispiel:

```
db2xml.XMLVarchar
```

XML Extender erstellt UDTs zum Speichern und Abrufen von XML-Dokumenten. Tabelle 22 beschreibt die UDTs.

Tabelle 22. Die XML Extender-UDTs

Spalte mit benutzerdefinierten Datentypen	**Quellendatentyp**	**Beschreibung**
XMLVARCHAR	VARCHAR(*varchar-länge*)	Speichert ein gesamtes XML-Dokument als VARCHAR in DB2.
XMLCLOB	CLOB(*clob-länge*)	Speichert ein vollständiges XML-Dokument als großes Zeichenobjekt (Character Large Object, CLOB) in DB2.
XMLFILE	VARCHAR(512)	Gibt den Dateinamen des lokalen Dateiservers an. Wird XMLFILE für die XML-Spalte angegeben, speichert XML Extender das XML-Dokument in einer externen Serverdatei. Das Text Extender kann nicht mit XMLFILE aktiviert werden. Sie müssen die Integrität zwischen dem Dateiinhalt und DB2 sowie den für die Indexierung erstellten Nebentabellen sicherstellen.

Hierbei sind *varchar-länge* und *clob-länge* für das betreffende Betriebssystem spezifisch.

Bei XML Extender unter DB2 UDB mit den Betriebssystemen Linux, Unix und Windows® gilt *varchar-länge* = 3K und *clob-länge* = 2G.

Um die Größe von XMLVARCHAR- oder XMLCLOB-UDTs zu ändern, erstellen Sie die betreffenden UDTs, bevor Sie die Datenbank für XML Extender aktivieren.

Vorgehensweise

Um die Größe von XMLVARCHAR oder XMLCLOB einer aktivierten Datenbank zu ändern, gehen Sie wie folgt vor:

1. Sichern Sie alle Daten der betreffenden für XML Extender aktivierten Datenbank.
2. Löschen Sie alle XML-Objektgruppentabellen bzw. XML-Spaltenseitentabellen.
3. Inaktivieren Sie die Datenbank mit Hilfe des Befehls `dxxadm disable_db`.
4. Erstellen Sie den benutzerdefinierten Datentyp XMLVARCHAR oder XMLCLOB.
5. Aktivieren Sie die Datenbank mit Hilfe des Befehls `dxxadm enable_db`.
6. Erstellen und laden Sie die Tabellen erneut.

Diese UDTs werden lediglich zum Angeben der Anwendungsspaltentypen verwendet. Sie gelten nicht für die von XML Extender erstellten Seitentabellen.

Zugehörige Konzepte:

- „XML-Spalten als Speicher- und Zugriffsmethode" auf Seite 80
- „XML-Objektgruppen als Speicher- und Zugriffsmethode" auf Seite 97
- „Verwaltung von XML Extender vorbereiten" auf Seite 39
- „Schemata für die Zuordnung von XML-Objektgruppen" auf Seite 110

Kapitel 8. Benutzerdefinierte XML Extender-Funktionen

Eine benutzerdefinierte Funktion (UDF) ist eine Funktion, die für das Datenbankverwaltungssystem definiert wird und auf die später in SQL-Anweisungen verwiesen wird. In diesem Abschnitt werden die benutzerdefinierten Funktionen beschrieben, die DB2 UDB XML Extender verwendet.

Arten benutzerdefinierter XML Extender-Funktionen

XML Extender bietet Funktionen zum Speichern, Abrufen, Durchsuchen und Aktualisieren von XML-Dokumenten sowie zum Extrahieren von XML-Elementen oder -Attributen. Sie verwenden die benutzerdefinierten XML-Funktionen (UDFs) für XML-Spalten, nicht jedoch für XML-Objektgruppen.

Alle UDFs haben den Schemanamen DB2XML.

Die Arten von XML Extender-Funktionen werden in der folgenden Liste beschrieben:

Speicherfunktionen
Speicherfunktionen fügen intakte XML-Dokumente in für XML aktivierte Spalten als XML-Datentypen ein.

Abruffunktionen
Abruffunktionen rufen XML-Dokumente aus XML-Spalten in einer DB2®-Datenbank ab.

Extraktionsfunktionen
Extraktionsfunktionen extrahieren den Elementinhalt und den Attributwert aus einem XML-Dokument und setzen ihn in den Datentyp um, der im Funktionsnamen angegeben ist. XML Extender bietet eine Gruppe von Extraktionsfunktionen für verschiedene SQL-Datentypen.

Aktualisierungsfunktion
Die Aktualisierungsfunktion ändert ein gesamtes XML-Dokument oder den angegebenen Elementinhalt bzw. Attributwerte und gibt eine Kopie eines XML-Dokuments mit einem aktualisierten Wert zurück, der durch den Standortpfad angegeben ist.

Die benutzerdefinierten XML-Funktionen ermöglichen die Ausführung von Suchfunktionen mit allgemeinen SQL-Datentypen. Darüber hinaus können Sie DB2 UDB Net Search Extender mit XML Extender verwenden, um eine strukturierte Textsuche und eine *Volltextsuche* in XML-Dokumenten durchzuführen. Mit dieser 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.

Einschränkung: Beim Verwenden von Parametermarken in UDFs erfordert eine Java™-Datenbankeinschränkung (JDBC-Einschränkung), dass die Parametermarke für die UDF in den Datentyp der Spalte umgesetzt wird, in die die zurückgegebenen Daten eingefügt werden sollen.

Speicherfunktionen

Übersicht: Speicherfunktionen in XML Extender

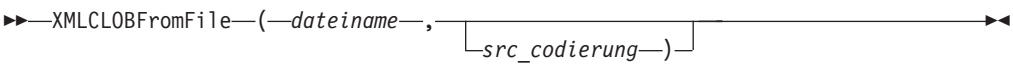
Verwenden Sie die Speicherfunktionen zum Einfügen von XML-Dokumenten in eine DB2 UDB-Datenbank. Sie können die Standardumsetzungsfunktionen eines UDT direkt in den Anweisungen INSERT oder SELECT verwenden. Darüber hinaus bietet XML Extender UDFs, um XML-Dokumente aus anderen Datenquellen als dem UDT-Basisdatentyp zu nehmen und sie in den angegebenen UDT umzusetzen.

Funktion XMLCLOBFromFile()

Zweck:

Liest ein XML-Dokument aus einer Serverdatei 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 Serverdatei.
src_codierung	VARCHAR(100)	Die Codierung einer Quellendatei.

Ergebnisse:
XMLCLOB as LOCATOR
Beispiel:
Mit dem folgenden Beispiel wird ein XML-Dokument aus einer Datei auf einem Server gelesen und als XMLCLOB-Typ in eine XML-Spalte eingefügt. Die Codierung der Serverdatei wird explizit als iso-8859-1 angegeben.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLCLOBFromFile('dxx_install_verz/samples/db2xml
           /xml/getstart.xml
           ', 'iso-8859-1'))
```

Dabei ist *dxx_install_verz* das Verzeichnis, in das XML Extender installiert wurde.

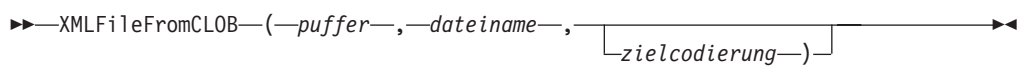
Die Spalte ORDER in der Tabelle SALES_TAB ist als Typ XMLCLOB definiert.

Funktion XMLFileFromCLOB()

Zweck:

Liest ein XML-Dokument als CLOB-Zeigereinheit ein, schreibt es in eine externe Serverdatei und gibt den Dateinamen und -pfad als Typ XMLFILE zurück.

Syntax:



Parameter:

Tabelle 24. 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 Serverdatei.
zielcodierung	VARCHAR(100)	Die Codierung der Ausgabedatei.

Ergebnisse:

XMLFILE

Beispiel:

Das folgende Beispiel liest ein XML-Dokument als CLOB-Querverweis (eine Host-Variable mit einem Wert, der einen einzelnen LOB-Wert im Datenbankserver darstellt) ein, schreibt ihn in eine externe Serverdatei und fügt den Dateinamen und -pfad als Typ XMLFILE in eine XML-Spalte ein. Die Funktion codiert die Ausgabedatei in 'ibm-808'.

```
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, 'dxx_install_verz/samples/db2xml
           /xml/getstart.xml', 'ibm-808'))
```

Dabei ist *dxx_install_verz* das Verzeichnis, in das XML Extender installiert wurde.

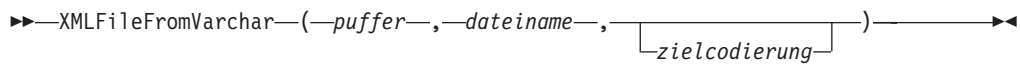
Die Spalte ORDER in der Tabelle SALES_TAB ist als Typ XMLFILE definiert. Wenn sich ein XML-Dokument in Ihrem Puffer befindet, können Sie es in einer Serverdatei speichern.

Funktion XMLFileFromVarchar()

Zweck:

Liest ein XML-Dokument aus dem Speicher als VARCHAR ein, schreibt es in eine externe Serverdatei und gibt den Dateinamen und -pfad als Typ XMLFILE zurück.

Syntax:



Parameter:

Tabelle 25. Parameter für XMLFileFromVarchar

Parameter	Datentyp	Beschreibung
puffer	VARCHAR(3K)	Der Puffer, der das XML-Dokument enthält.
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Serverdatei.
zielcodierung	VARCHAR(100)	Die Codierung der Ausgabedatei.

Ergebnisse:

XMLFILE

Beispiel:

Liest ein XML-Dokument aus dem Speicher als VARCHAR ein, schreibt es in eine externe Serverdatei und fügt den Dateinamen und -pfad als Typ XMLFILE in eine XML-Spalte ein. Die Funktion codiert die Ausgabedatei in 'iso-8859-1'.

```
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, 'dxx_install_verz/samples/db2xml
           /xml/getstart.xml', 'iso-8859-1'))
```

Dabei ist *dxx_install_verz* das Verzeichnis, in das XML Extender installiert wurde.

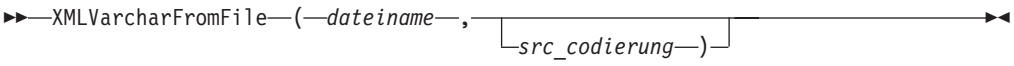
Die Spalte ORDER in der Tabelle SALES_TAB ist als Typ XMLFILE definiert.

Funktion XMLVarcharFromFile()

Zweck:

Liest ein XML-Dokument aus einer Serverdatei und gibt das Dokument als Typ XMLVARCHAR zurück.

Syntax:



Parameter:

Tabelle 26. Parameter für XMLVarcharFromFile

Parameter	Datentyp	Beschreibung
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Serverdatei.
src_codierung	VARCHAR(100)	Die Codierung einer Quellendatei.

Ergebnisse:

XMLVARCHAR

Beispiel:

Mit dem folgenden Beispiel wird ein XML-Dokument aus einer Serverdatei gelesen und als XMLVARCHAR-Typ in eine XML-Spalte eingefügt. Die Codierung der Serverdatei wird explizit als 'ibm-808' angegeben.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLVarcharFromFile('dxx_install_verz/samples/db2xml
           /xml/getstart.xml', 'ibm-808'))
```

Dabei ist *dxx_install_verz* das Verzeichnis, in das XML Extender installiert wurde.

In diesem Beispiel wird ein Datensatz in die Tabelle SALES_TAB eingefügt. Die Funktion XMLVarcharFromFile() importiert das XML-Dokument aus einer Datei, die explizit in 'ibm-808' codiert werden soll, in DB2 UDB und speichert es als XMLVARCHAR.

Abruffunktionen

Abruffunktionen in XML Extender

XML Extender bietet außerdem eine überladene Funktion Content(), die zum Abrufen verwendet wird. Diese überladene Funktion bezieht sich auf eine Gruppe von Abruffunktionen, die denselben Namen haben, sich aber unterschiedlich verhalten, je nachdem, wo die Daten abgerufen werden. Sie können außerdem die Standardumsetzungsfunktionen verwenden, um einen XML-UDT in den Basisdatentyp umzusetzen.

Die Content()-Funktionen ermöglichen 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 Serverdatei gespeichert ist. Sie können mit Content() aus XMLFILE in ein CLOB zu diesem Zweck abrufen.
- **Abrufen aus dem internen Speicher in eine externe Serverdatei**

  Sie können mit Content() auch ein in DB2 UDB gespeichertes XML-Dokument abrufen und es in einer Serverdatei auf dem Dateisystem des DB2 UDB-Servers speichern. Die folgenden Content()-Funktionen werden zum Speichern von Informationen in externen Serverdateien verwendet:
  - Content(): Abrufen von XMLVARCHAR in eine externe Serverdatei
  - Content(): Abrufen von XMLCLOB in eine externe Serverdatei

Die folgenden benutzerdefinierten Funktionen enthalten einen neuen Parameter, der die Codierung der Quellen- oder der Ausgabedatei angibt. Der Wert dieses Parameters ist ein beliebiger Codepagename, der vom ICU-Standard (International Components for Unicode) anerkannt wird.

```
db2xml.XMLVarcharFromFile(filename varchar(512), src_encoding  varchar(100))
returns XMLVarchar

db2xml.XMLCLOBFromFile(filename varchar(512), src_encoding varchar(100))
returns XMLCLOB AS LOCATOR

db2xml.XMLFileFromVarchar(doc varchar(3000), targetfilename varchar(512),
targetencoding varchar(100))
returns XMLFile
```

```
db2xml.XMLFileFromCLOB(doc CLOB(2G) as LOCATOR, targetfilename varchar(512),
 targetencoding varchar(100))
returns XMLFile

db2xml.Content(doc XMLVarchar, targetfilename varchar(512),
 targetencoding varchar(100))
returns varchar(512)

db2xml.Content(doc XMLCLOB as LOCATOR, targetfilename varchar(512),
 targetencoding varchar(100))
returns varchar(512)
```

Beispiele:

Verwenden Sie die folgende Funktion, um den Inhalt einer Datei `/home/collins/xml/entail.xml` in einen Varchar-Puffer zu importieren und um anzugeben, dass die Quellendatei mit iso-8859-1 verschlüsselt wurde:

```
db2xml.XMLVarcharFromFile('/home/collins/xml/entail.xml', 'iso-8859-1')
```

Die Datei wird in einen Varchar-Puffer importiert und von iso-8859-1 in die Codepage der Datenbank umgewandelt.

Verwenden Sie die folgende Funktion, um einen Varchar-Puffer in eine Datei /home/raskolnikov/xml/confession.xml zu exportieren und um anzugeben, dass die Ausgabedatei mit ibm-808 verschlüsselt werden muss:

```
db2xml.Content('<sequence><thought>I did it!</thought></sequence>',
'/home/raskolnikov/xml/confession.xml', 'ibm-808')
```

Die Pufferinhalte werden in eine Datei exportiert und von der Datenbankcodepage in ibm-808 umgewandelt. Die Codierungsdeklaration der XML-Datei wird dann entsprechend aktualisiert.

Bei den Beispielen im folgenden Abschnitt wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

Content(): Abrufen von XMLFILE in ein CLOB

Zweck:

Ruft Daten aus einer Serverdatei ab und speichert sie in einem CLOB LOCATOR.

Syntax:

Parameter:

Tabelle 27. Parameter für XMLFILE in CLOB

Parameter	Datentyp	Beschreibung
xmlobj	XMLFILE	Das XML-Dokument.

Ergebnisse:
CLOB (*clob_länge*) as LOCATOR

clob_länge für DB2 UDB ist 2 GB.

Beispiel:

Mit dem folgenden Beispiel werden Daten aus einer Serverdatei abgerufen und in einer CLOB-Zeigereinheit gespeichert.

```
char    subsystem[20];
long   retcode = 0, reason = 0;
extern "OS" { int DSNALI(char * functn, ...); }

extern "OS" short DSNTIAR(struct sqlca *sqlca,
                          error_struct *error_message,
                          long *data_len);

EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

/* Verbindung zu Subsystem herstellen */
rc = DSNALI("OPEN        ", subsystem, "PLANNAME",
     &retcode, &reason);
if ( retcode != 0 )
{
   /* Fehlernachricht drucken */
      goto exit;
}

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;

/* Verbindung zu Subsystem unterbrechen */
DSNALI("CLOSE       ", "SYNC", &retcode, &reason);
if ( retcode != 0 ) {
   /* Fehlernachricht drucken */
}
```

Die Spalte ORDER in der Tabelle SALES_TAB hat den Typ XMLFILE, so dass die UDF Content() Daten aus einer Serverdatei abruft und in einer CLOB-Zeigereinheit speichert.

Zugehörige Tasks:

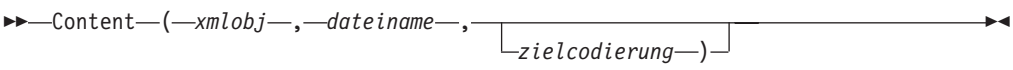
- „Daten in XML-Objektgruppen aktualisieren und löschen” auf Seite 106

Content(): Abrufen von XMLVARCHAR in eine externe Serverdatei

Zweck:

Ruft den als Typ XMLVARCHAR gespeicherten XML-Inhalt ab und speichert ihn in einer externen Serverdatei.

Syntax:



Wichtig: Wenn bereits eine Datei mit dem angegebenen Namen vorhanden ist, überschreibt die Content-Funktion ihren Inhalt.

Parameter:

Tabelle 28. Parameter für XMLVarchar in externe Serverdatei

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR	Das XML-Dokument.
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Serverdatei.
zielcodierung	VARCHAR(100)	Die Codierung der Ausgabedatei.

Ergebnisse:

VARCHAR(512)

Beispiel:

Im folgenden Beispiel wird der als Typ XMLVARCHAR gespeicherte XML-Inhalt abgerufen und in einer externen Datei gespeichert, die sich auf dem Server befindet. UDF codiert die Datei in 'ibm-808'.

```
CREATE table app1 (id int NOT NULL, order DB2XML.XMLVarchar);
INSERT into app1 values (1, '<?xml version="1.0"?>
   <!DOCTYPE SYSTEM "dxx_install_verz/samples/db2xml/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>
          </Part>
     </Order>');

SELECT DB2XML.Content(order, 'dxx_install_verz/samples/dad/getstart_column.dad'
, 'ibm-808')
  from app1 where ID=1;
```

Zugehörige Tasks:

- „Methode zum Abrufen eines XML-Dokuments" auf Seite 85

Zugehörige Referenzen:

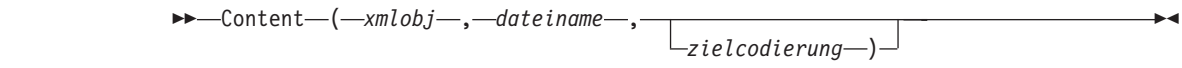
- „Abruffunktionen in XML Extender" auf Seite 149

Content(): Abrufen von XMLCLOB in eine externe Serverdatei

Zweck:

Ruft den als Typ XMLCLOB gespeicherten XML-Inhalt ab und speichert ihn in einer externen Serverdatei.

Syntax:



Wichtig: Wenn bereits eine Datei mit dem angegebenen Namen vorhanden ist, überschreibt die Content-Funktion ihren Inhalt.

Parameter:

Tabelle 29. Parameter für XMLCLOB in externe Serverdatei

Parameter	Datentyp	Beschreibung
xmlobj	XMLCLOB as LOCATOR	Das XML-Dokument.
dateiname	VARCHAR(512)	Der vollständig qualifizierte Name der Serverdatei.
zielcodierung	VARCHAR(100)	Die Codierung der Ausgabedatei.

Ergebnisse:

VARCHAR(512)

Beispiel:

Im folgenden Beispiel wird der als Typ XMLCLOB gespeicherte XML-Inhalt abgerufen und in einer externen Datei gespeichert, die sich auf dem Server befindet. UDF codiert die Datei in 'ibm-808'.

```
CREATE table app1 (id int NOT NULL, order DB2XML.XMLCLOB not logged);

INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM "dxx_install_verz/samples/db2xml/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>
        </Part>
    </Order>');

SELECT DB2XML.Content(order,
'dxx_install_verz/samples/db2xml/xml/getstart.xml', 'ibm-808')
 from app1 where ID=1;
```

Extraktionsfunktionen

Extraktionsfunktionen in XML Extender

Die Extraktionsfunktionen extrahieren den Elementinhalt oder den Attributwert aus einem XML-Dokument und geben die angeforderten SQL-Datentypen zurück. 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; ersterer wird als Skalarfunktion, letzterer als Tabellenfunktion bezeichnet.

Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

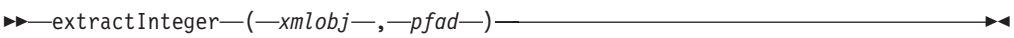
extractInteger() und extractIntegers()

Zweck:

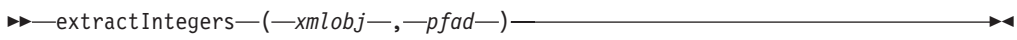
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ INTEGER zurück.

Syntax:

Skalarfunktion:



Tabellenfunktion:



Parameter:

Tabelle 30. Funktionsparameter für extractInteger und extractIntegers

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
INTEGER
Rückkehrcodes:
returnedInteger
Beispiele:
Skalarfunktion, Beispiel:

Im folgenden Beispiel wird ein Wert zurückgegeben, wenn der Attributwert für key = ″1″ ist. Der Wert wird als INTEGER extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2″ eingeben müssen.

```
CREATE TABLE t1(key INT);
INSERT INTO t1 values (
    DB2XML.extractInteger(DB2XML.XMLFile('/samples/db2xml
    /xml/getstart.xml'),
    '/Order/Part[@color="black "]/key'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird jeder Bestellschlüssel für die Bestellungen als INTEGER extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2″ eingeben müssen.

```
SELECT *
FROM TABLE(
  DB2XML.extractIntegers(DB2XML.XMLFile('/samples/db2xml/xml/getstart.xml'),
'/Order/Part/key'))   AS X;
```

Zugehörige Konzepte:

- Anhang D, „UDT- und UDF-Namen für XML Extender″, auf Seite 333
- „Arten benutzerdefinierter XML Extender-Funktionen″ auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender″ auf Seite 154

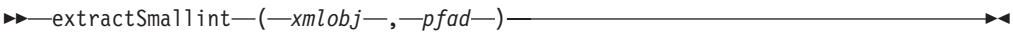
extractSmallint() und extractSmallints()

Zweck:

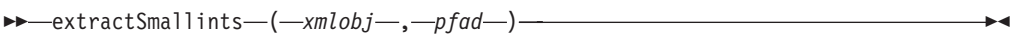
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ SMALLINT zurück.

Syntax:

Skalarfunktion:



Tabellenfunktion:



Parameter:

Tabelle 31. Funktionsparameter für extractSmallint und extractSmallints

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
SMALLINT
Rückkehrcodes:
returnedSmallint

Beispiele:

Skalarfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von key in allen Bestellungen als SMALLINT extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

```
CREATE TABLE t1(key INT);
INSERT INTO t1 values (
    DB2XML.extractSmallint(db2xml.xmlfile('dxx_install_verz
     /samples/db2xml/xml/getstart.xml'),
    '/Order/Part[@color="black "]/key'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von key in allen Bestellungen als SMALLINT extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

```
SELECT *
  FROM TABLE(
      DB2XML.extractSmallints(DB2XML.XMLFile('dxx_install_verz
      /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/key'))   AS X;
```

Zugehörige Konzepte:

- „Indizes für XML-Spaltendaten verwenden" auf Seite 81
- Anhang D, „UDT- und UDF-Namen für XML Extender", auf Seite 333
- „Arten benutzerdefinierter XML Extender-Funktionen" auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154
- „Rückkehrcodes von gespeicherten Prozeduren von XML Extender" auf Seite 291

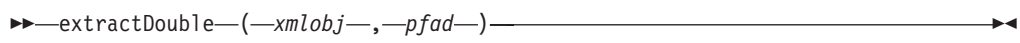
extractDouble() und extractDoubles()

Zweck:

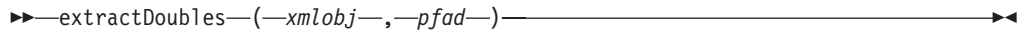
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ DOUBLE zurück.

Syntax:

Skalarfunktion:



Tabellenfunktion:



Parameter:

Tabelle 32. Funktionsparameter für extractDouble und extractDoubles

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
DOUBLE
Rückkehrcodes:
returnedDouble
Beispiele: Skalarfunktion, Beispiel:
Mit dem folgenden Beispiel wird der Preis in einer Bestellung automatisch von einem Typ DOUBLE in den Typ DECIMAL umgesetzt. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

```
CREATE TABLE t1(price DECIMAL(9,2));
INSERT INTO t1 values (
    DB2XML.extractDouble(DB2XML.xmlfile('dxx_install_verz
    /samples/db2xml/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. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang jedes Befehls nicht DB2 UDB eingeben müssen.

```
SELECT CAST(RETURNEDDOUBLE AS DOUBLE)
  FROM TABLE(
      DB2XML.extractDoubles(DB2XML.XMLFile('dxx_install_verz
       /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/ExtendedPrice'))   AS X;
```

Zugehörige Konzepte:

- Anhang D, „UDT- und UDF-Namen für XML Extender", auf Seite 333

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154

extractReal() und extractReals()

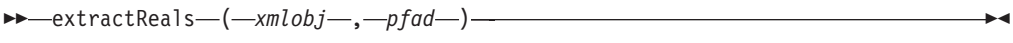
Zweck:

Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ REAL zurück.

Syntax:

Skalarfunktion:

Tabellenfunktion:



Parameter:

Tabelle 33. Funktionsparameter für extractReal und extractReals

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
REAL
Rückkehrcodes:
returnedReal
Beispiele:
Skalarfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von ExtendedPrice als REAL extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

```
CREATE TABLE t1(price DECIMAL(9,2));
INSERT INTO t1 values (
    DB2XML.extractReal(DB2XML.xmlfile('dxx_install_verz
    /samples/db2xml/xml/getstart.xml'),
    '/Order/Part[@color="black "]/ExtendedPrice'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von ExtendedPrice als REAL extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

```
SELECT CAST(RETURNEDREAL AS REAL)
  FROM TABLE(
      DB2XML.extractReals(DB2XML.XMLFile('dxx_install_verz
      /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/ExtendedPrice'))   AS X;
```

Zugehörige Konzepte:

- Anhang D, „UDT- und UDF-Namen für XML Extender", auf Seite 333
- „Arten benutzerdefinierter XML Extender-Funktionen" auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154
- „UDF-Rückkehrcodes von XML Extender" auf Seite 290

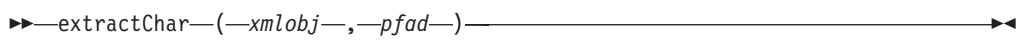
extractChar() und extractChars()

Zweck:

Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ CHAR zurück.

Syntax:

Skalarfunktion:



Tabellenfunktion:

Parameter:

Tabelle 34. Funktionsparameter für extractChar und extractChars

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
CHAR
Rückkehrcodes:
returnedChar
Beispiele:
Skalarfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von Name als CHAR extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang jedes Befehls nicht „DB2" eingeben müssen.

```
CREATE TABLE t1(name char(30));
INSERT INTO t1 values (
    DB2XML.extractChar(DB2XML.xmlfile('dxx_install_verz
    /samples/db2xml/xml/getstart.xml'),
    '/Order/Customer/Name'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von Color als CHAR extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang jedes Befehls nicht „DB2" eingeben müssen.

```
SELECT *
  FROM TABLE(
      DB2XML.extractChars(DB2XML.XMLFile('dxx_install_verz
       /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/@color'))   AS X;
```

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154
- „Syntaxdiagramme lesen" auf Seite ix

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 35. Funktionsparameter für extractVarchar und extractVarchars

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
VARCHAR(4K)
Rückkehrcodes:
returnedVarchar

Beispiele:
Skalarfunktion, 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
```

Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen. 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.

Tabellenfunktion, 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
```

Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen. 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.

Skalarfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von Name als VARCHAR extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

```
CREATE TABLE t1(name varchar(30));
INSERT INTO t1 values (
    DB2XML.extractVarchar(DB2XML.xmlfile('dxx_install_verz
     /samples/db2xml/xml/getstart.xml'),
    '/Order/Customer/Name'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von Color als VARCHAR extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

```
SELECT*
  FROM TABLE(
      DB2XML.extractVarchars(DB2XML.XMLFile('dxx_install_verz
      /samples/xml/getstart.xml'),
      '/Order/Part/@color'))   AS X;
```

Zugehörige Konzepte:

- Anhang D, „UDT- und UDF-Namen für XML Extender", auf Seite 333
- „Arten benutzerdefinierter XML Extender-Funktionen" auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154
- „UDF-Rückkehrcodes von XML Extender" auf Seite 290

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, die nur den Inhalt der Elemente und Attribute zurückgeben. Die Funktionen extractClob(s) werden zum Extrahieren von Dokumentteilen verwendet, während extractVarchar(s) und extractChar(s) zum Extrahieren einfacher Werte verwendet werden.

Syntax:

Skalarfunktion:

Tabellenfunktion:

Parameter:

Tabelle 36. Funktionsparameter für extractCLOB und extractCLOBs

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
CLOB(10K)
Rückkehrcodes:
returnedCLOB
Beispiele:
Skalarfunktion, Beispiel:

In diesem Beispiel werden Inhalt und Befehle aller name-Elemente aus einer Bestellung extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang jedes Befehls nicht „DB2" eingeben müssen.

```
CREATE TABLE t1(name DB2XML.xmlclob);
INSERT INTO t1 values (
    DB2XML.extractClob(DB2XML.xmlfile('dxx_install_verz
     /samples/db2xml/xml/getstart.xml'),
    '/Order/Customer/Name'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

In diesem Beispiel werden alle color-Attribute aus einer Bestellung extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang jedes Befehls nicht „DB2" eingeben müssen.

```
SELECT *
  FROM TABLE(
      DB2XML.extractCLOBs(DB2XML.XMLFile('dxx_install_verz
     /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/@color'))   AS X;
```

Zugehörige Konzepte:

- „Arten benutzerdefinierter XML Extender-Funktionen" auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154

extractDate() und extractDates()

Zweck:

Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ DATE zurück. Das Datum muss im folgenden Format angegeben sein: JJJJ-MM-TT.

Syntax:

Skalarfunktion:

Tabellenfunktion:

Parameter:

Tabelle 37. Funktionsparameter für extractDate und extractDates

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
DATE
Rückkehrcodes:
returnedDate

Beispiele:
Skalarfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von ShipDate als DATE extrahiert. Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang jedes Befehls nicht „DB2" eingeben müssen.

```
CREATE TABLE t1(shipdate DATE);
INSERT INTO t1 values (
    DB2XML.extractDate(DB2XML.xmlfile('dxx_install_verz
      /samples/db2xml/xml/getstart.xml'),
    '/Order/Part[@color="red   "]/Shipment/ShipDate'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

Im folgenden Beispiel wird der Wert von ShipDate als DATE extrahiert.

```
SELECT *
  FROM TABLE(
      DB2XML.extractDates(DB2XML.XMLFile('dxx_install_verz
      /samples/db2xml/xml/getstart.xml'),
      '/Order/Part[@color="black "]/Shipment/ShipDate'))   AS X;
```

Zugehörige Konzepte:

- „Arten benutzerdefinierter XML Extender-Funktionen" auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154
- „UDF-Rückkehrcodes von XML Extender" auf Seite 290

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 38. Funktionsparameter für extractTime und extractTimes

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
TIME
Rückkehrcodes:
returnedTime
Beispiele:
Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

Skalarfunktion, Beispiel:

```
CREATE TABLE t1(testtime  TIME);
INSERT INTO t1 values (
    DB2XML.extractTime(DB2XML.XMLCLOB(
         '<stuff><data>11.12.13</data></stuff>'), '//data'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

```
select *
from table(
    DB2XML.extractTimes(DB2XML.XMLCLOB(
        '<stuff><data>01.02.03</data><data>11.12.13</data></stuff>'),
        '//data')) as x;
```

Zugehörige Konzepte:

- Anhang D, „UDT- und UDF-Namen für XML Extender", auf Seite 333
- „Arten benutzerdefinierter XML Extender-Funktionen" auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154

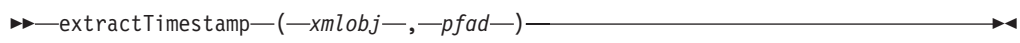
extractTimestamp() und extractTimestamps()

Zweck:

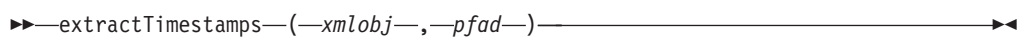
Extrahiert den Elementinhalt oder Attributwert aus einem XML-Dokument und gibt die Daten als Typ TIMESTAMP zurück.

Syntax:

Skalarfunktion:



Tabellenfunktion:



Parameter:

Tabelle 39. Funktionsparameter für extractTimestamp und extractTimestamps

Parameter	Datentyp	Beschreibung
xmlobj	XMLVARCHAR, XMLFILE oder XMLCLOB	Der Spaltenname.
pfad	VARCHAR	Der Standortpfad des Elements oder Attributs.

Zurückgegebener Typ:
TIMESTAMP
Rückkehrcodes:
returnedTimestamp
Beispiele:
Bei den Beispielen wird davon ausgegangen, dass Sie mit der DB2 UDB-Befehlsshell arbeiten, bei der Sie am Anfang eines Befehls nicht „DB2" eingeben müssen.

Skalarfunktion, Beispiel:

```
CREATE TABLE t1(testtimestamp  TIMESTAMP);
INSERT INTO t1 values (
    DB2XML.extractTimestamp(DB2XML.XMLCLOB(
        '<stuff><data>2003-11-11-11.12.13.888888</data></stuff>'),
        '//data'));
SELECT * from t1;
```

Tabellenfunktion, Beispiel:

```
select * from
table(DB2XML.extractTimestamps(DB2XML.XMLClob(
    '<stuff><data>2003-11-11-11.12.13.888888
    </data><data>2003-12-22-11.12.13.888888</data></stuff>'),
    '//data')) as x;
```

XML Extender normalisiert Zeitmarken automatisch, die aus XML-Dokumenten extrahiert wurden, damit sie, wenn nötig, mit dem DB2-Zeitmarkenformat übereinstimmen. Zeitmarken werden in das Format `jjjj-mm-tt-hh.mm.ss.nnnnnn` oder `jjjj-mm-tt-hh mm.ss.nnnnnn` normalisiert. Beispiel:

```
2003-1-11-11.12.13
```

wird wie folgt normalisiert:

```
2003-01-11-11.12.13.000000
```

Zugehörige Konzepte:

- Anhang D, „UDT- und UDF-Namen für XML Extender", auf Seite 333
- „Arten benutzerdefinierter XML Extender-Funktionen" auf Seite 145

Zugehörige Referenzen:

- „Extraktionsfunktionen in XML Extender" auf Seite 154
- „UDF-Rückkehrcodes von XML Extender" auf Seite 290

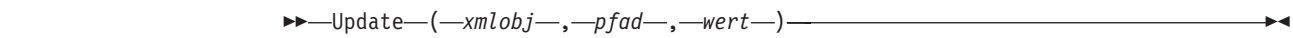
Aktualisierungsfunktionen in XML Extender

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.

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 40. 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 Aktualisierungszeichenfolge. **Einschränkung:** Die Aktualisierungsfunktion verfügt über keine Option, mit der eine Ausgabe-Escape-Operation inaktiviert werden kann. Die Ausgabe eines extractClob (ein Fragment mit Befehlen) kann mit dieser Funktion nicht eingefügt werden. Verwenden Sie nur Textwerte.

Einschränkung: Die Update-UDF unterstützt Standortpfade, die Prädikate mit Attributen 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 = db2xml.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 verloren gehen 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.

XML-Dokument durch die Funktion Update() verarbeiten

Wenn die Funktion Update() XML-Dokumente ersetzt, muss sie das Dokument entsprechend der Ausgabe des XML-Parsers wiederherstellen. Tabelle 41 beschreibt mit Beispielen, wie die Teile des Dokuments verarbeitet werden.

Tabelle 41. 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 41. Regeln der Aktualisierungsfunktion (Forts.)

Element oder Knotentyp	Codebeispiel zum XML-Dokument	Status nach der Aktualisierung
DOCTYPE-Deklaration		Die Dokumentartdeklaration wird beibehalten: • Der Stammelementname wird unterstützt. • Allgemeine ExternalIDs und ExternalIDs des System werden beibehalten und angezeigt, wenn sie im Originaldokument angegeben sind. • Interne DTD-Untergruppe wird *nicht* beibehalten. Entitäten werden ersetzt; Standardwerte für Attribute werden verarbeitet und erscheinen in den Ausgabedokumenten. • 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' (externe ID) oder interne DTD-Untergruppe enthält. Nach einer Aktualisierung fehlt in diesem Fall die 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 ] >`	

Verarbeitungsanweisungen	`<?xml-stylesheet`
	`  title="compact"`
	`  href="datatypes1.xsl"`
	`  type="text/xsl"?>`
		Verarbeitungsanweisungen werden beibehalten.

Tabelle 41. Regeln der Aktualisierungsfunktion (Forts.)

Element oder Knotentyp	Codebeispiel zum XML-Dokument	Status nach der Aktualisierung
Kommentare	`<!-- comment -->`	Kommentare innerhalb des Stammelements werden beibehalten. Kommentare außerhalb des Stammelements werden gelöscht.

Elemente	`<books>`
	`Inhalt`
	`</books>`
		Elemente werden beibehalten.

Attribute	`id='1' date="01/02/2003"`	Attribute von Elementen werden beibehalten. • Nach der Aktualisierung werden doppelte Anführungszeichen zur Beschreibung von Werten verwendet. • Daten in Attributen gehen verloren. • Entitäten werden ersetzt.
Textknoten	`This section is about my dog &mydog;.`	Textknoten (Elementinhalt) werden beibehalten. • Daten innerhalb von Textknoten gehen verloren. • 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. Wenn also ein Dokument in mehreren Standortpfaden vorkommt, ersetzt die Funktion Update() die vorhandenen Werte durch den Wert im Parameter *wert*.

Sie können ein Prädikat im Parameter *pfad* angeben, um eindeutige Standortpfade bereitzustellen und versehentliche Aktualisierungen zu verhindern. Die UDF Update() unterstützt Standortpfade, die über Prädikate mit Attributen verfügen, jedoch nicht mit Elementen.

Beispiele

Die folgenden Beispiele zeigen Exemplare eines XML-Dokuments vor und nach einer Aktualisierung.

Tabelle 42. XML-Dokumente 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="07/01/2003">
       <!-- first section -->
       <section>This is a section in Chapter
         One.</section>
  </chapter>
  <chapter id="2" date="01/02/2003">
       <section>This is a section in Chapter
         Two.</section>
        <footnote>A footnote in Chapter Two
         is here.</footnote>
  </chapter>
  <price date="12/22/2003" time="11.12.13"
       timestamp="2003-12-22-11.12.13.888888">
       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>

Tabelle 42. XML-Dokumente vor und nach einer Aktualisierung (Forts.)

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

Überprüfungsfunktionen

DB2 XML Extender bietet zwei benutzerdefinierte Funktionen (UDFs), die XML-Dokumente unter Verwendung eines XML-Schemas oder einer DTD überprüfen.

Ein Element in einem XML-Dokument ist gemäß eines bestimmten Schemas gültig, wenn die zugehörigen Elementartregeln erfüllt sind. Wenn alle Elemente gültig sind, ist das gesamte Dokument gültig. Mit einer DTD kann jedoch kein spezifi-

sches Stammelement angefordert werden. Die Überprüfungsfunktionen geben 1 zurück, wenn das Dokument gültig ist, bzw. geben 0 zurück und schreiben eine Fehlernachricht in die Tracedatei, wenn das Dokument ungültig ist. Die Funktionen sind:

db2xml.svalidate:
Überprüft ein XML-Dokumentexemplar anhand des angegebenen Schemas.

db2xml.dvalidate:
Überprüft ein XML-Dokumentexemplar anhand der angegebenen DTD.

Funktion SVALIDATE()

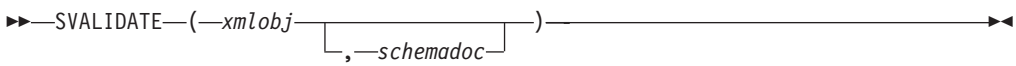
Diese Funktion überprüft ein XML-Dokument anhand eines angegebenen Schemas (oder des im XML-Dokument benannten Schemas) und gibt 1 zurück, wenn das Dokument gültig ist, oder 0, wenn es ungültig ist. Diese Funktion setzt voraus, dass ein XML-Dokument und ein Schema auf dem Dateisystem oder als CLOB in DB2 vorhanden sind.

Stellen Sie vor der Ausführung der Funktion SVALIDATE sicher, dass XML Extender für Ihre Datenbank aktiviert ist. Führen Sie hierzu folgenden Befehl aus:

```
dxxadm enable_db mein_datenbankname
```

Wenn die Prüfung des XML-Dokuments fehlschlägt, wird eine Fehlernachricht in die XML Extender-Tracedatei geschrieben. Aktivieren Sie den Trace, bevor Sie den Befehl SVALIDATE ausführen.

Syntax



Parameter

Tabelle 43. Parameter für SVALIDATE

Parameter	Datentyp	Beschreibung
xmlobj	VARCHAR(256)	Der Dateipfad des zu überprüfenden XML-Dokuments.
	CLOB(2G)	Die XML-Spalte mit dem zu überprüfenden Dokument.
schemadoc	VARCHAR(256)	Der Dateipfad des Schemadokuments.
	CLOB(2G)	Die XML-Spalte mit dem Schema.

Beispiele

Beispiel 1: Dieses Beispiel überprüft `equiplog2001.xml` anhand des Schemas, das im Dokument angegeben ist.

```
db2 values db2xml.svalidate("/home/jean/xml/equiplog2001.xml")
```

Beispiel 2: Dieses Beispiel überprüft ein XML-Dokument anhand des angegebenen Schemas. Sowohl das Dokument als auch das Schema werden in DB2 UDB-Tabellen gespeichert.

```
db2 select db2xml.svalidate(doc,schema) from equiplogs where id=1
```

Funktion DVALIDATE()

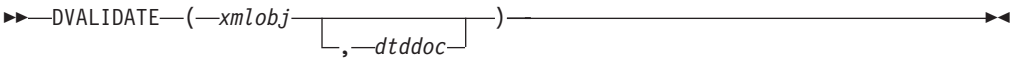
Diese Funktion überprüft ein XML-Dokument anhand einer angegebenen DTD (oder anhand einer im XML-Dokument benannten DTD) und gibt 1 zurück, wenn das Dokument gültig ist, oder 0, wenn es ungültig ist. Diese Funktion setzt voraus, dass ein XML-Dokument und eine DTD auf dem Dateisystem oder als CLOB in DB2 vorhanden sind.

Stellen Sie vor der Ausführung der Funktion DVALIDATE sicher, dass XML Extender für die Datenbank aktiviert ist. Führen Sie hierzu folgenden Befehl aus:

```
dxxadm enable_db meinedb
```

Wenn die Prüfung des XML-Dokuments fehlschlägt, wird in die XML Extender-Tracedatei eine Fehlernachricht geschrieben. Aktivieren Sie den Trace, bevor Sie den Befehl DVALIDATE ausführen.

Syntax



Parameter

Tabelle 44. Parameter für DVALIDATE

Parameter	Datentyp	Beschreibung
xmlobj	VARCHAR(256)	Der Dateipfad des zu überprüfenden XML-Dokuments.
	CLOB(2G)	Die XML-Spalte mit dem zu überprüfenden Dokument.
dtddoc	VARCHAR(256)	Der Dateipfad des DTD-Dokuments.
	CLOB(2G)	Die XML-Spalte mit der DTD, die entweder aus der Tabelle DTD_REF oder aus einer normalen Tabelle stammt.

Beispiele

Beispiel 1: Dieses Beispiel überprüft `equiplog2001.xml` anhand der DTD, die im Dokument angegeben ist.

```
db2 values db2xml.dvalidate(/home/jean/xml/equiplog2001.xml)
```

Beispiel 2: Dieses Beispiel überprüft ein XML-Dokument anhand der angegebenen DTD. Sowohl das Dokument als auch die DTD sind im Dateisystem gespeichert.

```
db2 values db2xml.dvalidate (c:/xml/equiplog.xml,c:/xml/dtds/equip.dtd)
```

Beispiel 3: Dieses Beispiel überprüft ein XML-Dokument anhand der angegebenen DTD. Sowohl das Dokument als auch die DTD sind in DB2 UDB-Tabellen gespeichert.

```
db2 values db2xml.dvalidate (doc,dtdid) from equiplogs, db2xml.dtd_ref \
    where dtdid="equip.dtd"
```

Zugehörige Referenzen:

- „Trace für XML Extender starten" auf Seite 289

Kapitel 9. DAD-Dateien (Dokumentzugriffsdefinitionsdateien)

DAD-Datei für XML-Spalten erstellen

Diese Task ist Teil der umfangreicheren Task zum Definieren und Aktivieren einer XML-Spalte.

Die aktuellen Informationen zu DAD-Dateien finden Sie auf der XML Extender-Website unter
www.ibm.com/software/data/db2/extenders/xmlext/downloads.html.

Zum Zugreifen auf Ihre XML-Daten und zum Aktivieren von Spalten für XML-Daten in einer XML-Tabelle müssen Sie eine DAD-Datei (DAD = Document Access Definition; Dokumentzugriffsdefinition) definieren. Diese Datei definiert die Attribute und Schlüsselelemente Ihrer Daten, die innerhalb der Spalte durchsucht werden müssen. Für XML-Spalten gibt die DAD-Datei in erster Linie an, wie die darin gespeicherten Dokumente indexiert werden sollen. Die DAD-Datei gibt außerdem eine DTD oder ein Schema an, die zur Überprüfung von Dokumenten, die in die XML-Spalte eingefügt werden, verwendet werden. DAD-Dateien werden als Datentyp CLOB gespeichert und ihre Größe ist auf 100 KB begrenzt.

Voraussetzungen:

Bevor Sie die DAD-Datei erstellen, müssen Sie folgende Aktionen ausführen:

- Entscheiden, welche Elemente oder Attribute bei der Suche häufig verwendet werden. Die Elemente oder Attribute, die Sie angeben, werden in die Nebentabellen für Schnellsuchvorgänge durch XML Extender extrahiert.
- Den Standortpfad definieren, der die einzelnen Elemente oder Attribute, die in einer Nebentabelle indexiert sind, repräsentiert. Sie müssen außerdem den Datentyp angeben, in den das Element oder Attribut umgesetzt werden soll.

Vorgehensweise:

Gehen Sie wie folgt vor, um eine DAD-Datei zu erstellen:

1. Erstellen Sie ein neues Dokument in einem Texteditor, und verwenden Sie folgende Syntax:

   ```
   <?XML version="1.0"?>
   <!DOCTYPE DAD SYSTEM <"pfad/dtd/dad.dtd">.
   ```

   "path/dtd/dad.dtd " ist der Pfad- und Dateiname der DTD für die DAD-Datei. Eine DTD wird in `dxx_install_verz\samples\db2xml\dtd` zur Verfügung gestellt.
2. Fügen Sie DAD-Befehle nach den Zeilen aus Schritt 1 ein.

   ```
   <DAD>
   </DAD>
   ```

   Dieses Element enthält alle anderen Elemente.

3. Geben Sie die Überprüfung für das Dokument und die Spalte an:
   - Wenn Ihr gesamtes XML-Dokument anhand einer DTD oder eines Schemas geprüft werden soll, bevor es in die Datenbank eingefügt wird:
     – Fügen Sie den entsprechenden Befehl ein, um anzugeben, wie das Dokument überprüft werden soll:

       ```
       <dtdid>pfad/dtd_name.dtd</dtdid>
       ```

     – Fügen Sie den folgenden Befehl ein, um das Dokument mit Hilfe eines Schemas zu überprüfen:

       ```
       <schemabindings>
       <nonamespacelocation location="pfad/schema_name.xsd"/>
       </schemabindings>
       ```

     – Überprüfen Sie die Spalten, indem Sie folgenden Befehl einfügen:

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

   - Wenn Sie das Dokument nicht überprüfen wollen, verwenden Sie folgenden Befehl:

     ```
     <validation>NO</validation>
     ```

4. Fügen Sie die Befehle <Xcolumn> </Xcolumn> ein, um anzugeben, dass Sie XML-Spalten als Zugriffs- und Speichermethode für Ihre XML-Daten verwenden.
5. Geben Sie Nebentabellen an. Führen Sie für jede Nebentabelle, die Sie erstellen wollen, folgende Schritte aus:
   a. Geben Sie die Befehle <table> </table> an. Beispiel:

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

   b. Fügen Sie innerhalb der Tabellenbefehle jeweils einen Befehl <column> für jede Spalte ein, die die Nebentabelle enthalten soll. Jede Spalte hat vier Attribute: name (Name), type (Typ), path (Pfad) und multi_occurrence (mehrfaches Vorkommen).

      Beispiel:

      ```
      <table name="person_names">>
      <column name ="fname"
           type="varchar(50)"
           path="/person/firstName"
           multi_occurrence="NO"/>
      <column name ="lname"
           type="varchar(50)"
           path="/person/lastName"
           multi_occurrence="NO"/>
      </table>
      ```

      Hierbei gilt Folgendes:

      name Gibt den Namen der Spalte an, die in der Nebentabelle erstellt wird.

      type Gibt den SQL-Datentyp in der Nebentabelle für jedes indexierte Element oder Attribut an.

      path Gibt den Standortpfad im XML-Dokument für jedes zu indexierende Element oder Attribut an.

multi_occurrence
Gibt an, ob das Element oder Attribut, auf das durch das Pfadattribut verwiesen wird, mehr als einmal im XML-Dokument vorkommen kann. Die möglichen Werte für **multi_occurrence** sind **YES** oder **NO**. Wenn der Wert **NO** ist, können mehrere Spalten pro Tabelle angegeben werden. Wenn der Wert **YES** ist, können Sie nur eine Spalte in der Nebentabelle angeben.

6. Speichern Sie Ihre Datei mit der Erweiterung DAD.

Das folgende Beispiel zeigt eine vollständige DAD-Datei:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx_install_verz\samples\db2xml\dtd\dad.dtd">
<DAD>
<dtid>C:\SG246130\code\person.dtd</dtdid>
<validation>YES</validation>
<Xcolumn>
  <table name="person_names">
   <column name="fname"
     type="varchar(50)"
     path="/person/firstName"
     multi_occurrence="NO"/>
   <column name="lname"
     type="varchar(50)"
     path="/person/lastName"
     multi_occurrence="NO"/>
</table>
<table name="person_phone_number">
   <column name="pnumber"
     type="varchar(20)"
     path="/person/phone/number"
     multi_occurrence="YES"/>
</table>
<table name="person_phone_number">
   <column name="pnumber"
     type="varchar(20)"
     path="/person/phone/number"
     multi_occurrence="YES"/>
</table>
<table name="person_phone_type">
   <column name="ptype"
     type="varchar(20)"
     path="/person/phone/type"
     multi_occurrence="YES"/>
</table>
<Xcolumn>
</DAD>
```

Nachdem Sie eine DAD-Datei erstellt haben, ist der nächste Schritt beim Definieren und Aktivieren einer XML-Spalte, die Tabelle zu erstellen, in der Ihre XML-Dokumente gespeichert werden.

Zugehörige Konzepte:

- „XML-Objektgruppen als Speicher- und Zugriffsmethode" auf Seite 97
- „DAD-Dateien für XML-Objektgruppen" auf Seite 178
- „DAD-Prüfprogramm" auf Seite 192

Zugehörige Tasks:

- „DAD-Prüfprogramm verwenden" auf Seite 193

DAD-Dateien für XML-Objektgruppen

Für XML-Objektgruppen ordnet die DAD-Datei die Struktur des XML-Dokuments den DB2-Tabellen zu, aus denen Sie das Dokument zusammensetzen. Sie können mit Hilfe der DAD-Datei auch Dokumente in DB2 UDB-Tabellen zerlegen.

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

Sie müssen die DAD-Datei entweder beim Aktivieren einer Objektgruppe oder beim Verwenden der DAD-Datei in den gespeicherten Prozeduren für XML-Objektgruppen 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 von XML Extender verwendet wird, hat sie den Datentyp CLOB. Diese Datei kann bis zu 100 KB groß sein.

Zum Angeben der Zugriffs- und Speichermethode "XML-Spalte" verwenden Sie den Befehl <Xcollection> in Ihrer 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 eine Gruppe relationaler Tabellen, die XML-Daten enthalten. Anwendungen können eine XML-Objektgruppe beliebiger Benutzertabellen aktivieren. Diese Benutzertabellen können Tabellen mit vorhandenen Geschäftsdaten sein oder Tabellen, die XML Extender kürzlich erstellt hat.

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. 14 auf Seite 179 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:\dxx\samples\db2xml\dtd\dad.dtd">
<DAD>
  ...
<Xcollection>
<SQL_stmt>
       ...
</SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM
                 ""c:\dxx\samples\db2xml\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> an
    <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 14. Knotendefinitionen für das XML-Dokument, wie sie der XML-Objektgruppentabelle zugeordnet werden

In diesem Beispiel wurden den ersten beiden Spalten Elemente und Attribute zugeordnet.

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>
```

Verwenden Sie einen beliebigen Texteditor, um eine DAD-Datei zu erstellen und zu aktualisieren.

Zugehörige Konzepte:

- „Schemata für die Zuordnung von XML-Objektgruppen" auf Seite 110

SQL-Zusammensetzung

Sie können XML-Dokumente unter Verwendung von Spalten mit demselben Namen zusammensetzen. Ausgewählte Spalten mit demselben Namen müssen, auch wenn sie aus verschiedenen Tabellen stammen, durch einen eindeutigen Aliasnamen identifiziert werden, so dass jede Variable in der Select-Klausel der SQL-Anweisung unterschiedlich ist. Das folgende Beispiel zeigt die Vergabe von eindeutigen Aliasnamen für Spalten, die denselben Namen haben.

```
<SQL_stmt>select o.order_key as oorder_key,
                 key customer_name, customer_email,
                 p.part_key p.order_key as porder_key,
                 color, qty, price, tax, ship_id, date, mode
            from order_tab o.part_tab p
            order by order_key, part_key</SQL_stmt>
```

Sie können außerdem XML-Dokumente unter Verwendung von Spalten mit generierten Zufallswerten zusammensetzen. Wenn eine SQL-Anweisung in einer DAD-Datei einen Zufallswert hat, müssen Sie der Funktion für Zufallswerte einen Aliasnamen geben, um sie in der Klausel ORDER BY verwenden zu können. Dies ist erforderlich, da der Wert keiner Spalte in einer angegebenen Tabelle zugeordnet ist. Beachten Sie den Aliasnamen für 'generate_unique' am Ende der Klausel ORDER BY im folgenden Beispiel.

```
<SQL_stmt>select o.order_key, customer_name,customer_email,
                 p.part_key,color,qty,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>2000 and
                o.order_key=o.order_key and s.part_key
          order by order_key, part_key,ship_id</SQL_stmt>
```

RDB_node-Zusammensetzung

Die folgenden Einschränkungen gelten für die RDB_node-Zusammensetzung:

- Die Bedingung, die einer Nicht-root_node-RDB_node-DAD-Datei zugeordnet ist, muss einen Vergleich mit einem Literal enthalten.
- Jede Gleichheit in der Bedingung, die einem RDB_node auf höchster Ebene zugeordnet ist, gibt eine Verknüpfungsbeziehung zwischen Spalten von zwei Tabellen an und gilt separat von anderen Gleichheiten. Dies bedeutet, dass alle Prädikate, die durch AND verbunden sind, nicht gleichzeitig für eine einzelne Verknüpfungsbedingung gelten. Sie simulieren daher bei der Dokumentzusammensetzung eine äußere Verknüpfung (Outer Join). Die Beziehung der Unter- und Überordnung zwischen den einzelnen Tabellenpaaren ist durch die relative Verschachtelung in der DAD-Datei festgelegt. Beispiel:

```
<condition>order_tab.order_key=part_tab.order_key AND
part_tab.part_key=ship_tab.part_key</condition>
```

Erstellung von Zeilen mit Nullwerten

Zum Zusammensetzen von XML-Dokumenten können Sie Spalten mit Nullwerten verwenden.

Das folgende Beispiel zeigt, wie ein XML-Dokument von einer Tabelle *MyTable* aus generiert werden kann, in der in Spalte *Col 1* eine Zeile mit Nullwert enthalten ist. Die im Beispiel verwendete DAD lautet *nullcol.dad*.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO validation>NO>
<Xcollection>
<SQL_stmt>SELECT 1 as X, Col1 FROM MyTable order by X, Col1<\SQL_stmt>
<prolog>?xml version="1.0"?prolog>?xml version="1.0"?>
<doctype>!DOCTYPE Order SYSTEM "e:\t3xml\x.dtd">
<root_node>
<element_node name="MyColumn">
<element_node name="Column1" multi_occurrence="YES">
  <text_node>
  <column name="Col1"/>
  </text_node>
 </element_node>
 </element_node>
</root_node>
</Xcollection>
</DAD>
```

```
MyTable
_____________
| Col 1 |
|_____________|
| 1 |
|_____________|
| 3 |
|_____________|

| - |
|_____________|
```

Führen Sie tests2x mydb nullcol.dad result_tab aus, oder verwenden Sie 'dxx-GenXML', um das folgende Dokument zu erstellen. Beachten Sie, dass das dritte Element 'Column1' einen Nullwert darstellt.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "e:\t3xml\x.dtd">
<MyColumn>
 <Column1>1</Column1>
 <Column1>3</Column1>
 <Column1></Column1>
</MyColumn>
```

- Die Bedingung, die einer Nicht-root_node-RDB_node-DAD-Datei zugeordnet ist, muss einen Vergleich mit einem Literal enthalten.
- Die Bedingung, die einem beliebigen RDB-Knoten auf niegrigerer Stufe zugeordnet ist, muss einen Vergleich mit einem Literal enthalten.
- Die Bedingung, die einem root_node zugeordnet ist, beschreibt die Beziehung zwischen den Tabellen, die bei der RDB_node-Zusammensetzung beteiligt sind. Ein Beispiel ist eine Beziehung zwischen einem Primär- und einem Fremdschlüssel.
- Jede Gleichheit in der Bedingung, die einem RDB_node auf höchster Ebene zugeordnet ist, gibt eine Verknüpfungsbeziehung zwischen Spalten von zwei Tabellen an und gilt separat von anderen Gleichheiten. Dies bedeutet, dass alle Prädikate, die durch AND verbunden sind, nicht gleichzeitig für eine einzelne Verknüpfungsbedingung gelten. Sie simulieren daher bei der Dokumentzusammensetzung eine äußere Verknüpfung (Outer Join). Die Beziehung der Unter- und Überordnung zwischen den einzelnen Tabellenpaaren ist durch die relative Verschachtelung in der DAD-Datei festgelegt. Beispiel:

```
<condition>order_tab.order_key=part_tab.order_key AND
part_tab.part_key=ship_tab.part_key</condition>
```

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. Das Name der DTD-Datei lautet `dad.dtd`. Das folgende Beispiel zeigt die DTD für die DAD-Datei.

```
<?xml encoding="US-ASCII"?>

  <!ELEMENT DAD ((schemabindings | dtdid)?, validation,
  (Xcolumn | Xcollection))>
  <!ELEMENT dtdid (#PCDATA)>
  <!ELEMENT schemabindings (nonamespacelocation)>
  <!ELEMENT nonamespacelocation (empty)>
  <!ATTLIST nonamespacelocation location CDATA #REQUIRED>
  <!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?, prolog, doctype, root_node)>
  <!ELEMENT SQL_stmt (#PCDATA)>
  <!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 process_instruction_node (#PCDATA)>
```

Die DAD-Datei umfasst vier wichtige Elemente:

- DTDID
- validation
- 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 dem vorherigen Beispiel.

DTDID-Element

DTDs, die XML Extender zur Verfügung stehen, werden in der Tabelle DTD_REF gespeichert. Jede DTD wird durch eine eindeutige ID identifiziert, die im Befehl `DTDID` der DAD-Datei angegeben wird. Die `DTDID` zeigt auf die DTD, die die Gültigkeit der XML-Dokumente prüft, oder die Zuordnung zwischen XML-Objektgruppentabellen und XML-Dokumenten steuert. Für XML-Objektgruppen ist dieses Element nur für die Überprüfung der Ein- und Ausgabe-XML-Dokumente erforderlich. Für XML-Spalten ist dieses Element nur für die Überprüfung der Eingabe-XML-Dokumente erforderlich. Die `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 verbindliches und zwei implizierte Attribute:

name-Attribut
Gibt den Namen der Nebentabelle an.

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.

Das Element `table` hat ein untergeordnetes Element:

column-Element
Ordnet ein Attribut eines CDATA-Knoten aus dem Eingabe-XML-Dokument einer Spalte in der Tabelle zu.

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, und muss der einfache Standortpfad sein, der in Tabelle 3.1.a angegeben wurde.

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 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.

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 wird verwendet, um den DOCTYPE des resultierenden Dokuments anzugeben. Definieren Sie einen expliziten Wert. Dieser Wert wird für alle Dokumente in der gesamten Objektgruppe verwendet.

doctype hat ein untergeordnetes Element:

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)>`

RDB_node
Definiert die DB2 UDB-Tabelle, in der der Inhalt eines XML-Elements oder der Wert eines XML-Attributs gespeichert werden soll, oder aus der er abgerufen wird. rdb_node ist ein untergeordnetes Element von element_node, text_node und attribute_node und hat die folgenden untergeordneten Elemente:

table Gibt die Tabelle an, in der der Element- oder Attributinhalt gespeichert wird.

column
Gibt die Spalte an, in der der Element- oder Attributinhalt gespeichert wird.

condition
Gibt eine Bedingung für die Spalte an. Wahlfrei.

element_node
(Elementknoten) Steht für ein XML-Element. Es muss in der für die Objektgruppe angegebenen DAD 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 Stammelement 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 text_node angegeben ist. Die Bedingung ist wahlfrei, je nach der DTD und der Abfragebedingung.

untergeordnete Knoten
Optional: Ein element_node kann auch die folgenden untergeordneten Knoten haben:

element_node
Stellt untergeordnete Elemente des aktuellen XML-Elements dar.

attribute_node
Stellt Attribute des aktuellen XML-Elements dar.

text_node
Stellt den CDATA-Text des aktuellen XML-Elements dar.

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.

Zugehörige Konzepte:

• „DAD-Dateien für XML-Objektgruppen" auf Seite 178

Zugehörige Tasks:

• „Werte in der DAD-Datei dynamisch überschreiben" auf Seite 187

Werte in der DAD-Datei dynamisch überschreiben

Vorgehensweise:

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. Der Parameter *overrideType* hat die folgenden Werte:

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.

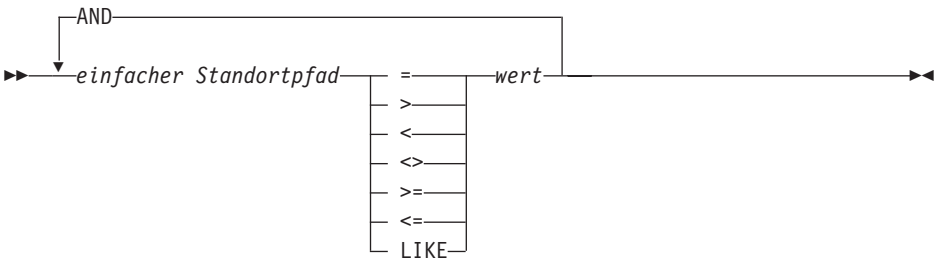
XML_OVERRIDE
Gibt an, dass eine Bedingung in der DAD-Datei durch eine XPath-Bedingung überschrieben werden soll.

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 den Parameter *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.

  Wenn Sie SQL_OVERRIDE als 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 angeben, ist der Eingabewert eine Zeichenfolge, die ein oder mehrere Ausdrücke enthält.

  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
Gibt einen einfachen Standortpfad an, wobei die von XPath definierte Syntax verwendet wird.

Operatoren
Die SQL-Operatoren, die im Syntaxdiagramm gezeigt werden, können Leerzeichen enthalten, um den Operator von den anderen Teilen des Ausdrucks zu trennen.

Leerzeichen um die Operatoren sind optional. Um den LIKE-Operator sind Leerzeichen obligatorisch.

wert
Ein numerischer Wert oder eine Zeichenfolge, in einfache Anführungszeichen eingeschlossen.

AND
AND wird als logischer Operator in demselben Standortpfad behandelt. Wenn in der Überschreibungszeichenfolge ein einfacher Standortpfad mehrmals angegeben ist, werden alle Prädikate für diesen einfachen Standortpfad gleichzeitig angewendet.

Wenn Sie XML_OVERRIDE angeben, 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.

Die folgenden Beispiele verwenden SQL_OVERRIDE und XML_OVERRIDE, um das dynamische Überschreiben zu demonstrieren.

Beispiel 1: Eine gespeicherte Prozedur, die SQL_OVERRIDE verwendet. 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 durch Ändern des Datums auf einen Wert größer als 1998-12-01.

```
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;        /* Rückkehrfehlercode */
         char    returnMsg[1024];   /* Fehlernachrichtentext */
         short   rtab_ind;
         short   collection_ind;

         short   ovtype_ind;
         short   ov_ind;
         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");
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)  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 db2xml.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);
```

Beispiel 2: Eine gespeicherte Prozedur, die XML_OVERRIDE verwendet. 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. XML Extender wandelt den einfachen Standortpfad in die zugeordnete DB2 UDB-Spalte um.

```
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, XPATH-Bedingung */
        short   overrideType;    /* definiert in dxx.h */
        short   max_row;         /* maximale Anzahl Zeilen */
        short   num_row;           /* tatsächliche Anzahl Zeilen */
        long    returnCode;        /* Rückkehrfehlercode */
        char    returnMsg[1024];   /* Fehlernachrichtentext */
        short   dadbuf_ind;
        short   rtab_ind;
        short   collection_ind;
        short   ovtype_ind;
        short   ov_ind;
        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");
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);
```

Mehrfache Überschreibungsvorgänge

XML Extender unterstützt mehrfache Überschreibungsvorgänge für denselben Pfad. Alle für den RDB-Knoten angegebenen Überschreibungsvorgänge werden akzeptiert.

Sie können mehrere XML-Überschreibungsvorgänge im selben Standortpfad angeben, um definierte Bedingungen für die Suche einzugrenzen. Im folgenden Beispiel wird unter Verwendung der Datei test.dad ein XML-Dokument aus zwei Tabellen zusammengesetzt.

Tabelle 45. Abteilungstabelle

Abteilungsnummer	Abteilungsname
10	Konstruktion
20	Controlling
30	Marketing

Tabelle 46. Mitarbeitertabelle

Mitarbeiternummer	Abteilungsnummer	Gehalt
123	10	EURO 98.000
456	10	EURO 87.000
111	20	EURO 65.000
222	20	EURO 71.000
333	20	EURO 66.000
500	30	EURO 55.000

Die unten dargestellte DAD-Datei test.dad enthält eine Bedingung, die die Variable *deptno* mit dem Wert 10 vergleicht. Sie müssen diese Bedingung überschreiben, wenn die Suche erweitert werden und zwischen den Werten >10 und <30 ausgeführt werden soll. Sie müssen den zu überschreibenden Parameter wie folgt definieren, wenn Sie dXXGenXML aufrufen:

```
/ABC.com/Department>10 AND /ABC.com/Department<30

<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "C:\dxx_xml\test\dtd\dad.dtd">
<DAD>
 <dtdid>E:\dtd\lineItem.dtd</dtdid>
<validation>NO</validation>     <Xcollection>
<porlog>?xmol version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd"</doctype>
   <root_node>
<element_node name="ABC.com">
<TDB_node>
<table name="dept" key="deptno"/>
<table name="empl" key="emplno"/>
<condition>dept deptno=empl.deptno</condition>
          </RDB_node>


<element_node name="Department" multi_occurrence="YES">
     <text_node>

       <RDB_node>
<table name="dept"/>
<column name="deptno">
<condition>deptno=10</condition><RDB_node></RDB_node><text_node></text_node>
<element_node name="Employees" multi_occurrence="YES">

     <text_node>

       <RDB_node>

<table name="dept"><column name="deptnot"><condition>deptno=10</condition>
</table></RDB_node></text_node>
<element_node name="Employees" multi_occurence="YES">

<element_node name="EmployeeNo">

     <text_node>

       <RDB_node>

<table name="empl"><column name="emplno"><condition>emplno<500</condition>
</table></RDB_node></text_node></element_node>
<element_node name="Salary">

     <text_node>
       <RDB_node>

<table name="empl"><column name="salary"><condition>salary>5000.00</condition>
</table></RDB_node></text_node></element_node></element_node></element_node>
```

Um ein XML-Dokument ohne Überschreibungsvorgang zusammenzusetzen, geben Sie tests2x mydb test.dad result_tab ein. Sie können aber auch dxxGenXML aufrufen, ohne einen Überschreibungsvorgang zu definieren. Dadurch wird ein ähnliches Dokument wie das Folgende generiert:

```
<?xml version="1.0">
<!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd">
<ABC.com>
<Department>10
```

```
<Employees>
<EmployeeNo>123</EmployeeNO>
<Salary>98,000.00</Salary>
</Employees>
<Employees>
<EmployeeNo>456</EmployeeNo>
<Salary>87,000.00</Salary>
</Employees>
</Department>
</ABC.COM>
```

Zum Überschreiben der DAD-Datei können Sie, wie bereits oben erwähnt, dxxGenXML aufrufen oder das Programm test2x mit den angegebenen Bedingungen ausführen:

tests2x mydb test.dad result_tab -o 2 ″/ABC.com/Department>10 AND /ABC.com/Department<30″

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

<!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd">
<ABC.com>
<Department>20
<Employees>
<EmployeeNo>111</EmployeeNo>
<Salary>65,000.00</Salary>
</Employees>
<EmployeeNo>222</EmployeeNo>
<Salary>71,000.00</Salary>
</Employees>
<Employees>
<EmployeeNo>333</EmployeeNo>
<Salary>66,000.00</Salary>
</Employees>
</Department>
</ABC.com>
```

Zugehörige Konzepte:

- „DAD-Dateien für XML-Objektgruppen” auf Seite 178
- „DAD-Prüfprogramm” auf Seite 192

Zugehörige Tasks:

- „DAD-Datei für XML-Spalten erstellen” auf Seite 175
- „DAD-Prüfprogramm verwenden” auf Seite 193

Zugehörige Referenzen:

- „DTD für die DAD-Datei” auf Seite 182

DAD-Prüfprogramm

Das DAD-Prüfprogramm kann verwendet werden, um die Gültigkeit von DAD-Dateien zu prüfen, die als Speichermethode die XML-Objektgruppe verwenden. In jeder Datei wird ein Zuordnungsschema angegeben, das die Beziehung zwischen den Tabellen und der Struktur des XML-Dokuments angibt.

Ähnlich wie DTDs (Document Type Description = Dokumentartbeschreibung) zur Prüfung der Syntax von XML-Dokumenten verwendet werden, wird das DAD-Prüfprogramm verwendet, um sicherzustellen, dass eine DAD-Datei semantisch korrekt ist. Diese Prüfung kann ohne Verbindung zu einer Datenbank stattfinden.

Die Verwendung des DAD-Prüfprogramms kann beim Minimieren von Fehlern helfen, die beim Übergeben der Datei zur Verarbeitung an XML Extender auftreten. Das DAD-Prüfprogramm ist eine Java-Anwendung, die von der Befehlszeile aus aufgerufen wird. Nach dem Aufruf erstellt die Anwendung zwei Ausgabedateien, die Fehler, Warnungen und Erfolgsanzeiger enthalten. Die beiden Dateien sind äquivalent. Bei der einen handelt es sich um eine Textdatei, die Sie zur Prüfung auf Fehler oder Warnungen verwenden, die andere ist eine XML-Datei, `errorsOutput.xml`, die die Ergebnisse des DAD-Prüfprogramms an andere Anwendungen meldet. Der Name der Ausgabetextdatei ist benutzerdefiniert. Wenn kein Name angegeben ist, wird die Standardausgabe verwendet.

Zugehörige Konzepte:

- „DAD-Dateien für XML-Objektgruppen" auf Seite 178

Zugehörige Tasks:

- „Werte in der DAD-Datei dynamisch überschreiben" auf Seite 187
- „DAD-Datei für XML-Spalten erstellen" auf Seite 175
- „DAD-Prüfprogramm verwenden" auf Seite 193

DAD-Prüfprogramm verwenden

Voraussetzungen:

Sie müssen JRE oder SDK Version 1.3.1 oder höher auf Ihrem System installiert haben.

Vorgehensweise:

So verwenden Sie das DAD-Prüfprogramm:

1. Laden Sie die Datei DADChecker.zip herunter, und extrahieren Sie alle Dateien in ein Verzeichnis Ihrer Wahl.
2. Wechseln Sie von einer Befehlszeile aus in das Unterverzeichnis `/bin` des Verzeichnisses, in dem Sie das DAD-Prüfprogramm installiert haben.
3. Setzen Sie den Klassenpfad, indem Sie die Datei setCP.bat ausführen, die sich im Verzeichnis `/bin` befindet.
4. Führen Sie den folgenden Befehl aus:

   ```
   java dadchecker.Check_dad_xml [-dad | -xml] [-all][-tag befehlsname]
   [-out ausgabedatei] zu_prüfende_datei
   ```

   Dabei gilt Folgendes:

   -dad
   gibt an, dass die zu prüfende Datei eine DAD-Datei ist. Dies ist die Standardoption.

   -xml
   gibt an, dass die zu prüfende Datei statt einer DAD-Datei ein XML-Dokument ist. Bei großen XML-Dokumenten reicht möglicherweise der Speicher der Java Virtual Machine nicht aus, wodurch die Ausnahme java.lang.OutOfMemoryError erzeugt wird. In solchen Fällen kann die Option -Xmx verwendet werden, um der Java Virtual Machine mehr Speicher zuzuordnen. Weitere Informationen finden Sie in der Dokumentation zu SDK.

-all
gibt an, dass in der Ausgabe alle fehlerhaften Vorkommen von Befehlen angezeigt werden.

-tag
gibt an, dass nur die doppelten Befehle angezeigt werden, deren Namensattributwerte *befehlsname* sind. Bei XML-Dokumenten werden nur die doppelten Befehlsnamen angezeigt, deren Namen mit dem hier angegebenen Befehlsnamen übereinstimmen.

-out
ausgabedatei gibt den Namen der Ausgabetextdatei an. Wird keine Datei angegeben, wird die Standardausgabe verwendet. Außerdem wird eine zweite Ausgabedatei, `errorsOutput.xml`, im selben Verzeichnis wie die DAD-Datei erstellt. Diese Datei wird immer generiert und enthält im XML-Format dieselben Informationen wie die Ausgabetextdatei, mit Ausnahme der Parser-Warnungen und -Fehler.

Um Befehlszeilenoptionen anzuzeigen, geben Sie `java dadchecker.Check_dad_xml help` ein.

Um die Versionsnummer anzuzeigen, geben Sie `java dadchecker.Check_dad_xml version` ein.

Beispieldateien für Dad-Prüfprogramm:

Die folgenden Beispieldateien finden Sie im Verzeichnis `samples`:

bad_dad.dad
Beispiel-DAD-Datei, die alle möglichen semantischen Fehler demonstriert.

bad_dad.chk
Ausgabetextdatei, die vom DAD-Prüfprogramm für bad_dad.dad generiert wurde.

bad_dad.chk
Ausgabetextdatei, die vom DAD-Prüfprogramm für bad_dad.dad generiert wurde.

errorsOutput.xml
Ausgabe-XML-Datei, die vom DAD-Prüfprogramm für bad_dad.dad generiert wurde.

dup.xsl
XSL-Formatvorlage, die für die Umsetzung der Datei errorsOutput.xml in eine HTML-Datei, die nur die doppelten Befehle zeigt, verwendet wird.

dups.html
generierte HTML-Datei, die nur die doppelten Befehle anzeigt, die in bad_dad.dad enthalten sind.

Fehler und Warnungen in der Ausgabetextdatei:

Fehler und Warnungen werden nach Befehlsvorkommen angezeigt. Zwei Befehle werden als Vorkommen desselben Befehls angesehen, wenn Folgendes zutrifft:

- Ihre Namensattribute haben denselben Wert.
- Sie haben die gleiche Anzahl an übergeordneten Befehlen.
- Die Namensattribute ihrer entsprechenden übergeordneten Befehle haben denselben Wert.

Vorkommen des gleichen Befehls können unterschiedliche untergeordnete Befehle haben.

Befehlsvorkommen, die nicht den semantischen Regeln der DAD entsprechen, werden wie folgt in der Ausgabetextdatei gekennzeichnet:

- Alle übergeordneten Befehle und deren Attribute werden nacheinander angezeigt.
- Der fehlerhafte Befehl wird angezeigt, wobei ihm eine Zahl voransteht, die die Tiefe in der XML-Baumstruktur angibt. Auf den Befehlsnamen folgt eine Liste von Zeilennummern, in denen der Befehl in der DAD-Datei erscheint. Sie können jedes Fehlervorkommen separat aufrufen, indem Sie die Befehlszeilenoption *-all* verwenden.
- Die direkten untergeordneten Befehle des ersten Befehlsvorkommens werden angezeigt. Bei den untergeordneten Befehlen, die eine Datenzuordnung angeben, werden die Datenzuordnungsbefehle ebenfalls angezeigt. Sie können die Befehlszeilenoption *-all* verwenden, um jedes Fehlervorkommen separat aufzurufen.

Beispiel eines Fehlerberichts für DAD-Prüfprogramm:

In diesem Beispiel ist der Befehl element_node, dessen Namensattribut den Wert "Password" hat, fehlerhaft. Dieser Befehl kommt zweimal in der DAD-Datei vor, in Zeile 49 und in Zeile 75. Der fehlerhafte Befehl kann aus der Liste der über- und untergeordneten Befehle isoliert werden, indem die Tiefenkennzeichnung des Befehls (in diesem Beispiel 4) lokalisiert wird. Die Liste der über- und untergeordneten Befehle kann beim Ermitteln des Kontextes, in dem der Fehler auftrat, hilfreich sein.

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4       <element_node name="Password">  line(s): 49 75
        <element_node name="Pswd1">
     <element_node name="Pswd2">
```

Wenn Sie die Option 'all' verwendet hätten, sähe die Ausgabetextdatei z. B. aus wie folgt:

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4         <element_node name="Password">  line: 49
        <element_node name="Pswd1">
        <element_node name="Pswd2">


<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4      <element_node name="Password">  line: 75
        <element_node name="Pswd1">
    <element_node name="Pswd3">
```

In diesem Beispiel haben die zwei Vorkommen identische übergeordnete Elemente und Namensattributwerte, aber unterschiedliche untergeordnete Elemente.

Durch das DAD-Prüfprogramm ausgeführte Prüfungen

Wenn Sie das DAD-Prüfprogramm aufrufen, empfangen Sie die folgende Nachricht:

```
DAD-Dokument wird überprüft: dateipfad
```

Dabei ist *dateipfad* der Pfad der DAD-Datei, die überprüft wird.

Das DAD-Prüfprogramm führt die folgenden Gültigkeitsprüfungen durch:

1. Prüfung auf korrekte Formatierung und DTD-Prüfung.
2. Ermittlung doppelter <attribute_node>-Elemente und <element_node>-Blattelemente (RDB_node-Zuordnung).
3. Ermittlung fehlender Typattribute.
4. Ermittlung fehlender Tabellendeklaration.
5. Ermittlung fehlender <text_node>- oder <attribute_node>-Elemente.
6. Prüfung der Zuordnungsreihenfolge von <attribute_node> und <element_node>.
7. Konsistenzprüfung der Datenzuordnung für Befehle mit identischen Namensattributwerten.
8. Prüfung von mehrfach vorkommenden Attributwerten für übergeordnete <element_node>-Elemente mit zugeordneten untergeordneten Elementen (RDB_node-Zuordnung).
9. Prüfung auf mögliche Namensunverträglichkeiten von Attributen und Elementen (XML-Dokumente).

Diese Gültigkeitsprüfungen werden in den folgenden Abschnitten beschrieben.

Korrekte Formatierung und DTD-Prüfung

DAD-Dateien müssen anhand der DAD-DTD überprüft werden, die sich in "c:*dxx_install_verz*\samples\db2xml\dtd\dad.dtd" befindet. Wenn die DAD-Datei nicht korrekt formatiert ist oder wenn die DTD nicht gefunden werden kann, tritt ein schwer wiegender Fehler auf, der zum Beenden des DAD-Prüfprogramms führt und der in der Ausgabetextdatei angegeben ist. Beispiel:

```
org.xml.sax.SAXException: Stopping after fatal error,
line 1, col 22. The XML declaration must end with "?>".
```

Fehler und Warnungen bei der Prüfung werden auch in der Ausgabetextdatei berichtet, führen aber nicht zum Beenden des DAD-Prüfprogramms. Das folgende Beispiel ist ein Fragment einer Ausgabetextdatei, die zwei mögliche Prüfungsfehler zeigt, die beim Parsen der DAD-Datei auftreten können:

```
 ** The document is not valid against the DTD, line 5, col 15. Element type
     "XCollection" must be declared

** The document is not valid against the DTD, line 578, col 21. The content of
      element type "text_node" must match "(column|RDB_node)".
```

Ermittlung doppelter <attribute_node>-Elemente und <element_node>-Blattelemente (RDB_node-Zuordnung)

Diese Überprüfung ist nur für DAD-Dateien relevant, die die RDB_node-Zuordnung verwenden.

Zwei Elemente werden als gleich angesehen, wenn zwei oder mehr <attribute_node>- oder <element_node>-Befehle den gleichen Wert in ihrem Namensattribut haben und dieselben übergeordneten Elemente haben.

Zwei oder mehr Befehle haben dieselben übergeordneten Elemente, wenn die Namensattribute ihrer entsprechenden übergeordneten Befehle denselben Wert haben.

Ein <element_node>-Blattbefehl ist ein element_node, der für die Zuordnung eines Befehls verwendet wird, der in der XML-Dokumentbaumstruktur keine untergeordneten Elemente hat. Daher müssen <element_node>-Blattbefehle einen Textknotenbefehl als eines ihrer direkten untergeordneten Elemente haben. Kein anderer <element_node>-Befehl kann Textknotenbefehle als direkte untergeordnete Befehle haben.

Dieser Konflikt kann zwischen zwei oder mehr <element_node>-Blattbefehlen, zwischen zwei oder mehr <attribute_node>-Befehlen oder zwischen <element_node>-Blattbefehlen und <attribute_node>-Befehlen auftreten.

Beispiele:

Beispiel 1:

Konflikt beim <element_node>-Blattelement:

```
<element_node name = "A1">
    <element_node name = "B">
       <element_node name = "C">
       <text_node
          ....
<element_node name = "A2">
    <element_node name = "B">
       <element_node name = "C">
          <text_node>
       ....
</element_node>
```

In diesem Beispiel ist <element_node name = ″C″> doppelt, da es über zwei unterschiedliche Pfade zugeordnet ist: \A1\B\C und \A2\B\C. Beachten Sie, dass <element_node name=″B″> nicht als doppelt angesehen wird, da es kein <element_node>-Blattelement ist.

Beispiel 2:

Dieses Beispiel zeigt einen <attribute node>-Konflikt.

```
<element_node name = "A1">
<attribute_node name = "B">
                   ....
<element_node name = "A2">
<attribute_node name = "B">
 /element_node>               ....
<
```

In diesem Beispiel ist <attribute_node name = ″B″> doppelt, da es über zwei unterschiedliche Pfade zugeordnet ist: \A1\B und \A2\B.

Beispiel 3:

Dieses Beispiel zeigt einen Konflikt beim <element_node>-Blattelement und <attribute_node>.

```
<element_node name = "A">
    <element_node name = "B">
          <text_node>
            ....
```

```
       </element_node>
</element_node>
           ....
<attribute_node name = "B">
           ....
<attribute_node name = "A">
           ....
```

In diesem Beispiel ist <element_node name = ″B″> im Konflikt mit <attribute_node name = ″B″>. Beachten Sie, dass <element_node name = ″A″> und <attribute_node name = ″A″> nicht in Konflikt stehen, da <element_node name = ″A″> kein <element_node>-Blattelement ist.

Wenn Konflikte auftreten, muss, um sie zu beseitigen, die DTD für das XML-Dokument überarbeitet werden. Das XML-Dokument und die DAD-Datei müssen ebenfalls überarbeitet werden, um die DTD-Änderungen wiederzugeben.

Beispiel 4:

```
7 Unverträglichkeit durch doppelte Namen gefunden
Insgesamt 16 Befehle fehlerhaft (gesamtes Vorkommen dieser Befehle: 20)

Die folgenden Befehle sind doppelt:

<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4       <element_node name="Country">  line(s): 127 135
          <text_node>
           <RDB_node>
             <table name="advertiser">
               <column type="VARCHAR(63)" name="country">

------------
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
     <element_node name="Campaign" multi_occurrence="YES">
      <element_node name="Target" multi_occurrence="YES">
       <element_node name="Location" multi_occurrence="YES">
7         <element_node name="Country">  line(s): 460
            <text_node>
                      <RDB_node>
               <table name="target_location">
               <column type="VARCHAR(63)" name="country">
-----------------------------------------------------------
```

Fehlerhafte Befehle sind nach Namenskonflikt gruppiert. Die Gruppen sind durch Linien voneinander getrennt, und die Befehle sind durch kurze Linien getrennt. Sie können außerdem alle Fehlervorkommen anzeigen, indem Sie die Befehlszeilenoption *all* verwenden.

Wenn keine doppelten Werte in der DAD-Datei vorhanden sind, wird die folgende Nachricht in die Ausgabetextdatei geschrieben:

```
Keine doppelten Befehle gefunden.
```

Ermittlung fehlender Typattribute

Wenn eine DAD-Datei zum Aktivieren einer Objektgruppe oder für die Zerlegung verwendet wird, muss das Typattribut für jeden <column>-Befehl angegeben werden.

Beispiel:

```
<column name="email" type="varchar(20)">
```

Der Befehl enable_collection verwendet die Spaltentypspezifikation, um die Tabellen in der Objektgruppe zu erstellen, wenn sie nicht vorhanden sind. Wenn die Tabellen vorhanden sind, muss der in der DAD angegebene Typ mit dem tatsächlichen Spaltentyp in der Datenbank übereinstimmen.

Beispiel:

Das folgende Beispiel ist ein Fragment einer Ausgabetextdatei, die <column>-Befehle zeigt, die kein Typattribut haben:

```
Wenn diese DAD für die Zerlegung oder für die Aktivierung einer Objektgruppe
verwendet werden soll, fehlen die Typattribute für die
folgenden <column>-Befehle:
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
      <element_node name="Address">
          <text_node>
           <RDB_node>
7           <column name="address">  line: 86
```

Wenn keine Typattribute fehlen, wird die folgende Nachricht in die Ausgabetextdatei geschrieben:

```
Es fehlen keine Typattribute für <column>-Befehle.
```

Ermittlung fehlender Tabellendeklaration

Der erste <RDB_node>-Befehl in der DAD-Datei muss die Tabellendeklaration umfassen, einschließlich aller <table>-Befehle, die die für die Datenzuordnung verwendeten relationalen Tabellen deklarieren. Dieser Befehl muss im ersten <element_node>-Befehl eingeschlossen sein. Alle nachfolgenden <RDB_node>-Befehle müssen in einen <text_node>-Befehl eingeschlossen sein.

Ein Fehler wird außerdem zur Ausgabedatei hinzugefügt, wenn der erste auftretende <RDB_node>-Befehl einen <column>-Befehl enthält. Dieser Fehler gibt an, dass entweder die Tabellendeklaration fehlt oder die Tabellendeklaration fälschlicherweise einen <column>-Befehl enthält.

Ermittlung fehlender <text_node>- oder <attribute_node>-Elemente

Jeder <RDB_node>-Befehl, der nicht der erste, der für die Tabellendeklaration verwendet wird, ist, muss in einen <attribute_node>-Befehl oder einen <text_node>-Befehl eingeschlossen sein.

Beispiele:

Beispiel 1:

```
 <element_node name ="amount">
          <text_node>
           <RDB_node>
            <table name="fakebank.payments"/>
           <column name="amount" type="decimal(8,2)"/>
          </RDB_node>
</element_node>
```

Beispiel 2:

Das folgende Beispiel ist ein Fragment einer Ausgabetextdatei, die einen fehlenden <text_node>- oder <attribute_node>-Befehl zeigt:

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
      <element_node name="PostalCode">
5         <RDB_node>  line: 107
             <table name="advertiser">
        <column type="VARCHAR(10)" name="postal_code">
```

Prüfung der Zuordnungsreihenfolge von <attribute_node> und <element_node>

Diese Prüfung ist für FixPak 3 und früher erforderlich. Die <attribute_node>-Befehle müssen einer Tabelle zugeordnet sein, bevor <element_node>-Befehle der Tabelle zugeordnet werden.

Beispiel:

Das folgende Beispiel zeigt Befehle, die einer Tabelle zugeordnet werden müssen.

```
<element_node name="payment-request"
multi_occurrence="YES">
  <element_node name="payment-request-id">
          <text_node>
           <RDB_node>
            <table name="fakebank.payments"/>
            <column name="statement_id" type="varchar(30)"/>
                  ....
<element_node name="bank-customer-info">
    <element_node name="account">
          <attribute_node name="type">
            <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
                 <column name="payor_account" type="char(6)"/
```

In diesem Beispiel wird <attribute_node name=″type″> derselben Tabelle (fakebank.payments) zugeordnet wie <element_node name = ″payment-request-id″>. Die Zuordnung von <attribute_node> muss vor der Zuordnung von <element_node> erfolgen.

Konsistenzprüfung der Datenzuordnung für Befehle mit identischen Namensattributwerten

Innerhalb der DAD-Datei sollten alle <element_node>-Befehle und alle <attribute_node>-Befehle, die zugeordnet und durch eindeutige Namensattributwerte identifiziert werden, nur einmal zugeordnet werden. Wenn zwei oder mehr Vorkommen eines <element_node>- oder <attribute_node>-Befehls unterschiedlichen Spalten zugeordnet werden, sollten ihren Namensattributen unterschiedliche Werte zugeordnet werden.

Beispiel:

Beispiel 1: In diesem Beispiel hat das zweite Vorkommen des <element_node name=″type″>-Befehls eine andere Zuordnung als das erste Vorkommen. Doppelte <attribute_node>-Befehle und doppelte <element_node>-Blattbefehle werden als Ergebnis dieser Prüfung nicht angezeigt.

```
<element_node name="bank-customer-info">
    <element_node name="account">
      <element_node name="type">
          <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
              <column name="payor_account" type="char(20)
          </RDB_node>
         </text_node>
       </element_node>
      <element_node>
<element_node>
<element_node name="bank-customer-info">
    <element_node name="account">
      <element_node name="type">
          <text_node>
           <RDB_node>
            <table name="fakebank.payments"/>
              <column name="payto_account" type="char(20)"/>
          </RDB_node>
        </text_node>
       </element_node>
</element_node>
<element_node>
```

Sie können diesen Fehler korrigieren, indem Sie ein neues Element für die Verwendung mit der zweiten Zuordnung erstellen. Sie müssen außerdem die DTD, das XML-Dokument und die DAD-Datei ändern.

Beispiel 2:: Dieses Beispiel ist ein Fragment einer Ausgabetextdatei, die <element_node>-Befehle angibt, die dieselben Namen und übergeordneten Befehle haben, aber nicht dieselbe Zuordnung.

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4      <element_node name="PostalCode">  line(s): 127
          <text_node>
           <RDB_node>
             <table name="advertiser">
        <column type="VARCHAR(10)" name="postal_code">
----------------
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4         <element_node name="PostalCode">  line(s): 135 143
          <text_node>
           <RDB_node>
             <table name="advertiser">
              <column type="VARCHAR(10)" name="postal_code2">
```

In diesem Beispiel wird ein Vorkommen von <element_node name=″PostalCode″> in Zeile 127 der Spalte 'postal_code' zugeordnet, und zwei weitere Vorkommen des gleichen Befehls in den Zeilen 135 und 143 werden der Spalte 'postal_code2' zugeordnet.

Prüfung von mehrfach vorkommenden Attributwerten für übergeordnete <element_node>-Elemente mit zugeordneten untergeordneten Elementen

Diese Prüfung ist nur für DAD-Dateien relevant, die die RDB_node-Zuordnung verwenden.

Der Standardwert für das Attribut 'multi_occurrence' ist NO. Dem Attribut 'multi-_occurrence' muss der Wert YES für jeden <element_node>-Befehl zugeordnet werden, der als direkte untergeordnete Befehle einen <attribute_node>-Befehl hat, oder wenn bei zwei oder mehr <element_node>-Befehlen ein oder zwei der folgenden Kriterien zutreffen:

- Der <element_node>-Befehl ist zugeordnet (er hat einen <text_node>-Befehl als direkten untergeordneten Befehl).
- Der <element_node>-Befehl hat mindestens einen <attribute_node>-Befehl als direkten untergeordneten Befehl.

Beispiel:

Beispiel 1: Im folgenden Beispiel werden einer DB2 UDB-Tabelle 'payment-request-id' und 'amount' zugeordnet. 'sender' hat einen <attribute_node>-Befehl als direkten untergeordneten Befehl. 'payment-request-id', 'amount' und 'sender' sind alle direkte untergeordnete Elemente von 'payment-request':

```
<element_node name="payment-request" multi_occurrence="YES">
  <element_node name="payment-request-id">
          <text_node>
           <RDB_node>
            <table name="fakebank.payments"/>
            <column name="statement_id" type="varchar(30)"/>
          </RDB_node>
      </text_node>
  </element_node>
 <element_node name ="amount">
          <text_node>
           <RDB_node>
            <table name="fakebank.payments"/>
           <column name="amount" type="decimal(8,2)"/>
          </RDB_node>
      </text_node>
  </element_node>
  <element_node name ="sender">
     <attribute_node name ="ID">
           <RDB_node>
            <table name="fakebank.payments"/>
            <column name="sender_ID" type="decimal(8,2)"/>
          </RDB_node>
      </attribute_node>
  </element_node>
  </element_node>
```

Das DAD-Prüfprogramm gibt alle <element_node>-Befehle an, deren Attribut 'multi_occurrence' auf NO gesetzt ist.

Beispiel 2: Das folgende Beispiel ist ein Fragment einer Ausgabetextdatei, die vorschlägt, dass für bestimmte <element_node>-Befehle das Attribut 'multi_occurrence' auf YES gesetzt werden sollen.

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4       <element_node name="Password">  line(s): 49 75
        <element_node name="Pswd1">
        <element_node name="Pswd2">
```

Namensunverträglichkeiten von Attributen und Elementen

In XML-Dokumenten können Elemente mit demselben Namen in unterschiedlichen Kontexten auftreten, z. B. wenn sie unterschiedliche übergeordnete Elemente haben. Attribute und Elemente können identische Namen haben.

Mit dem DAD-Prüfprogramm können XML-Dokumente auf Namenskonflikte überprüft werden. Wenn mehr als eines der im Konflikt stehenden Elemente oder Attribute zugeordnet werden muss, sollten Namensänderungen am Dokument und an der DTD vorgenommen werden.

Am besten überprüfen Sie das XML-Dokument, bevor die DAD-Datei erstellt wird. Das DAD-Prüfprogramm prüft das XML-Dokument nicht anhand seiner DTD.

Beispiel:

Das folgende Beispiel ist ein Fragment eines XML-Dokuments, in dem Namenskonflikte auftreten:

```
<A1>
   <B>
      <C>
          ....
<A2>
   <B>
      <C>
          ....
<D C="attValue">
.....

Wenn das <C>-Element und das C-Attribut zugeordnet werden sollen, bestehen
in der resultierenden DAD-Datei die folgenden Konflikte mit doppelten Namen:
 <element_node name = "A1">
    <element_node name = "B">
       <element_node name = "C">
          <text_node>
            .....
<element_node name = "A2">
    <element_node name = "B">
       <element_node name = "C">
          <text_node>
            ....
      <element_node name = "D">
         <attribute_node name = "C">
          ....
  </element_node>
```

Die zwei <element_node name = ″C″>-Befehle und der <attribute_node name = ″C″>-Befehl sind in der DAD-Datei doppelt.

Kapitel 10. Gespeicherte Prozeduren von XML Extender

Übersicht: Gespeicherte Prozeduren von XML Extender

XML Extender bietet gespeicherte Prozeduren zur Verwaltung von XML-Spalten und -Objektgruppen. Diese gespeicherten Prozeduren können vom DB2-Client aus aufgerufen werden. Die Clientschnittstelle 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 Systemverwaltung* for details about how to call stored procedures.

Die gespeicherten Prozeduren verwenden das Schema DB2XML, das dem Schemanamen von XML Extender entspricht.

XML Extender bietet drei Arten von gespeicherten Prozeduren:

Gespeicherte Prozeduren für Verwaltung
Sie unterstützen Benutzer beim Ausführen von Verwaltungstasks.

Gespeicherte Prozeduren für Zusammensetzung
Sie generieren XML-Dokumente mit Hilfe von Daten in vorhandenen Datenbanktabellen.

Gespeicherte Prozeduren für Zerlegung
Sie zerlegen eingehende XML-Dokumente und speichern Daten in neuen oder vorhandenen Datenbanktabellen.

Stellen Sie sicher, dass Sie die externen Header-Dateien für XML Extender in dem Programm angeben, das die gespeicherten Prozeduren aufruft. Die Header-Dateien befinden sich im Verzeichnis *"$dxx_install_verz$*\dxx\samples\db2xml\include". Dabei ist *$dxx_install_verz$* das Verzeichnis, in dem Sie DB2 XML Extender installiert haben. Die Header-Dateien sind:

dxx.h Die von XML Extender definierten Konstanten und Datentypen

dxxrc.h Der XML Extender-Rückkehrcode

Die Syntax zur Angabe dieser Header-Dateien lautet:

```
#include "dxx.h"
#include "dxxrc.h"
```

Stellen Sie sicher, dass der Pfad für die Header-Dateien in der Make-Datei mit der Kompilierungsoption angegeben ist.

Gespeicherte Prozeduren von XML Extender aufrufen

Wenn Sie die Namen von gespeicherten Prozeduren in Groß- und Kleinbuchstaben schreiben, können Sie XML Extender unter verschiedenen Betriebssystemen von einer einzigen Clientanwendung aus verwenden. Um die gespeicherten Prozeduren auf diese Weise aufzurufen, verwenden Sie die Versionen `result_colname` und `valid_colname` der gespeicherten Prozeduren für die Zusammensetzung. Durch die Verwendung dieser Methoden ergeben sich folgende Vorteile:

- Sie können diese gespeicherten Prozeduren in allen DB2 Universal Database-Umgebungen verwenden, da Sie viele Spalten in der Ergebnistabelle

einschließen können. Für die Versionen der gespeicherten Prozeduren, die `result_colname` und `valid_colname` nicht unterstützen, ist genau eine Spalte in der Ergebnistabelle erforderlich.

- Sie können eine deklarierte temporäre Tabelle als Ergebnistabelle verwenden. Ihre temporäre Tabelle ist durch ein Schema gekennzeichnet, das auf "session" gesetzt ist. Mit deklarierten temporären Tabellen können Sie Mehrbenutzer-Client-Umgebungen unterstützen.

Verwenden Sie Großbuchstaben, wenn Sie die gespeicherten Prozeduren von DB2 XML Extender aufrufen, um konsistent über die verschiedenen Plattformen auf die gespeicherten Prozeduren zuzugreifen.

Voraussetzungen: Binden Sie Ihre Datenbank mit der gespeicherten Prozedur von XML Extender und den DB2 UDB-CLI-Bindedateien. Sie können eine Beispielbefehlsdatei `getstart_prep.cmd` zum Binden der Dateien verwenden. Diese Befehlsdatei befindet sich im Verzeichnis "c:*dxx_install_verz*\samples\db2xml\cmd". Gehen Sie wie folgt vor, um eine Bindeoperation durchzuführen:

1. Stellen Sie die Verbindung zur Datenbank her. Beispiel:
   ```
   db2 "connect to SALES_DB"
   ```
2. Wechseln Sie in das Verzeichnis "c:*dxx_install_verz*\samples\db2xml\bnd", und binden Sie XML Extender an die Datenbank.
   ```
   db2 "bind @dxxbind.lst"
   ```
3. Wechseln Sie in das Verzeichnis "c:*dxx_install_verz*\samples\db2xml\bnd" und binden Sie die CLI an die Datenbank.
   ```
   db2 "bind @db2cli.lst"
   ```
4. Beenden Sie die Verbindung.
   ```
   db2 "terminate"
   ```

Vorgehensweise:

Rufen Sie XML Extender mit der folgenden Syntax auf:

```
CALL DB2XML.funktionseingangspunkt
```

Hierbei gilt Folgendes:

funktionseingangspunkt
Gibt den Namen der Funktion 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.

Beispiele für den Aufruf von gespeicherten Prozeduren finden Sie in den Verzeichnissen `dxx_install_verz/samples/db2xml/c` und `dxx_install_verz/samples/db2xml/cli`. Im Verzeichnis `dxx_install_verz/samples/db2xml/c` werden SQX-Codedateien unter Verwendung von eingebettetem SQL zum Aufrufen gespeicherter Prozeduren für die XML-Objektgruppe zur Verfügung gestellt. Die Beispieldateien im Verzeichnis `dxx_install_verz/samples/db2xml/cli` zeigen, wie gespeicherte Prozeduren über die CLI (Call Level Interface) aufgerufen werden.

CLOB-Begrenzung für gespeicherte Prozeduren vergrößern

Der Standardgrenzwert für CLOB-Parameter bei der Übergabe an eine gespeicherte Prozedur ist 1 MB. Sie können den Grenzwert vergrößern.

Vorgehensweise

So erhöhen Sie den CLOB-Grenzwert:

1. Löschen Sie alle gespeicherten Prozeduren. Beispiel:

```
db2   "drop procedure DB2XML.dxxShredXML restrict"
```

2. Erstellen einer neuen Prozedur mit der vergrößerten CLOB-Begrenzung. Geben Sie Folgendes ein, um den CLOB-Grenzwert beispielsweise auf 10 Megabyte zu vergrößern:

```
db2   "create procedure DB2XML!dxxShredXML(in     dadBuf      clob(100K),
                                           in     XMLObj      clob(10M),
                                           out    returnCode  integer,
                                           out    returnMsg   varchar(1024)
                                           )
        external name 'DB2XML.dxxShredXML_STP'
        language C
        parameter style SQL
        not deterministic
        fenced
        null call"
```

Zugehörige Tasks:

- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205
- „Gespeicherte Prozeduren, die CLOBs zurückgeben" auf Seite 207

Gespeicherte Prozeduren, die CLOBs zurückgeben

Wenn Sie über CLOB-Dateien verfügen, die größer als 1 MB sind, können Sie die Parameter der gespeicherten Prozedur erneut definieren. Hierzu erstellen Sie mit folgenden Befehlen eine Datei und führen diese aus:

```
drop procedure db2xml.dxxGenXMLClob;

create procedure db2xml.dxxGenXMLClob(
          in     dadBuf        clob(100K),
          in     overrideType  integer,
          in     override      varchar(32672),
          out    resultDoc     clob(1M),
          out    valid         integer,
          out    numDocs       integer,
                                      out    returnCode  integer,
                                      out    returnMsg   varchar(1024)
          )
          external name 'db2xml!dxxGenXMLClob'
          specific DB2XML.DXXGENXMLCLOB
       language C
          parameter style DB2DARI
       not deterministic
       fenced
       null call;
drop procedure db2xml.dxxRetrieveXMLClob;

create procedure db2xml.dxxRetrieveXMLClob(
          in   collectionName    varchar(128),
          in   overrideType      integer,
          in   override          varchar(32672),
```

```
   out  resultDoc         clob(1M),
   out  valid             integer,
   out  numDocs           integer,
                                    out    returnCode  integer,
                                    out    returnMsg   varchar(1024)
   )
   external name 'db2xml!dxxRetrieveXMLClob'
   specific DB2XML.DXXRETRIEVEXMLCLOB
language C
   parameter style DB2DARI
not deterministic
fenced
```

Gehen Sie wie folgt vor, um die CLOB-Länge anzugeben: Öffnen Sie die Datei in einem Editor, und ändern Sie den Parameter *resultDoc*, wie im folgenden Beispiel dargestellt:

```
out resultDoc clob(clob_größe),
```

Wenn mehr als ein Dokument generiert wurde, gibt die gespeicherte Prozedur das erste Dokument zurück.

Größenempfehlung: Die Größenbeschränkung des Parameters *resultDoc* hängt von Ihrer Systemkonfiguration ab. Beachten Sie, dass es sich bei diesem Parameter um die durch JDBC zugeordnete Größe handelt, wobei die Größe des Dokuments keine Rolle spielt. Die Größe sollte Platz für Ihre umfangreichsten XML-Dateien bieten, aber 1,5 Gigabyte nicht überschreiten.

Geben Sie zur Ausführung der Befehlsdatei von der DB2-Befehlszeile aus in dem Verzeichnis, in dem sich die Datei befindet, Folgendes ein:

```
db2 -tf crtgenxc.db2
```

Zugehörige Tasks:

• „Gespeicherte Prozeduren von XML Extender aufrufen” auf Seite 205

Gespeicherte Prozeduren von XML Extender zur Verwaltung

Übersicht: Gespeicherte Verwaltungsprozeduren von XML Extender

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()
- dxxDisableDB()
- dxxEnableColumn()
- dxxDisableColumn()
- dxxEnableCollection()
- dxxDisableCollection()

Gespeicherte Prozedur dxxEnableDB()

Zweck:

Aktiviert die Datenbank. Wenn die Datenbank aktiviert ist, erstellt XML Extender die folgenden Objekte:
- Die benutzerdefinierten Typen (UDTs) von XML Extender.
- Die benutzerdefinierten Funktionen (UDFs) von XML Extender.
- Die DTD-Repositorytabelle von XML Extender, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden.
- Die XML Extender-Nutzungstabelle, XML_USAGE, in der allgemeine Informationen für die einzelnen Spalten, die für XML aktiviert sind, und für die einzelnen Objektgruppen gespeichert werden.

Syntax:

```
DB2XML.dxxEnableDB(char(dbName) dbName,       /* Eingabe */
            long           returnCode,   /* Ausgabe */
            varchar(1024)  returnMsg)    /* Ausgabe */
```

Parameter:

Tabelle 47. 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

Zugehörige Konzepte:
- „Übersicht: Gespeicherte Verwaltungsprozeduren von XML Extender" auf Seite 208
- Kapitel 13, „Tabellen zur Verwaltungsunterstützung von XML Extender", auf Seite 287

Zugehörige Tasks:
- „Datenbanken für XML aktivieren" auf Seite 57
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:
- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxDisableDB()

Zweck:

Inaktiviert die Datenbank. Wenn der XML Extender die Datenbank inaktiviert, werden die folgenden Objekte freigegeben:
- Die benutzerdefinierten Typen (UDTs) von XML Extender.
- Die benutzerdefinierten Funktionen (UDFs) von XML Extender.

- Die DTD-Repositorytabelle von XML Extender, DTD_REF, in der DTDs und Informationen zu jeder DTD gespeichert werden.
- Die XML Extender-Nutzungstabelle, XML_USAGE, in der allgemeine Informationen für die einzelnen Spalten, die für XML aktiviert sind, und für die einzelnen Objektgruppen gespeichert werden.

Wichtig: Sie müssen alle XML-Spalten inaktivieren, bevor Sie versuchen, eine Datenbank zu inaktivieren. XML Extender kann eine Datenbank nicht inaktivieren, wenn diese Spalten oder Objektgruppen enthält, die für XML aktiviert wurden.

Syntax:

```
DB2XML.dxxDisableDB(char(dbName)     dbName,       /* Eingabe */
                    long          returnCode,   /* Ausgabe */
                    varchar(1024) returnMsg)    /* Ausgabe */
```

Parameter:

Tabelle 48. 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

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Verwaltungsprozeduren von XML Extender" auf Seite 208
- Kapitel 13, „Tabellen zur Verwaltungsunterstützung von XML Extender", auf Seite 287

Zugehörige Tasks:

- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxEnableColumn()

Zweck:

Aktiviert eine XML-Spalte. Beim Aktivieren einer Spalte führt XML Extender die folgenden Tasks aus:

- Festlegen, ob die XML-Tabelle einen Primärschlüssel hat; falls nicht, ändert XML Extender die XML-Tabelle und fügt eine Spalte mit dem Namen DXXROOT_ID hinzu.
- Erstellen der in der DAD-Datei angegebenen Nebentabellen mit einer Spalte für die eindeutige Kennung jeder Zeile in der XML-Tabelle. Diese Spalte ist entweder die Root-ID, die vom Benutzer angegeben wurde, oder die von XML Extender benannte DXXROOT_ID.
- Erstellen einer Standardsicht für die XML-Tabelle und ihre Nebentabellen, wobei optional ein von Ihnen angegebener Name verwendet wird.

Syntax:

```
DB2XML.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 49. 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
tabellenbereich	Der Tabellenbereich, der die Nebentabelle enthält, sofern es sich dabei nicht um den Standardtabellenbereich handelt. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.	IN
standardsicht	Der Name der Standardsicht, die die Anwendungstabelle und die Nebentabellen verknüpft.	IN
rootID	Der Name des einzelnen Primärschlüssels in der Anwendungstabelle, der als Root-ID für die Nebentabelle verwendet werden soll.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Zugehörige Konzepte:

- „XML-Spalten als Speicher- und Zugriffsmethode" auf Seite 80
- „Übersicht: Gespeicherte Verwaltungsprozeduren von XML Extender" auf Seite 208

Zugehörige Tasks:

- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxDisableColumn()

Zweck:

Inaktiviert die für XML aktivierte Spalte. Wenn eine XML-Spalte inaktiviert ist, kann sie keine XML-Datentypen mehr enthalten.

Syntax:

```
DB2XML.dxxDisableColumn(char(dbName) dbName,      /* Eingabe */
                char(tbName) tbName,      /* Eingabe */
                char(colName) colName,     /* Eingabe */
                long           returnCode,  /* Ausgabe */
                varchar(1024)  returnMsg)   /* Ausgabe */
```

Parameter:

Tabelle 50. 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

Zugehörige Referenzen:

- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxEnableCollection()

Zweck:

Aktiviert eine XML-Objektgruppe, die einer Anwendungstabelle zugeordnet ist.

Syntax:

```
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 51. 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

Tabelle 51. Parameter für dxxEnableCollection() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
tablespace	Der Tabellenbereich, der die Nebentabelle enthält, sofern es sich dabei nicht um den Standard-tabellenbereich handelt. Ist kein Tabellenbereich angegeben, wird der Standardtabellenbereich verwendet.	IN
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Zugehörige Konzepte:

- „XML-Objektgruppen als Speicher- und Zugriffsmethode" auf Seite 97
- „Übersicht: Gespeicherte Verwaltungsprozeduren von XML Extender" auf Seite 208

Zugehörige Tasks:

- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxDisableCollection()

Zweck:

Inaktiviert eine für XML aktivierte Objektgruppe durch das Entfernen von Markierungen, die Tabellen und Spalten als Teil einer Objektgruppe kennzeichnen.

Syntax:

```
dxxDisableCollection(char(dbName) dbName,      /* Eingabe */
                     char(colName) colName,    /* Eingabe */
                      long           returnCode,             /* Ausgabe */
                      varchar(1024)  returnMsg)              /* Ausgabe */
```

Parameter:

Tabelle 52. 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

Zugehörige Referenzen:

- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozeduren von XML Extender zur Zusammensetzung

Übersicht: Gespeicherte Zusammensetzungsprozeduren von XML Extender

Die gespeicherten Prozeduren zur Zusammensetzung, dxxGenXML(), dxxRetrieveXML(), dxxGenXMLCLOB() und dxxRetrieveXMLCLOB(), 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.

Die folgenden Leistungsverbesserungen wurden für gespeicherte Prozeduren für Zusammensetzung vorgenommen.

- Für die Betriebssysteme UNIX® und Windows® wurde die Länge des Überschreibungsparameters von 1 KB auf 32 KB vergrößert.

  Der Überschreibungsparameter von 1 KB schränkte die Länge der SQL-Anweisung für die SQL-Zusammensetzung ein. Die Einschränkung führte zur Verwendung von Datenbanksichten, um die Länge der erforderlichen SQL-Anweisung zu verringern. Datenbanksichten führten jedoch in manchen Fällen zu zusätzlichen Pfadlängen aufgrund der Sichtdatenspeicherung. Mit einem größeren Überschreibungsparameter sind Sichten nicht mehr in dem Maße erforderlich.
- Zwischenergebnistabellen sind nicht mehr erforderlich.
- Durch Verwendung dieser gespeicherten Prozeduren können Sie:
  - Die Anweisungspfadlänge verringern, da keine Ergebnistabellen erstellt werden müssen.
  - Ihre Programmierung vereinfachen.
- Verwenden Sie die gespeicherten Prozeduren, für die eine Zwischenergebnistabelle erforderlich ist, wenn Sie mehr als ein Dokument erzeugen wollen.
- Die benutzerdefinierten Funktionen für die XML-Spalte wurden aus Leistungsgründen verbessert.
- Die benutzerdefinierten Funktionen von DB2® UDB XML Extender behalten jetzt während der Verarbeitung kleine (512 KB) XML-Dokumente im Hauptspeicher. Dadurch wird die Ein-/Ausgabeaktivität und die Konkurrenzsituation für die für temporäre Dateien verwendete Platte verringert.
- Die Definition der skalaren (nicht tabellenbezogenen) benutzerdefinierten Funktionen von DB2 UDB XML Extender wurde geändert, so dass sie parallel ausgeführt werden können. Diese Änderung führt zu wesentlichen Leistungsverbesserungen bei der Ausführung von Abfragen, die sich mehr als einmal auf benutzerdefinierte Funktionen beziehen. Sie müssen die Migrationsprozedur ausführen, um auf die parallele Funktionsfähigkeit für die skalaren UDFs zugreifen zu können. Wenn Sie bereits Spalten mit Hilfe der skalaren UDFs aktiviert haben, müssen Sie alle Spalten inaktivieren, die Migrationsprozedur ausführen und die Spalten erneut aktivieren.

Gespeicherte Prozedur 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.

Syntax:

```
dxxGenXML(CLOB(100K)     DAD,       /* Eingabe */
          char(resultTabName) resultTabName, /* Eingabe */
          char resultColName, char resultValidCol /* Eingabe */

     char(30)       valid_column, /* Eingabe */
    integer        overrideType  /* Eingabe */
          varchar(1024)   override,     /* Eingabe */
      integer        maxRows,      /* Eingabe */
          integer         numRows,      /* Ausgabe */
          long            returnCode,   /* Ausgabe */
          varchar(1024)   returnMsg       /* Ausgabe */
```

Dabei hat *varchar_wert* den Wert 32672 für Windows und UNIX sowie den Wert 16366 für iSeries und z/OS.

Parameter:

Tabelle 53. 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
result_column	Der Name der Spalte in der Ergebnistabelle, in der die zusammengesetzten XML-Dokumente gespeichert werden.	IN

Tabelle 53. Parameter für dxxGenXML() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
valid_column	Der Name der Spalte, die angibt, ob das XML-Dokument gültig ist, wenn es anhand der Dokumentartdefinition (DTD) überprüft wird.	IN
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
resultDoc	Ein CLOB mit dem zusammengesetzten XML-Dokument.	OUT
valid	Der Parameter 'valid' wird wie folgt gesetzt: • Wenn VALIDATION=YES definiert ist, valid=1 für erfolgreiche Prüfung bzw. valid=0 für nicht erfolgreiche Prüfung. • Wenn VALIDATION=NO definiert ist, valid=NULL.	OUT
maxRows	Die maximale Anzahl von Zeilen in der Ergebnistabelle.	IN
numRows	Die tatsächliche Anzahl generierter Zeilen in der Ergebnistabelle.	OUT
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Beispiele:

Im folgenden Beispielfragment wird davon ausgegangen, dass die Ergebnistabelle mit dem Namen XML_ORDER_TAB erstellt wird und eine Spalte des Typs XML-VARCHAR enthält. Ein vollständiges ausführbares Beispiel befindet sich in DXXSAMPLES/QCSRC(GENX).

```
#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;       /* tatsächliche Anzahl Zeilen */
long    returnCode;         /* Rückkehrfehlercode */
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 SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

              /* Daten aus einer Datei in ein CLOB einlesen */
              strcpy(dadfile.name,"dxxinstall/dad/litem3.dad");
              dadfile.name_length = strlen("dxxinstall/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);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zusammensetzungsprozeduren von XML Extender" auf Seite 214

Zugehörige Tasks:

- „XML-Dokumente mit Hilfe der SQL-Zuordnung zusammensetzen" auf Seite 65
- „XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen" auf Seite 68
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix

Gespeicherte Prozedur dxxRetrieveXML()

Zweck:

Die gespeicherte Prozedur dxxRetrieveXML() dient 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.

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.

Syntax:

```
dxxRetrieveXML(char(collectionName) collectionName,   /* Eingabe */
               char(resultTabName) resultTabName,    /* Eingabe */
               char resultColName, char resultValidCol   /* Eingabe */

               integer        overrideType,     /* Eingabe */
               varchar_wert   override,         /* Eingabe */
          integer        maxRows,          /* Eingabe */
               integer        numRows,          /* Ausgabe */
               long           returnCode,       /* Ausgabe */
               varchar(1024)  returnMsg)       /* Ausgabe */
```

Dabei hat *varchar_wert* den Wert 32672 für Windows und UNIX sowie den Wert 16366 für iSeries und z/OS.

Parameter:

Tabelle 54. 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
result_column	Der Name der Spalte in der Ergebnistabelle, in der die zusammengesetzten XML-Dokumente gespeichert werden.	IN
valid_column	Der Name der Spalte, die angibt, ob das XML-Dokument gültig ist, wenn es anhand der Dokumentartdefinition (DTD) überprüft wird.	IN
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
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Beispiele:

Das folgende Fragment 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. Ein vollständiges ausführbares Beispiel befindet sich in dxx_install_verz\samples\db2xml\qcsrc(rtrx).

```
#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];       /* Überschr., auf NULL gesetzt */
              short   overrideType;     /* definiert in dxx.h */
              short   max_row;          /* maximale Anzahl Zeilen */
              short   num_row;          /* tatsächliche Anzahl Zeilen */
              long    returnCode;       /* Rückkehrfehlercode */
              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);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zusammensetzungsprozeduren von XML Extender" auf Seite 214

Zugehörige Tasks:

- „XML-Dokumente mit Hilfe der SQL-Zuordnung zusammensetzen" auf Seite 65
- „XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen" auf Seite 68
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxGenXMLClob

Zweck:

Als Eingabe erhält dxxGenXMLClob einen Puffer mit der DAD. Es erstellt XML-Dokumente mit Hilfe von Daten in XML-Objektgruppentabellen, die durch das Element `<Xcollection>` in der DAD angegeben sind, und es gibt das erste und normalerweise einzige XML-Dokument, im CLOB *resultDoc* generiert, zurück.

Syntax:

```
dxxGenXMLClob(CLOB(100k)                    DAD             /*Eingabe*/
              integer                       overrideType,   /*Eingabe*/
              varchar(varchar_wert)         override,       /*Eingabe*/
              CLOB(1M)                      resultDoc,      /*Ausgabe*/
              integer                       valid,          /*Ausgabe*/
              integer                       numDocs,        /*Ausgabe*/
              long                          returnCode,     /*Ausgabe*/
              varchar(1024)                 returnMsg),     /*Ausgabe*/
```

Dabei hat *varchar_wert* den Wert 32672 für Windows und UNIX sowie den Wert 16366 für iSeries und z/OS.

Parameter:

Tabelle 55. Parameter für dxxGenXMLClob

Parameter	Beschreibung	IN/OUT-Parameter
DAD	Ein CLOB mit der DAD-Datei.	IN
overrideType	Eine Markierung, die den Typ des Parameters *override* angibt: **NO_OVERRIDE** Kein Überschreiben. **SQL_OVERRIDE** Überschreiben durch eine SQL_stmt. **XML_OVERRIDE** Überschreiben durch eine XPath-Bedingung.	IN

Tabelle 55. Parameter für dxxGenXMLClob (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
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
resultDoc	Ein CLOB mit dem zusammengesetzten XML-Dokument.	OUT
valid	Der Parameter 'valid' wird wie folgt gesetzt: • Wenn VALIDATION=YES definiert ist, valid=1 für erfolgreiche Prüfung bzw. valid=0 für nicht erfolgreiche Prüfung. • Wenn VALIDATION=NO definiert ist, valid=NULL.	OUT
numDocs	Die Anzahl von XML-Dokumenten, die aus den Eingabedaten generiert wird. **Anmerkung:** Momentan wird nur das erste Dokument zurückgegeben.	OUT
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Die CLOB-Parametergröße ist 1 MB. Wenn Sie CLOB-Dateien haben, die größer als 1 MB sind, bietet XML Extender eine Befehlsdatei, mit der die Parameter der gespeicherten Prozedur erneut definieren werden können. Laden Sie die Datei `crtgenxc.zip` von der Website von DB2 UDB XML Extender herunter. Diese ZIP-Datei enthält die folgenden Programme:

crtgenxc.db2
Zur Verwendung mit XML Extender V7.2 FixPak 5 und später für UNIX und Windows.

crtgenxc.iseries
Zur Verwendung mit XML Extender für iSeries.

crtgenxc.zox.jci und crtgenxc.zos.cmd
Zur Verwendung mit XML Extender für OS/390 V7, APAR PQ58249 und später.

So geben Sie die CLOB-Länge an: Öffnen Sie die Datei in einem Editor, und ändern Sie den Parameter *resultDoc*, wie im folgenden Beispiel gezeigt.

```
out    resultDoc     clob(clob_größe),
```

Größenempfehlung: Die Größenbeschränkung des Parameters *resultDoc* hängt von Ihrer Systemkonfiguration ab. Beachten Sie, dass es sich bei diesem Parameter um die durch JDBC zugeordnete Größe handelt, wobei die Größe des Dokuments keine Rolle spielt. Die Größe sollte Platz für Ihre umfangreichsten XML-Dateien bieten, aber 1,5 Gigabyte nicht überschreiten.

Geben Sie zur Ausführung der Befehlsdatei unter UNIX oder Windows von der DB2 UDB-Befehlszeile aus in dem Verzeichnis, in dem sich die Datei befindet, Folgendes ein:

```
db2   -tf  crtgenxc.db2
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zusammensetzungsprozeduren von XML Extender" auf Seite 214

Zugehörige Tasks:

- „XML-Dokumente mit Hilfe der SQL-Zuordnung zusammensetzen" auf Seite 65
- „XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen" auf Seite 68
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxRetrieveXMLClob

Zweck:

Die gespeicherte Prozedur dxxRetrieveXMLClob ermöglicht die Zusammensetzung von Dokumenten aus relationalen Daten.

Die Voraussetzungen für die Verwendung von dxxRetrieveXMLClob entsprechen denen von dxxGenXMLClob. Der einzige Unterschied liegt darin, dass die DAD-Datei kein Eingabeparameter für dxxRetrieveXMLClob ist, sondern der Name einer aktivierten XML-Objektgruppe.

Syntax:

```
dxxRetrieveXMLClob(char(collectionName)          collectionName  /*Eingabe*/
                integer                          overrideType,   /*Eingabe*/
                varchar(varchar_value)   override,       /*Eingabe*/
                CLOB(1M)                         resultDoc,      /*Ausgabe*/
                integer                          valid,          /*Ausgabe*/
                integer                          numDocs,        /*Ausgabe*/
                long                             returnCode,     /*Ausgabe*/
                varchar(1024)                    returnMsg),     /*Ausgabe*/
```

Parameter:

Tabelle 56. Parameter für dxxRetrieveXMLClob

Parameter	Beschreibung	IN/OUT-Parameter
collectionName	Der Name einer aktivierten XML-Objektgruppe	IN
overrideType	Eine Markierung, die den Typ des Parameters *override* angibt: **NO_OVERRIDE** Kein Überschreiben. **SQL_OVERRIDE** Überschreiben durch eine SQL_stmt. **XML_OVERRIDE** Überschreiben durch eine XPath-Bedingung.	IN

Tabelle 56. Parameter für dxxRetrieveXMLClob (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
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
resultDoc	Die maximale Anzahl von Zeilen in der Ergebnistabelle.	IN
valid	Der Parameter 'valid' wird wie folgt gesetzt: • Wenn VALIDATION=YES definiert ist, valid=1 für erfolgreiche Prüfung bzw. valid=0 für nicht erfolgreiche Prüfung. • Wenn VALIDATION=NO definiert ist, valid=NULL.	OUT
numDocs	Die Anzahl von XML-Dokumenten, die aus den Eingabedaten generiert wird. Anmerkung: Momentan wird nur das erste Dokument zurückgegeben.	OUT
returnCode	Der Rückkehrcode von der gespeicherten Prozedur.	OUT
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Die CLOB-Parametergröße ist 1 MB. Wenn Sie CLOB-Dateien haben, die größer als 1 MB sind, bietet XML Extender eine Befehlsdatei, mit der die Parameter der gespeicherten Prozedur erneut definieren werden können. Laden Sie die Datei `crtgenxc.zip` von der Website von DB2 UDB XML Extender herunter. Diese ZIP-Datei enthält die folgenden Programme:

crtgenxc.db2
Zur Verwendung mit XML Extender V7.2 FixPak 5 und später für UNIX und Windows.

crtgenxc.iseries
Zur Verwendung mit XML Extender für iSeries.

crtgenxc.zox.jci und crtgenxc.zos.cmd
Zur Verwendung mit XML Extender für OS/390 V7, APAR PQ58249 und später.

So geben Sie die CLOB-Länge an: Öffnen Sie die Datei in einem Editor und ändern Sie den Parameter *resultDoc*, wie im folgenden Beispiel gezeigt.

```
out    resultDoc     clob(clob_größe),
```

Größenempfehlung: Die Größenbeschränkung des Parameters *resultDoc* hängt von Ihrer Systemkonfiguration ab. Beachten Sie, dass es sich bei diesem Parameter um die durch JDBC zugeordnete Größe handelt, wobei die Größe des Dokuments keine Rolle spielt. Die Größe sollte Platz für Ihre umfangreichsten XML-Dateien bieten, aber 1,5 Gigabyte nicht überschreiten.

Geben Sie zur Ausführung der Befehlsdatei unter UNIX oder Windows von der DB2 UDB-Befehlszeile aus in dem Verzeichnis, in dem sich die Datei befindet, Folgendes ein:

```
db2   -tf  crtgenxc.db2
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zusammensetzungsprozeduren von XML Extender" auf Seite 214

Zugehörige Tasks:

- „XML-Dokumente mit Hilfe der SQL-Zuordnung zusammensetzen" auf Seite 65
- „XML-Objektgruppen mit Hilfe der RDB_node-Zuordnung zusammensetzen" auf Seite 68
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozeduren von XML Extender zum Zerlegen

Übersicht: Gespeicherte Zerlegungsprozeduren von XML Extender

Mit den gespeicherten Prozeduren zur Zerlegung, 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.

Gespeicherte Prozedur dxxShredXML()

Zweck:

Zerlegt auf der Basis einer DAD-Dateizuordnung XML-Dokumente, wobei der Inhalt der XML-Elemente und -Attribute in den angegebenen DB2 UDB-Tabellen gespeichert wird. 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. Für die gespeicherte Prozedur ist erforderlich, dass die in der Verknüpfungsbedingung, in der DAD, angegebenen Spalten den Primär-/Fremdschlüsselbeziehungen in den vorhandenen Tabellen entsprechen. Die im RDB_node des Root-element_node angegebenen Spalten mit den Verknüpfungsbedingungen müssen in den Tabellen vorhanden sein.

Das Fragment einer gespeicherten Prozedur in diesem Abschnitt dient zur Erläuterung.

Syntax:

```
dxxShredXML(CLOB(100K)    DAD,           /* Eingabe */
            CLOB(1M)      xmlobj,        /* Eingabe */
            long          returnCode,    /* Ausgabe */
            varchar(1024) returnMsg)     /* Ausgabe */
```

Parameter:

Tabelle 57. 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 Fragment zeigt ein Beispiel für 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,"dxx_install_verz
              /samples/db2xml/dad/getstart_xcollection.dad
              ");
              dadFile.name_length=strlen("dxx_install_verz
              /samples/db2xml/dad/getstart_xcollection.dad
              ");
              dadFile.file_option=SQL_FILE_READ;
              strcpy(xmlFile.name,"dxx_install_verz
                /samples/db2xml/xml/getstart.xml");
```

```
xmlFile.name_length=strlen("dxx_install_verz
  /samples/db2xml/xml/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);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zerlegungsprozeduren von XML Extender" auf Seite 226

Zugehörige Tasks:

- „XML-Objektgruppe mit Hilfe der RDB_node-Zuordnung zerlegen" auf Seite 71
- „XML-Dokumente in DB2 UDB-Daten zerlegen" auf Seite 102
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur 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.

Syntax:

```
dxxInsertXML(char(collectionName) collectionName, /* Eingabe */
             CLOB(1M)       xmlobj,          /* Eingabe */
             long           returnCode,      /* Ausgabe */
             varchar(1024)  returnMsg)       /* Ausgabe */
```

Parameter:

Tabelle 58. 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

Tabelle 58. Parameter für dxxInsertXML() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
returnMsg	Der Nachrichtentext, der im Fehlerfall zurückgegeben wird.	OUT

Beispiele:

Im folgenden Beispielfragment zerlegt der Aufruf von dxxInsertXML() das XML-Eingabedokument `dxx_install_verz/xml/order1.xml` und fügt entsprechend der in der DAD-Datei angegebenen Zuordnung, mit der sie aktiviert wurde, Daten in die Objektgruppentabellen 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,"dxx_install_verz/samples
              db2xml/xml/getstart.xml");
              xmlobj.name_length=strlen("dxx_install_verz/samples
              db2xml/xml/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);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Zerlegungsprozeduren von XML Extender" auf Seite 226

Zugehörige Tasks:

- „XML-Objektgruppe mit Hilfe der RDB_node-Zuordnung zerlegen" auf Seite 71
- „XML-Dokumente in DB2 UDB-Daten zerlegen" auf Seite 102
- „Gespeicherte Prozeduren von XML Extender aufrufen" auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Kapitel 11. Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries

Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries

XML Extender bietet zwei Methoden zum Speichern von und Zugreifen auf XML-Daten. Unter Verwendung der XML-Spaltenmethode können Sie XML-Dokumente in einer DB2®-Tabelle speichern, während Sie den Dokumentinhalt abfragen, aktualisieren und abrufen. Mit den benutzerdefinierten MQ-XML-Funktionen können Sie XML-Dokumente abfragen und anschließend die Ergebnisse in einer Nachrichtenwarteschlange veröffentlichen. Darüber hinaus können Sie die XML-Objektgruppenmethode verwenden, um den nicht markierten Inhalt eines XML-Dokuments in einer oder mehreren Tabellen zu speichern bzw. XML-Dokumente aus mehreren Tabellen zusammenzusetzen. Unter Verwendung der gespeicherten MQ-XML-Prozeduren können Sie ein XML-Dokument aus einer Nachrichtenwarteschlange abrufen, es in nicht markierte Daten zerlegen und die Daten in DB2 UDB-Tabellen speichern. Außerdem können Sie ein XML-Dokument aus DB2-Daten zusammensetzen und das Dokument an eine MQSeries®-Nachrichtenwarteschlange senden.

MQSeries unterstützt drei Nachrichtenübertragungsmodelle zum Verteilen von XML-Daten und -Dokumenten:

Datagramme
Nachrichten werden an eine einzelne Zieladresse geschickt, wobei keine Antwort erwartet wird.

Publish/Subscribe (Veröffentlichen/Subskribieren)
Ein oder mehrere Absender senden eine Nachricht an einen Veröffentlichungsservice, der die Nachricht an interessierte Teilnehmer verteilt.

Request/Reply (Anfordern/Antworten)
Nachrichten werden an eine einzelne Zieladresse geschickt und der Absender erwartet eine Antwort.

MQSeries kann auf vielfältige Weise verwendet werden. Einfache Datagramme werden ausgetauscht, um mehrere Anwendungen zu koordinieren, um Informationen auszutauschen, Services anzufordern und um Hinweise zu interessanten Ereignissen zu verteilen. 'Publish/Subscribe' wird meist dazu verwendet, um Echtzeitinformationen möglichst aktuell zu verbreiten. 'Request/Reply' wird im Allgemeinen als einfache Form des pseudo-synchronen Aufrufs ferner Prozeduren verwendet. Komplexere Modelle können auch konstruiert werden, indem diese Basismodelle kombiniert werden.

Die hier beschriebenen grundlegenden Nachrichtenübertragungstechniken werden auf vielfältige Weise verwendet. Da MQSeries für eine Vielzahl von Betriebssystemen verfügbar ist, bietet es einen wichtigen Mechanismus zum Verbinden unterschiedlicher und voneinander unabhängiger Anwendungen, in gleichartigen oder nicht gleichartigen Umgebungen.

Um die MQXML-Funktionen und gespeicherten Prozeduren verwenden zu können, stellen Sie sicher, dass die folgende Software installiert ist.

- DB2 Universal Database™ ab Version 7.2
- DB2 MQSeries Functions Version 7.2, die als optionales Installations-Feature von DB2 Universal Database Version 7.2 verfügbar ist. Installationsinformationen finden Sie in den Release-Informationen von DB2 Universal Database Version 7.2.
- MQSeries Publish/Subscribe oder MQSeries Integrator bei Verwendung der Veröffentlichungsfunktionen.

MQSeries-Funktionen von XML Extender

Übersicht: XML Extender und MQSeries-Funktionen

DB2® XML Extender enthält die folgenden Funktionen zur Verwendung mit MQSeries®:

- MQPublishXML
- MQReadXML
- MQReadAllXML
- MQReadXMLCLOB
- MQReadAllXMLCLOB
- MQReceiveXML
- MQReceiveAllXML
- MQRcvAllXMLCLOB
- MQReceiveXMLCLOB
- MQSENDXML
- MQSENDXMLFILE
- MQSendXMLFILECLOB

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231
- „Übersicht: Gespeicherte Prozeduren von XML Extender für MQSeries" auf Seite 255

Zugehörige Referenzen:

- „Funktion MQReadXML" auf Seite 234
- „Funktion MQReadAllXML" auf Seite 236
- „Funktion MQReceiveXML" auf Seite 242
- „Funktion MQReadXMLCLOB" auf Seite 238
- „Funktion MQReadAllXMLCLOB" auf Seite 239
- „Funktion MQRcvXMLCLOB" auf Seite 249
- „Funktion MQReceiveXMLCLOB" auf Seite 248
- „Funktion MQPublishXML" auf Seite 233
- „Funktion MQReceiveAllXML" auf Seite 243
- „Funktion MQRcvAllXMLCLOB" auf Seite 246
- „Funktion MQSendXMLFILECLOB" auf Seite 254
- „Funktion MQSENDXML" auf Seite 250
- „Funktion MQSENDXMLFILE" auf Seite 252

Funktion MQPublishXML

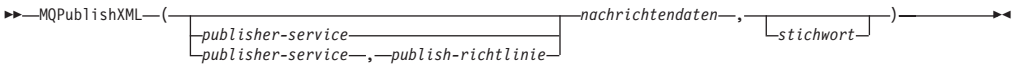
Zweck:

Die Funktion MQPublishXML veröffentlicht XMLVARCHAR- und XMLCLOB-Daten für MQSeries. Für diese Funktion ist die Installation von entweder MQSeries Publish/Subscribe oder MQSeries Integrator erforderlich. Die folgende Website enthält weitere Informationen hierzu.

`http://www.software.ibm.com/MQSeries`

Die Funktion MQPublishXML veröffentlicht die XML-Daten, die in *nachrichtendaten* enthalten sind, für den MQSeries Publisher, der in *publisher-service* angegeben ist, wobei die Eigenschaften der Veröffentlichungsrichtlinie *publish-richtlinie* verwendet werden. Das Stichwort der Nachricht kann optional durch *stichwort* angegeben werden. Eine optionale benutzerdefinierte Nachrichtenkorrelations-ID kann durch *korrel-id* angegeben werden. Die Funktion gibt im Erfolgsfall '1' zurück.

Syntax:



Parameter:

Tabelle 59. Parameter für MQPublishXML

Parameter	Datentyp	Beschreibung
publisher-service	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn *publisher-service* angegeben ist, bezieht er sich auf einen Publisher-Service, der in der Repository-Datei AMT.XML definiert ist. Wenn *publisher-service* nicht angegeben ist, wird DB2.DEFAULT.PUBLISHER verwendet. Die maximale Größe von *publisher-service* ist 48 Byte.
publish-richtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries AMI-Veröffentlichungsrichtlinie, die für die Bearbeitung dieser Nachricht verwendet werden soll. Wenn angegeben, bezieht sich *publish-richtlinie* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Die Veröffentlichungsrichtlinie definiert auch eine Gruppe von Veröffentlichungseigenschaftsoptionen, die für Optionen der Nachrichtenübertragungsoperationen angewendet werden sollen. Zu diesen Optionen gehört die Nachrichtenpriorität und die Nachrichtenpermanenz. Wenn die *servicerichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *servicerichtlinie* ist 48 Byte. Weitere Informationen finden Sie in den Informationen zur MQSeries Application Messaging Interface.

Tabelle 59. Parameter für MQPublishXML (Forts.)

Parameter	Datentyp	Beschreibung
nachrichtendaten	XMLVARCHAR oder XMLCLOB	Ein XMLVARCHAR- oder XMLCLOB-Ausdruck mit den Daten, die über MQSeries gesendet werden.
stichwort	VARCHAR(40)	Eine Zeichenfolge mit dem Stichwort, unter dem die Nachricht veröffentlicht werden soll. Wenn kein Stichwort angegeben ist, wird der Nachricht kein Stichwort zugeordnet. Die maximale Größe von *stichwort* ist 48 Byte. Innerhalb einer Stichwortzeichenfolge können mehrere Stichwörter aufgelistet werden, wobei die einzelnen Stichwörter durch ″:″ voneinander getrennt werden.

Rückkehrcodes:
Falls erfolgreich, gibt die Funktion MQPublishXML 1 zurück. Der Wert 0 wird zurückgegeben, wenn die Funktion nicht erfolgreich ausgeführt wurde.
Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

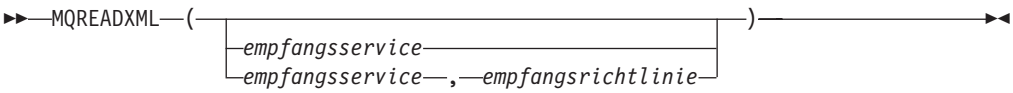
- „Syntaxdiagramme lesen" auf Seite ix

Funktion MQReadXML

Zweck:

Die Funktion MQREADXML gibt XMLVARCHAR-Daten aus dem MQSeries-Standort zurück, der durch das Element *empfangsservice* angegeben wird. Sie verwendet die Eigenschaften von *empfangsrichtlinie*. Die Funktion MQREADXML entfernt keine Nachrichten aus der Warteschlange, die *empfangsservice* zugeordnet ist.

Syntax:



Parameter:

Tabelle 60. Parameter für MQReadXML

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht empfangen werden soll. Wenn der *empfangsservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.

Tabelle 60. Parameter für MQReadXML (Forts.)

Parameter	Datentyp	Beschreibung
empfangsrichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung einer Nachricht verwendet wird. Wenn die *empfangsrichtlinie* angegeben ist, bezieht sie sich auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Eine Empfangsrichtlinie definiert eine Gruppe von Empfangseigenschaftsoptionen, die für Nachrichtenübertragungsoperationen angewendet werden. Zu diesen Optionen gehören die Nachrichtenpriorität und die Nachrichtenpermanenz. Wenn die *empfangsrichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *empfangsrichtlinie* ist 48 Byte.

Ergebnisse:
Wenn eine Nachricht in der Warteschlange erfolgreich gelesen wurde, gibt MQREADXML ein db2xml.xmlvarchar zurück. NULL wird zurückgegeben, wenn keine Nachrichten verfügbar sind.
Beispiele:
Beispiel 1: Dieses Beispiel liest die Nachricht am Kopf der Warteschlange, die durch den Standardservice *DB2.DEFAULT.SERVICE* angegeben ist. Es verwendet die Standardrichtlinie *DB2.DEFAULT.POLICY* zum Lesen der Nachricht.

```
values DB2XML.MQREADXML()
```

Dieses Beispiel gibt den Inhalt der Nachricht als XMLVARCHAR zurück. Wenn keine Nachrichten verfügbar sind, wird NULL zurückgegeben.

Beispiel 2: Dieses Beispiel liest die Nachricht am Kopf der Warteschlange, die durch den Service *MYSERVICE* angegeben ist, wobei die Standardrichtlinie *DB2.DEFAULT.POLICY* verwendet wird.

```
values DB2XML.MQREADXML('MYSERVICE')
```

Im Beispiel wird der Inhalt der Nachricht als XMLVARCHAR zurückgegeben. Wenn keine Nachrichten verfügbar sind, wird NULL zurückgegeben.

Beispiel 3: Dieses Beispiel liest die Nachricht am Kopf der Warteschlange, die durch den Service *MYSERVICE* angegeben ist, wobei die Richtlinie *MYPOLICY* verwendet wird.

```
values DB2XML.MQREADXML('MYSERVICE','MYPOLICY')
```

Wenn das Lesen erfolgreich ist, wird der Inhalt der Nachricht als XMLVARCHAR zurückgegeben. Wenn keine Nachrichten verfügbar sind, wird NULL zurückgegeben.

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix

Funktion MQReadAllXML

Zweck:

Die Funktion MQReadAllXML gibt eine Tabelle mit den Nachrichten und Nachrichtenmetadaten aus dem MQSeries-Standort zurück, der durch *empfangsservice* angegeben ist. Dabei werden die Eigenschaften von *servicerichtlinie* verwendet. Durch die Ausführung dieser Operation werden keine Nachrichten aus der Warteschlange entfernt, die *empfangsservice* zugeordnet ist. Falls *anzahl_zeilen* angegeben ist, wird maximal diese Anzahl an Zeilen zurückgegeben. Wenn 'anzahl_zeilen' nicht angegeben ist, werden alle Nachrichten zurückgegeben.

Syntax:

Parameter:

Tabelle 61. Parameter für MQReadAllXML

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht gelesen werden soll. Wenn der *empfangsservice* angegeben ist, muss er sich auf einen Service beziehen, der in der Repository-Datei AMT.XML definiert ist. Wenn jedoch *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.
servicerichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet wird. Wenn die *servicerichtlinie* angegeben ist, bezieht sie sich auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Die maximale Größe von *empfangsservice* ist 48 Byte. Weitere Informationen finden Sie im Handbuch 'MQSeries Application Messaging Interface'.
anzahl_zeilen	INTEGER	Eine positive ganze Zahl mit der maximalen Anzahl an Nachrichten, die durch die Funktion zurückgegeben werden sollen.

Ergebnisse:
Die Funktion MQReadAllXML gibt eine Tabelle mit Nachrichten und Nachrichtenmetadaten zurück, die im Folgenden beschrieben werden.

Tabelle 62. Tabelle der Ergebnismenge

Spaltenname	Datentyp	Beschreibung
MSG	XMLVARCHAR	Der Inhalt der MQSeries-Nachricht. Die maximale Länge ist 4 Kilobyte.
CORRELID	VARCHAR(24)	Eine Korrelations-ID, die verwendet werden kann, um Bezug auf Nachrichten zu nehmen.
TOPIC	VARCHAR(40)	Das Stichwort, unter dem die Nachricht veröffentlicht wurde, falls verfügbar.
QNAME	VARCHAR(48)	Der Name der Warteschlange, in der die Nachricht empfangen wurde.
MSGID	VARCHAR(24)	Die von MQSeries zugeordnete eindeutige Kennung für eine Nachricht.
MSGFORMAT	VARCHAR(8)	Das von MQSeries definierte Format der Nachricht. Normale Zeichenfolgen haben das Format MQSTR.

Beispiele:
Beispiel 1: Alle Nachrichten aus der Warteschlange, die durch den Standardservice DB2.DEFAULT.SERVICE angegeben sind, werden gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Die Nachrichten und alle Metadaten werden im Tabellenformat zurückgegeben.

```
select * from table (DB2XML.MQREADALLXML()) t
```

Beispiel 2: Alle Nachrichten, die durch den Service MYSERVICE angegeben sind, werden gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Nur die Spalten *msg* und *correlid* werden zurückgegeben. Die Nachrichtenwarteschlange hat das Format einer Tabelle, aus der Sie die gewünschten Felder auswählen können.

```
select t.MSG, t.CORRELID from table (DB2XML.MQREADALLXML('MYSERVICE')) t
```

Beispiel 3: Die Warteschlange, die durch den Standardservice DB2.DEFAULT.SERVICE angegeben ist, wird gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Nur Nachrichten mit einer *CORRELID* von '1234' werden zurückgegeben. Bis zu 10 Nachrichten werden gelesen und zurückgegeben. Alle Spalten werden zurückgegeben.

```
select * from table (DB2XML.MQREADALLXML()) t where t.CORRELID = '1234'
```

Beispiel 4: Die Nachrichten, die durch den Standardservice DB2.DEFAULT.SERVICE angegeben sind, werden gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Alle Spalten werden zurückgegeben.

```
select * from table (DB2XML.MQREADALLXML(10)) t
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

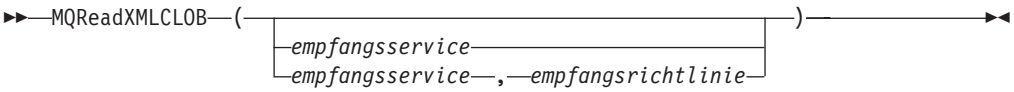
- „Syntaxdiagramme lesen" auf Seite ix

Funktion MQReadXMLCLOB

Zweck:

Die Funktion MQREADXMLCLOB gibt XMLCLOB-Daten aus dem MQSeries-Standort zurück, der durch *empfangsservice* angegeben ist, wobei die Eigenschaften der Servicerichtlinie *empfangsrichtlinie* verwendet werden. Durch die Ausführung dieser Operation wird keine Nachricht aus der Warteschlange entfernt, die dem *empfangsservice* zugeordnet ist. Die Nachricht am Kopf der Warteschlange wird zurückgegeben. Der Rückgabewert ist ein XMLCLOB mit den Nachrichten. Wenn keine Nachrichten verfügbar sind, wird NULL zurückgegeben.

Syntax:



Parameter:

Tabelle 63. Parameter für MQReadXMLCLOB

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht empfangen werden soll. Wenn *empfangsservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.

Tabelle 63. Parameter für MQReadXMLCLOB (Forts.)

Parameter	Datentyp	Beschreibung
empfangsrichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet wird. Wenn die *empfangsrichtlinie* angegeben ist, bezieht sie sich auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Eine Servicerichtlinie definiert eine Gruppe von Serviceeigenschaftsoptionen, die für Nachrichtenübertragungsoperationen angewendet werden. Zu diesen Optionen gehören die Nachrichtenpriorität und die Nachrichtenpermanenz. Wenn die *empfangsrichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.

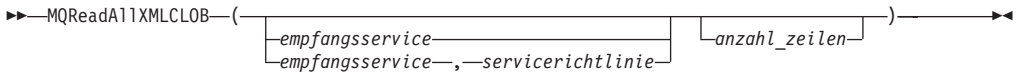
Ergebnisse:
Wenn eine Nachricht in der Warteschlange erfolgreich gelesen wurde, gibt MQREADXMLCLOB ein db2xml.xmlclob zurück. NULL wird zurückgegeben, wenn keine Nachrichten verfügbar sind.

Funktion MQReadAllXMLCLOB

Zweck:

Die Funktion MQReadAllXMLCLOB gibt eine Tabelle mit den Nachrichten und Nachrichtenmetadaten aus dem MQSeries-Standort zurück, der durch *empfangsservice* angegeben ist. Dabei werden die Eigenschaften der Servicerichtlinie *empfangsrichtlinie* verwendet. Durch die Ausführung dieser Operation werden keine Nachrichten aus der Warteschlange entfernt, die *empfangsservice* zugeordnet ist. Falls *anzahl_zeilen* angegeben ist, wird maximal diese Anzahl an Zeilen zurückgegeben. Wenn 'anzahl_zeilen' nicht angegeben ist, werden alle Nachrichten zurückgegeben.

Syntax:



Parameter:

Tabelle 64. Parameter für MQReadAllXMLCLOB

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht gelesen werden soll. Wenn der *empfangsservice* angegeben ist, muss er sich auf einen Service beziehen, der in der Repository-Datei AMT.XML definiert ist. Wenn jedoch *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.
servicerichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet wird. Wenn die *servicerichtlinie* angegeben ist, bezieht sie sich auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Die maximale Größe von *servicerichtlinie* ist 48 Byte.
anzahl_zeilen	INTEGER	Eine positive ganze Zahl mit der maximalen Anzahl an Nachrichten, die durch die Funktion zurückgegeben werden sollen.

Ergebnisse:

Die Funktion MQReadAllXMLCLOB gibt eine Tabelle mit Nachrichten und Nachrichtenmetadaten zurück, die im Folgenden beschrieben werden.

Tabelle 65. Tabelle der Ergebnismenge von MQReadAllXMLCLOB

Spaltenname	Datentyp	Beschreibung
MSG	XMLCLOB	Der Inhalt der MQSeries-Nachricht, bis zu 1 MB lang.
CORRELID	VARCHAR(24)	Eine Korrelations-ID, die verwendet werden kann, um Bezug auf Nachrichten zu nehmen.

Tabelle 65. Tabelle der Ergebnismenge von MQReadAllXMLCLOB (Forts.)

Spaltenname	Datentyp	Beschreibung
TOPIC	VARCHAR(40)	Das Stichwort, unter dem die Nachricht veröffentlicht wurde, falls verfügbar.
QNAME	VARCHAR(48)	Der Name der Warteschlange, in der die Nachricht empfangen wurde.
MSGID	VARCHAR(24)	Die von MQSeries zugeordnete eindeutige Kennung für diese Nachricht.
MSGFORMAT	VARCHAR(8)	Das von MQSeries definierte Format der Nachricht. Normale Zeichenfolgen haben das Format MQSTR.

Beispiel 1: Alle Nachrichten aus der Warteschlange, die durch den Standardservice DB2.DEFAULT.SERVICE angegeben sind, werden gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Die Nachrichten und alle Metadaten werden im Tabellenformat zurückgegeben.

```
select * from table (DB2XML.MQREADALLXMLCLOB()) t
```

Beispiel 2: Die Nachrichten aus dem Kopf der Warteschlange, die durch den Service MYSERVICE angegeben sind, werden gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Nur die Spalten *msg* und *correlid* werden zurückgegeben.

```
select t.MSG, t.CORRELID
from table (DB2XML.MQREADALLXMLCLOB('MYSERVICE')) t
```

Beispiel 3: Der Kopf der Warteschlange, der durch den Standardservice DB2.DEFAULT.SERVICE angegeben ist, wird gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Nur Nachrichten mit einer *CORRELID* von '1234' werden zurückgegeben. Alle Spalten werden zurückgegeben.

```
select *
from table (DB2XML.MQREADALLXMLCLOB()) t where t.CORRELID = '1234'
```

Beispiel 4: Die ersten 10 Nachrichten aus der Warteschlange, die durch den Standardservice DB2.DEFAULT.SERVICE angegeben sind, werden gelesen, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird. Alle Spalten werden zurückgegeben.

```
select * from table (DB2XML.MQREADALLXMLCLOB(10)) t
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

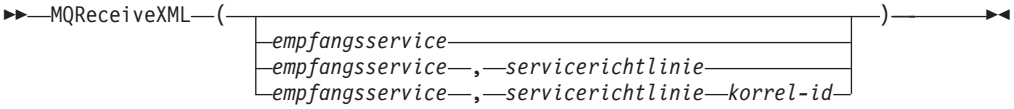
- „Syntaxdiagramme lesen" auf Seite ix

Funktion MQReceiveXML

Zweck:

Die Funktion MQReceiveXML entfernt eine Nachricht, die *empfangsservice* zugeordnet ist, aus der Warteschlange. Die Funktion gibt XMLVARCHAR-Daten aus dem MQSeries-Standort zurück, der durch die Funktion *empfangsservice* angegeben ist, die die Eigenschaften der *servicerichtlinie* verwendet.

Syntax:



Parameter:

Tabelle 66. Parameter für MQReceiveXML

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht empfangen werden soll. Wenn der *empfangsservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.
servicerichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet wird. Wenn die *servicerichtlinie* angegeben ist, muss sie sich auf eine Richtlinie beziehen, die in der Repository-Datei AMT.XML definiert ist. Wenn die *servicerichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *servicerichtlinie* ist 48 Byte.

Tabelle 66. Parameter für MQReceiveXML (Forts.)

Parameter	Datentyp	Beschreibung
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die dieser Nachricht zugeordnet werden soll. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht angegeben ist, wird keine Korrelations-ID gezeigt. Die maximale Größe Größe von *korrel-id* ist 24 Byte.

Ergebnisse:
MQReceiveXML-Funktionen geben einen db2xml.XMLVARCHAR zurück, wenn die Nachrichten erfolgreich aus der Warteschlange empfangen wurden. Die maximale Nachrichtenlänge ist 4000 Byte. NULL wird zurückgegeben, wenn keine Nachrichten verfügbar sind. Wenn die *korrel-id* angegeben ist, wird die erste Nachricht mit einer übereinstimmenden Korrelationskennung zurückgegeben. Wenn *korrel-id* nicht angegeben ist, wird die Nachricht am Kopf der Warteschlange zurückgegeben. Die Nachricht wird aus der Warteschlange entfernt.
Beispiele:
Beispiel 1: Dieses Beispiel empfängt die Nachricht am Kopf der Warteschlange, die durch den Standardservice DB2.DEFAULT.SERVICE angegeben ist, wobei die Standardrichtlinie DB2.DEFAULT.POLICY verwendet wird.

```
values db2xml.MQRECEIVEXML()
```

Wenn dieses Beispiel erfolgreich ausgeführt wird, gibt es den Inhalt einer Nachricht als XMLVARCHAR zurück. Wenn keine Nachrichten verfügbar sind, wird NULL zurückgegeben.

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

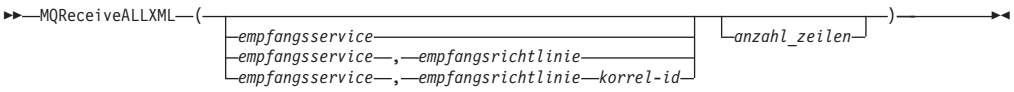
- „Syntaxdiagramme lesen" auf Seite ix

Funktion MQReceiveAllXML

Zweck:

Die Funktion MQReceiveAllXML entfernt Nachrichten, die *empfangsservice* zugeordnet sind, aus der Warteschlange. Wenn *korrel-id* angegeben ist, werden nur die Nachrichten mit einer übereinstimmenden Korrelations-ID zurückgegeben. Wenn *korrel-id* nicht angegeben ist, wird die Nachricht am Kopf der Warteschlange zurückgegeben. Wenn *anzahl_zeilen* angegeben ist, wird maximal die Anzahl *anzahl_zeilen* an Nachrichten zurückgegeben. Wenn diese Angabe fehlt, werden alle verfügbaren Nachrichten zurückgegeben.

Syntax:



Parameter:

Tabelle 67. Parameter für MQReceiveAllXML

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn *empfangsservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei ATM.XML definiert ist. Wenn *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.
empfangsrichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet wird. Wenn die *empfangsrichtlinie* angegeben ist, muss sie sich auf eine Richtlinie beziehen, die in der Repository-Datei AMT.XML definiert ist. Wenn die *empfangsrichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *empfangsrichtlinie* ist 48 Byte.
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die dieser Nachricht zugeordnet werden soll. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht angegeben ist, wird keine Korrelations-ID gezeigt. Die maximale Größe Größe von *korrel-id* ist 24 Byte.
anzahl_zeilen	INTEGER	Eine positive ganze Zahl mit der maximalen Anzahl an Nachrichten, die durch die Funktion zurückgegeben wird.

Ergebnisse:
Wenn eine Tabelle mit Nachrichten erfolgreich von der Warteschlange empfangen wurde, gibt MQRECEIVEXML einen Wert db2xml.xmlvarchar zurück. NULL wird zurückgegeben, wenn keine Nachrichten verfügbar sind. Die Nachrichten werden als Nachrichtentabelle mit Metadaten zurückgegeben.

Spaltenname	Datentyp	Beschreibung
MSG	XMLVARCHAR	Der Inhalt der MQSeries-Nachricht.
CORRELID	VARCHAR(24)	Eine Korrelations-ID, die verwendet werden kann, um Bezug auf Nachrichten zu nehmen.
TOPIC	VARCHAR(40)	Das Stichwort, unter dem die Nachricht veröffentlicht wurde, falls verfügbar.
QNAME	VARCHAR(48)	Der Name der Warteschlange, in der die Nachricht empfangen wurde.
MSGID	CHAR(24)	Die von MQSeries zugeordnete eindeutige Kennung für diese Nachricht.
MSGFORMAT	VARCHAR(8)	Das von MQSeries definierte Format der Nachricht. Normale Zeichenfolgen haben das Format MQSTR.

Beispiele:
Beispiel 1: Alle Nachrichten, die aus der Warteschlange empfangen wurden, sind durch den Standardservice (DB2.DEFAULT.SERVICE) angegeben, wobei die Standardrichtlinie (DB2.DEFAULT.POLICY) verwendet wird. Die Nachrichten und alle Metadaten werden als Tabelle zurückgegeben.

```
select * from table (MQRECEIVEALLXML()) t
```

Beispiel 2: Alle Nachrichten werden aus dem Kopf der Warteschlange empfangen und sind durch den Service MYSERVICE angegeben, wobei die Standardrichtlinie (DB2.DEFAULT.POLICY) verwendet wird. Nur die Spalten MSG und CORRELID werden zurückgegeben. Die Nachrichten liegen in Form einer Tabelle vor, aus der Sie die gewünschten Felder auswählen können.

```
select t.MSG, t.CORRELID from table (MQRECEIVEALLXML('MYSERVICE')) t
```

Beispiel 3: Alle Nachrichten, die aus dem Kopf der Warteschlange empfangen wurden, sind durch den Service MYSERVICE angegeben, wobei die Richtlinie MYPOLICY verwendet wird, die mit der ID '1234' übereinstimmt. Nur die Spalten MSG und CORRELID werden zurückgegeben.

```
select t.MSG, t.CORRELID from table
(MQRECEIVEALLXML('MYSERVICE','MYPOLICY','1234')) t
```

Beispiel 4: Die ersten 10 Nachrichten werden aus dem Kopf der Warteschlange empfangen. Sie sind durch den Standardservice (DB2.DEFAULT.SERVICE) angegeben, wobei die Standardrichtlinie (DB2.DEFAULT.POLICY) verwendet wird. Alle Spalten werden zurückgegeben.

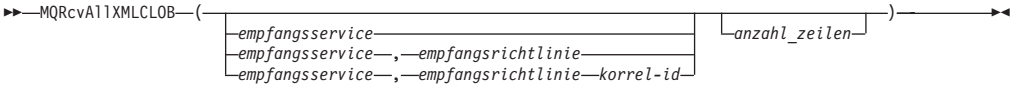
```
 select * from table (MQRECEIVEALLXML(10)) t
```

Funktion MQRcvAllXMLCLOB

Zweck:

Die Funktion MQRcvAllXMLCLOB entfernt die Nachrichten aus der Warteschlange, die *empfangsservice* zugeordnet ist. Wenn *korrel-id* angegeben ist, werden nur die Nachrichten mit einer übereinstimmenden Korrelations-ID zurückgegeben. Wenn *korrel-id* nicht angegeben ist, werden alle Nachrichten zurückgegeben. Wenn *anzahl_zeilen* angegeben ist, wird maximal die Anzahl *anzahl_zeilen* an Nachrichten als XMLCLOB zurückgegeben. Wenn diese Angabe fehlt, werden alle verfügbaren Nachrichten zurückgegeben.

Syntax:



Parameter:

Tabelle 68. Parameter für MQRcvAllXMLCLOB

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht empfangen werden soll. Wenn *empfangsservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.
empfangsrichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet wird. Wenn die *empfangsrichtlinie* angegeben ist, muss sie sich auf eine Richtlinie beziehen, die in der Repository-Datei AMT.XML definiert ist. Wenn die *empfangsrichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *empfangsrichtlinie* ist 48 Byte.

Tabelle 68. Parameter für MQRcvAllXMLCLOB (Forts.)

Parameter	Datentyp	Beschreibung
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die dieser Nachricht zugeordnet werden soll. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht angegeben ist, wird keine Korrelations-ID gezeigt. Die maximale Größe Größe von *korrel-id* ist 24 Byte.
anzahl_zeilen	INTEGER	Eine positive ganze Zahl mit der maximalen Anzahl an Nachrichten, die durch die Funktion zurückgegeben wird.

Ergebnisse:
Wenn eine Nachricht erfolgreich von der Warteschlange empfangen wurde, gibt MQRcvAllXMLCLOB einen XMLCLOB zurück. NULL wird zurückgegeben, wenn keine Nachrichten verfügbar sind. Die Nachrichten werden in einer Tabelle zurückgegeben, die im Folgenden beschrieben wird.

Tabelle 69. Tabelle der Ergebnismenge von MQRcvAllXML

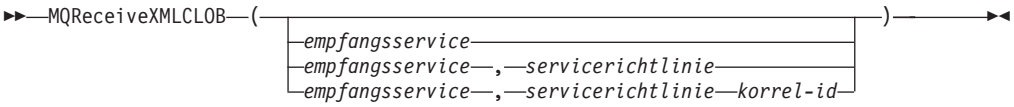
Spaltenname	Datentyp	Beschreibung
MSG	XMLCLOB	Der Inhalt der MQSeries-Nachricht.
CORRELID	VARCHAR(24)	Eine Korrelations-ID, die verwendet werden kann, um Bezug auf Nachrichten zu nehmen.
TOPIC	VARCHAR(40)	Das Stichwort, unter dem die Nachricht veröffentlicht wurde, falls verfügbar.
QNAME	VARCHAR(48)	Der Name der Warteschlange, in der die Nachricht empfangen wurde.
MSGID	CHAR(24)	Die von MQSeries zugeordnete eindeutige Kennung für diese Nachricht.
MSGFORMAT	VARCHAR(8)	Das von MQSeries definierte Format der Nachricht. Normale Zeichenfolgen haben das Format MQSTR.

Funktion MQReceiveXMLCLOB

Zweck:

Die Funktion MQReceiveXMLCLOB entfernt Nachrichten, die *empfangsservice* zugeordnet sind, aus der Warteschlange. Die Funktion gibt XMLVARCHAR-Daten aus dem MQSeries-Standort zurück, der durch den *empfangsservice* angegeben ist, der die Eigenschaften der *servicerichtlinie* verwendet.

Syntax:



Parameter:

Tabelle 70. Parameter für MQReceiveXMLCLOB

Parameter	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht empfangen werden soll. Wenn der *empfangsservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn jedoch *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.
servicerichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet werden soll. Wenn die *servicerichtlinie* angegeben ist, muss sie sich auf eine Richtlinie beziehen, die in der Repository-Datei AMT.XML definiert ist. Wenn *servicerichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *servicerichtlinie* ist 48 Byte.

Tabelle 70. Parameter für MQReceiveXMLCLOB (Forts.)

Parameter	Datentyp	Beschreibung
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die dieser Nachricht zugeordnet werden soll. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht angegeben ist, wird keine Korrelations-ID gezeigt. Die maximale Größe Größe von *korrel-id* ist 24 Byte.

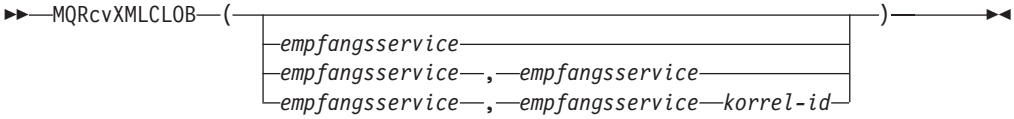
Ergebnisse:
MQReceiveXMLCLOB-Funktionen geben einen db2xml.XMLCLOB zurück, wenn Nachrichten erfolgreich aus der Warteschlange empfangen wurden. NULL wird zurückgegeben, wenn keine Nachrichten verfügbar sind. Wenn die *korrel-id* angegeben ist, wird die erste Nachricht mit einer übereinstimmenden Korrelationskennung zurückgegeben. Wenn *korrel-id* jedoch nicht angegeben ist, wird die Nachricht am Kopf der Warteschlange zurückgegeben.

Funktion MQRcvXMLCLOB

Zweck:

Die Funktion MQRcvXMLCLOB entfernt Nachrichten, die *empfangsservice* zugeordnet sind, aus der Warteschlange. Die Funktion gibt XMLVARCHAR-Daten aus dem MQSeries-Standort zurück, der durch die Funktion *empfangsservice* angegeben ist, die die Eigenschaften des *empfangsservice* verwendet.

Syntax:



Parameter:

Tabelle 71. Parameter für MQRcvXMLCLOB

Datentyp	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, von der die Nachricht empfangen werden soll. Wenn der *empfangsservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn jedoch *empfangsservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.

Tabelle 71. Parameter für MQRcvXMLCLOB (Forts.)

Datentyp	Datentyp	Beschreibung
empfangsservice	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung dieser Nachricht verwendet werden soll. Wenn der *empfangsservice* angegeben ist, muss er sich auf eine Richtlinie beziehen, die in der Repository-Datei AMT.XML definiert ist. Wenn *empfangsrichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *empfangsservice* ist 48 Byte.
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die dieser Nachricht zugeordnet werden soll. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht entworfen ist, wird keine Korrelations-ID angegeben. Die maximale Größe Größe von *korrel-id* ist 24 Byte.

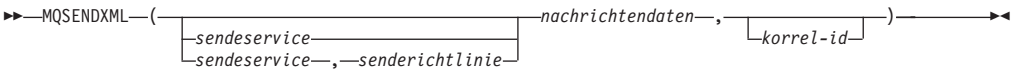
Ergebnisse:
MQRcvMLCLOB-Funktionen geben einen db2xml.XMLCLOB zurück, wenn Nachrichten erfolgreich aus der Warteschlange empfangen wurden. Die maximale Nachrichtenlänge ist 1 MB. NULL wird zurückgegeben, wenn keine Nachrichten verfügbar sind. Wenn die *korrel-id* angegeben ist, wird die erste Nachricht mit einer übereinstimmenden Korrelationskennung zurückgegeben. Wenn *korrel-id* jedoch nicht angegeben ist, wird die Nachricht am Kopf der Warteschlange zurückgegeben.

Funktion MQSENDXML

Zweck:

Die Funktion MQSENDXML sendet die Daten, die in *nachrichtendaten* enthalten sind, an den MQSeries-Standort, der durch *sendeservice* angegeben ist, wobei die *senderichtlinie* verwendet wird. Außerdem kann eine optionale benutzerdefinierte Nachrichtenkorrelations-ID durch *korrel-id* angegeben werden. Die Funktion gibt im Erfolgsfall '1' zurück.

Syntax:



Parameter:

Tabelle 72. Parameter für MQSendXML

Parameter	Datentyp	Beschreibung
nachrichtendaten	XMLVARCHAR oder XMLCLOB	Ein Ausdruck mit den Daten, die über MQSeries gesendet werden.
sendeservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *sendeservice* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *sendeservice* nicht angegeben ist. Die maximale Größe von *sendeservice* ist 48 Byte.
senderichtlinie	VARCHAR(48)	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung der Nachricht verwendet wird. Wenn angegeben, bezieht sich die *senderichtlinie* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn die *senderichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *senderichtlinie* ist 48 Byte.
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die der Nachricht zugeordnet ist. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht angegeben ist, wird keine Korrelations-ID gezeigt. Die maximale Größe Größe von *korrel-id* ist 24 Byte.

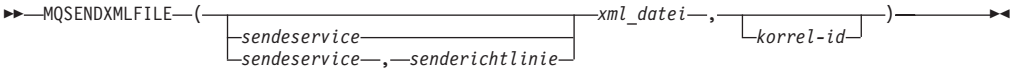
Ergebnisse:
Eine erfolgreiche Nachricht führt zu einem Rückgabewert 1. Eine Nachricht mit *nachrichtendaten* wird an den Standort gesendet, der durch *sendeservice* angegeben ist, wobei die Richtlinie verwendet wird, die durch *senderichtlinie* definiert ist.

Funktion MQSENDXMLFILE

Zweck:

Die Funktion MQSENDXMLFILE sendet die Daten, die in *xml_datei* enthalten sind, an den MQSeries-Standort, der durch *sendeservice* angegeben ist, wobei die Eigenschaften der *servicerichtlinie* verwendet werden. Eine optionale benutzerdefinierte Nachrichtenkorrelations-ID kann durch *korrel-id* angegeben werden. Die Funktion gibt im Erfolgsfall '1' zurück.

Syntax:



Parameter:

Tabelle 73. Parameter für MQSENDXMLFILE

Parameter	Datentyp	Beschreibung
xml_datei	XMLCLOB	Der Name einer XML-Datei mit einer maximalen Größe von 80 Byte. Die Datei enthält die Daten, die über MQSeries gesendet werden sollen.
sendeservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn *sendeservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn *sendeservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *sendeservice* ist 48 Byte.
senderichtlinie	VARCHAR(48)	Eine Zeichenfolge mit dem MQSeries-AMI-Service, der für die Bearbeitung dieser Nachricht verwendet wird. Wenn angegeben, bezieht sich die *senderichtlinie* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn die *senderichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *senderichtlinie* ist 48 Byte.

Tabelle 73. Parameter für MQSENDXMLFILE (Forts.)

Parameter	Datentyp	Beschreibung
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die dieser Nachricht zugeordnet werden soll. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht angegeben ist, wird keine Korrelations-ID aufgelistet. Die maximale Größe Größe von *korrel-id* ist 24 Byte.

Ergebnisse:
Wenn die Funktion erfolgreich ist, wird eine '1' zurückgegeben. Als Nebeneffekt der erfolgreichen Ausführung dieser Funktion wird eine Nachricht mit den *nachrichtendaten* an den Standort gesendet, der durch *sendeservice* angegeben ist, wobei die Richtlinie verwendet wird, die durch *senderichtlinie* definiert ist.
Beispiele:
Beispiel 1: XML-Dokumente, die in der Datei "c:\xml\test1.xml" enthalten sind, werden an den Standardservice (DB2.DEFAULT.SERVICE) gesendet, wobei die Standardrichtlinie (DB2.DEFAULT.POLICY) ohne Korrelations-ID verwendet wird.

```
Values MQSENDXMLFILE('c:\xml\test1.xml');
```

Dieses Beispiel gibt im Erfolgsfall den Wert '1' zurück.

Beispiel 2: XML-Dokumente, die in der Datei `c:\xml\test2.xml` enthalten sind, werden an den Service MYSERVICE gesendet, wobei die Richtlinie MYPOLICY ohne Korrelations-ID verwendet wird.

```
Values MQSENDXMLFILE('MYSERVICE', 'MYPOLICY', 'c:\xml\test2.xml');
```

Dieses Beispiel gibt im Erfolgsfall den Wert '1' zurück.

Beispiel 3: XML-Dokumente, die in der Datei "c:\xml\test3.xml" enthalten sind, werden an den Service MYSERVICE gesendet, wobei die Richtlinie MYPOLICY mit der Korrelations-ID "Test3" verwendet wird.

```
Values MQSENDXML('MYSERVICE','MYPOLICY', 'c:\xml\test3.xml', 'Test3');
```

Dieses Beispiel gibt im Erfolgsfall den Wert '1' zurück.

Beispiel 4: XML-Dokumente, die in der Datei "c:\xml\test4.xml" enthalten sind, werden an den Service MYSERVICE gesendet, wobei die Standardrichtlinie (DB2.DEFAULT.POLICY) ohne Korrelations-ID verwendet wird.

```
Values MQSENDXMLFILE('MYSERVICE', 'c:\xml\test4.xml');
```

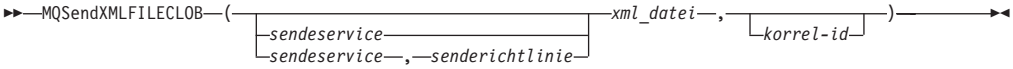
Dieses Beispiel gibt im Erfolgsfall den Wert '1' zurück.

Funktion MQSendXMLFILECLOB

Zweck:

Die Funktion MQSendXMLFILECLOB sendet die Daten, die in *xml_datei* enthalten sind, an den MQSeries-Standort, der durch *sendeservice* angegeben ist, wobei die Eigenschaften der *senderichtlinie* verwendet werden. Der gesendete Datentyp ist XMLCLOB. Eine optionale benutzerdefinierte Nachrichtenkorrelations-ID kann durch *korrel-id* angegeben werden. Die Funktion gibt im Erfolgsfall '1' zurück.

Syntax:



Parameter:

Tabelle 74. Parameter für MQSendXMLFILECLOB

Parameter	Datentyp	Beschreibung
xml_datei	XMLCLOB	Der Name einer XML-Datei mit einer maximalen Größe von 80 Byte. Die Datei enthält die Daten, die über MQSeries gesendet werden sollen.
sendeservice	VARCHAR(48)	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn *sendeservice* angegeben ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Wenn *sendeservice* nicht angegeben ist, wird DB2.DEFAULT.SERVICE verwendet. Die maximale Größe von *sendeservice* ist 48 Byte.
senderichtlinie	VARCHAR(48)	Eine Zeichenfolge mit dem MQSeries-AMI-Service, der für die Bearbeitung dieser Nachricht verwendet wird. Wenn angegeben, bezieht sich die *senderichtlinie* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn die *senderichtlinie* nicht angegeben ist, wird die DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *senderichtlinie* ist 48 Byte.

Tabelle 74. Parameter für MQSendXMLFILECLOB (Forts.)

Parameter	Datentyp	Beschreibung
korrel-id	VARCHAR(24)	Eine Zeichenfolge mit der optionalen Korrelations-ID, die dieser Nachricht zugeordnet werden soll. Die *korrel-id* wird oft in Request/Reply-Szenarios angegeben, um Anforderungen und Antworten zuzuordnen. Wenn sie nicht angegeben ist, wird keine Korrelations-ID aufgelistet. Die maximale Größe Größe von *korrel-id* ist 24 Byte.

Ergebnisse:
Wenn die Funktion erfolgreich ist, wird eine '1' zurückgegeben. Als Nebeneffekt der erfolgreichen Ausführung dieser Funktion wird eine Nachricht mit den *nachrichtendaten* an den Standort gesendet, der durch *sendeservice* angegeben ist, wobei die Richtlinie verwendet wird, die durch *senderichtlinie* definiert ist.

Gespeicherte Prozeduren von XML Extender für MQSeries

Übersicht: Gespeicherte Prozeduren von XML Extender für MQSeries

Gespeicherte Prozeduren für Zusammensetzung

Verwenden Sie die gespeicherten Prozeduren für Zusammensetzung dxxmqGen(), dxxmqGenCLOB(), dxxmqRetrieve() und dxxmqRetrieveCLOB(), um XML-Dokumente aus Daten in bestehenden Datenbanktabellen zu generieren und um die generierten XML-Dokumente an eine Nachrichtenwarteschlange zu senden. Die gespeicherten Prozeduren dxxmqGen() und dxxmqGenCLOB() verwenden eine DAD-Datei als Eingabe. Für sie sind keine aktivierten XML-Objektgruppen erforderlich. Die gespeicherten Prozeduren dxxmqRetrieve und dxxmqRetrieveCLOB verwenden Objektgruppennamen als Eingabe.

Gespeicherte Prozeduren für Zerlegung

Mit den gespeicherten Prozeduren für Zerlegung dxxmqInsert(), dxxmqInsertAll(), dxxInsertCLOB(), dxxmqShred(), dxxmqShredCLOB und dxxmqShredAll() werden ankommende XML-Dokumente aus einer Nachrichtenwarteschlange zerlegt und die Daten in neuen oder bestehenden Datenbanktabellen gespeichert.

Die gespeicherten Prozeduren dxxmqInsert(), dxxmqInsertAll(), dxxmqInsertAllCLOB() und dxxInsertCLOB() verwenden den Namen einer aktivierten XML-Objektgruppe als Eingabe.

Die gespeicherten Prozeduren dxxmqShred(), dxxmqShredAll(), dxxmqShredCLOB und dxxmqShredAllCLOB verwenden eine DAD-Datei als Eingabe. Für sie ist keine aktivierte XML-Objektgruppe erforderlich.

Die nachfolgende Tabelle fasst die unterschiedlichen gespeicherten Prozeduren mit Erläuterung ihrer Funktionen zusammen.

Tabelle 75. Gespeicherte XML-Prozeduren für MQSeries®

Funktion	Zweck
dxxmqGen	Rufen Sie die gespeicherte Prozedur dxxmqGen auf, um XML-Dokumente zusammenzusetzen, wobei eine DAD-Datei als Eingabeparameter verwendet wird. Die resultierende Dokumentart ist XMLVARCHAR(4000).
dxxmqGenCLOB	Erstellt ein XML-Dokument aus Daten, die in den in der DAD-Datei angegebenen XML-Objektgruppentabellen gespeichert sind, und sendet das XML-Dokument an eine MQ-Nachrichtenwarteschlange. Die resultierende Dokumentart ist XMLCLOB(1M).
dxxmqRetrieve	Rufen Sie die gespeicherte Prozedur dxxmqRetrieve auf, um XML-Dokumente zusammenzusetzen, wobei ein Objektgruppenname als Eingabeparameter verwendet wird. Die resultierende Dokumentart ist XMLVARCHAR(4000).
dxxmqRetrieveCLOB	Rufen Sie die gespeicherte Prozedur dxxmqRetrieve auf, um XML-Dokumente zusammenzusetzen, wobei ein Objektgruppenname als Eingabeparameter verwendet wird. Die resultierende Dokumentart ist XMLCLOB(1M).
dxxmqShred	Rufen Sie die gespeicherte Prozedur dxxmqShred auf, um ein XML-Dokument zu zerlegen, wobei eine DAD-Datei als Eingabeparameter verwendet wird. Die resultierende Dokumentart ist XMLVARCHAR(4000).
dxxmqShredAll	Rufen Sie die gespeicherte Prozedur dxxmqShredAll auf, um mehrere XML-Dokumente zu zerlegen, wobei eine DAD-Datei als Eingabeparameter verwendet wird. Die resultierende Dokumentart ist XMLVARCHAR(4000).
dxxmqShredCLOB	Zerlegt auf der Basis einer DAD-Dateizuordnung ein ankommendes XML-Dokument aus einer Nachrichtenwarteschlange und speichert den Inhalt der XML-Elemente und -Attribute in den angegebenen DB2® UDB-Tabellen. Die resultierende Dokumentart ist XMLCLOB(1M).
dxxmqShredAllCLOB	Zerlegt auf der Basis einer DAD-Dateizuordnung ein ankommendes XML-Dokument aus einer Nachrichtenwarteschlange und speichert den Inhalt der XML-Elemente und -Attribute in den angegebenen DB2 UDB-Tabellen. Die resultierende Dokumentart ist XMLCLOB(1M).

Tabelle 75. Gespeicherte XML-Prozeduren für MQSeries® (Forts.)

Funktion	Zweck
dxxmqInsert	Rufen Sie die gespeicherte Prozedur dxxmqInsert auf, um ein XML-Dokument zu zerlegen, wobei ein Objektgruppenname als Eingabeparameter verwendet wird. Die resultierende Dokumentart ist XMLVARCHAR(4000).
dxxmqInsertAll	Rufen Sie die gespeicherte Prozedur dxxmqInsertAll auf, um mehrere XML-Dokumente zu zerlegen, wobei ein Objektgruppenname als Eingabeparameter verwendet wird. Die resultierende Dokumentart ist XMLVARCHAR(4000).
dxxmqInsertCLOB	Zerlegt ein ankommendes XML-Dokument aus der Nachrichtenwarteschlange und speichert die Daten in neuen oder vorhandenen Datenbanktabellen. Die resultierende Dokumentart ist XMLCLOB(1M).
dxxmqInsertAllCLOB	Zerlegt alle ankommenden XML-Dokumente aus der Nachrichtenwarteschlange und speichert die Daten in neuen oder vorhandenen Datenbanktabellen. dxxmqInsertAllCLOB verwendet an Stelle eines DAD-Dateinamens einen Objektgruppennamen, um zu bestimmen, wie die Daten gespeichert werden sollen. Die resultierende Dokumentart ist XMLCLOB(1M).

Zugehörige Referenzen:

- „Gespeicherte Prozedur dxxmqGenCLOB" auf Seite 260
- „Gespeicherte Prozedur dxxmqRetrieve" auf Seite 262
- „Gespeicherte Prozedur dxxmqRetrieveCLOB" auf Seite 265
- „Gespeicherte Prozedur dxxmqShred" auf Seite 267
- „Gespeicherte Prozedur dxxmqShredAll" auf Seite 269
- „Gespeicherte Prozedur dxxmqShredCLOB" auf Seite 271
- „Gespeicherte Prozedur dxxmqInsert" auf Seite 273
- „Gespeicherte Prozedur dxxmqInsertAll" auf Seite 277
- „Gespeicherte Prozedur dxxmqInsertCLOB" auf Seite 275
- „Gespeicherte Prozedur dxxmqGen()" auf Seite 257
- „Gespeicherte Prozedur dxxmqShredAllCLOB" auf Seite 272
- „Gespeicherte Prozedur dxxmqInsertAllCLOB" auf Seite 279

Gespeicherte Prozedur dxxmqGen()

Zweck:

Erstellt ein XML-Dokument aus Daten, die in den in der DAD-Datei angegebenen XML-Objektgruppentabellen gespeichert sind, und sendet das XML-Dokument an eine MQ-Nachrichtenwarteschlange. Die gespeicherte Prozedur gibt eine Zeichenfolge zurück, um den Status der gespeicherten Prozedur anzugeben.

Zur Unterstützung dynamischer Abfragen verwendet dxxmqGen() 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.

Syntax:

```
dxxmqGen(varchar(48)    serviceName,     /*Eingabe*/
         varchar(48)    policyName,      /*Eingabe*/
         varchar(80)    dadFileName,     /*Eingabe*/
         integer        overrideType,    /* Eingabe */
         varchar(1024)  override,        /* Eingabe */
         integer        maxRows,         /* Eingabe */
         integer        numRows,         /* Ausgabe */
         char(20)       status)          /*Ausgabe*/
```

Parameter:

Tabelle 76. Parameter für dxxmqGen()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
dadFileName	Der Name der DAD-Datei.	IN
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

Tabelle 76. Parameter für dxxmqGen() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
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 an generierten Nachrichten in der Nachrichtenwarteschlange.	IN
numRows	Die tatsächliche Anzahl an generierten Zeilen in der Nachrichtenwarteschlange.	OUT
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Das folgende Beispielfragment generiert ein XML-Dokument und sendet es an die Warteschlange. Dabei wird davon ausgegangen, dass ein MQ/AMI-Service, *myService*, und eine Richtlinie, *myPolicy*, in der Repository-Datei definiert wurden. Diese Datei speichert die Repository-Definitionen im XML-Format.

```
#include "dxx.h"
#include "dxxrc.h"
              EXEC SQL INCLUDE SQLCA;
              EXEC SQL BEGIN DECLARE SECTION;
              char           serviceName[48];      /* Name des MQ/AMI-Services*/
              char           policyName[48];       /* Name der MQ/AMI-Richtlinie*/
              char           dadFileName[80];      /* Name der DAD-Datei */
              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 */
              char           status[20]        /* Statuscode oder Nachricht */
              short          ovtype_ind;
              short   ov_ind;
              short          maxrow_ind;
              short          numrow_ind;
              short          dadFileName_ind;
              short          serviceName_ind;
```

```
short          policyName_ind;
short          status_ind;

EXEC SQL END DECLARE SECTION;
strcpy(dadFileName,"c:\dxx\dad\litem3.dad");
strcpy(serviceName,"myService");
strcpy(policyName,"myPolicy");
override[0] = '\0';
overrideType = NO_OVERRIDE;
max_row = 500;
num_row = 0;
status[0] = '\0';
dadFileName_ind = 0;
serviceName_ind = 0;
policyName_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
ovtype_ind=0;
ov_ind=-1;
status_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxmqGen(:serviceName:serviceName_ind,
       :policyName:policyName_ind,
       :dadFileName:dadFileName_ind,
       :overrideType:ovtype_ind,
       :override:ov_ind,
       :max_row:maxrow_ind,
       :num_row:numrow_ind,
       :status:status_ind);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries” auf Seite 231

Zugehörige Tasks:

- „Gespeicherte Prozeduren von XML Extender aufrufen” auf Seite 205

Zugehörige Referenzen:

- „Syntaxdiagramme lesen” auf Seite ix
- Anhang C, „Grenzwerte für XML Extender”, auf Seite 329

Gespeicherte Prozedur dxxmqGenCLOB

Zweck:

Erstellt ein XML-Dokument aus Daten, die in den in der DAD-Datei angegebenen XML-Objektgruppentabellen gespeichert sind, und sendet das XML-Dokument an eine MQ-Nachrichtenwarteschlange. Die Dokumentart ist XMLCLOB. Die gespeicherte Prozedur gibt eine Zeichenfolge zurück, um den Status der gespeicherten Prozedur anzugeben. Diese gespeicherte Prozedur wird für die Enterprise Server Edition (ESE) nicht unterstützt.

Zur Unterstützung dynamischer Abfragen verwendet dxxmqGenCLOB als 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.

Syntax:

```
dxxmqGenCLOB(varchar(48)    serviceName,     /*Eingabe*/
             varchar(48)    policyName,      /*Eingabe*/
             varchar(80)    dadFileName,     /*Eingabe*/
             integer        overrideType,    /*Eingabe*/
             varchar(1024)  override,        /*Eingabe*/
             integer        maxRows,         /* Eingabe */
             integer        numRows,         /*Ausgabe*/
             char(20)       status)          /*Ausgabe*/
```

Parameter:

Tabelle 77. Parameter für dxxmqGenCLOB

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
dadFileName	Der Name der DAD-Datei.	IN
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

Tabelle 77. Parameter für dxxmqGenCLOB (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
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 an generierten Nachrichten in der Nachrichtenwarteschlange.	IN
numRows	Die tatsächliche Anzahl an generierten Zeilen in der Nachrichtenwarteschlange.	OUT
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqRetrieve

Zweck:

Die gespeicherte Prozedur dxxmqRetrieve() dient als Mittel zum Abrufen zerlegter XML-Dokumente. Als Eingabe erhält dxxmqRetrieve() einen Puffer mit dem Namen einer aktivierten XML-Objektgruppe, dem MQ/AMI-Service und den Richtliniennamen. Die gespeicherte Prozedur sendet das zusammengesetzte XML-Dokument an eine MQ-Warteschlange; sie gibt die Anzahl an Zeilen zurück, die an die Warteschlange gesendet wurden, sowie eine Statusnachricht. Die gespeicherte Prozedur dxxmqRetrieve ermöglicht, dass dieselbe DAD-Datei sowohl für die Zusammensetzung als auch für die Zerlegung verwendet werden kann.

Zur Unterstützung dynamischer Abfragen verwendet dxxmqRetrieve() 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.

Die Anforderungen der DAD-Datei für dxxmqRetrieve() sind die gleichen wie die Anforderungen für dxxmqGen(). Der einzige Unterschied liegt darin, dass die DAD-Datei kein Eingabeparameter für dxxmqRetrieve() ist; stattdessen ist der Name einer aktivierten XML-Objektgruppe der erforderliche Parameter.

Syntax:

```
dxxmqRetrieve(varchar(48)    serviceName,        /*Eingabe*/
              varchar(48)    policyName,         /*Eingabe*/
              varchar(80)    collectionName,     /*Eingabe*/
              integer        overrideType,       /*Eingabe*/
              varchar(1024)  override,           /*Eingabe*/
              integer        maxrows,            /*Eingabe*/
              integer        numrows,            /*Ausgabe*/
              char(20)       status)             /*Ausgabe*/
```

Parameter:

Tabelle 78. Parameter für dxxmqRetrieve()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
collectionName	Der Name einer aktivierten Objektgruppe.	IN
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

Tabelle 78. Parameter für dxxmqRetrieve() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
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. Die maximale Länge ist 1024 Byte. Für die Zeichenfolge *overrideType* ist erforderlich, dass die RDB_node-Zuordnung in der DAD-Datei verwendet wird.	IN
maxRows	Die maximale Anzahl von Zeilen in der Ergebnistabelle.	IN
numRows	Die tatsächliche Anzahl generierter Zeilen in der Ergebnistabelle.	OUT
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Das folgende Fragment zeigt ein Beispiel für einen Aufruf von dxxmqRetrieve().

```
#include "dxx.h"
#include "dxxrc.h"
              EXEC SQL INCLUDE SQLCA;
              EXEC SQL BEGIN DECLARE SECTION;
              char           serviceName[48];      /* Name des MQ/AMI-Services*/
              char           policyName[48];       /* Name der MQ/AMI-Richtlinie*/
              char           collection[32];   /* Name der XML-Objektgruppe */
              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 */
              char        status[20];         /* Statuscode oder Nachricht */
              short     ovtype_ind;
              short     ov_ind;
              short     maxrow_ind;
              short     numrow_ind;
              short     collection_ind;
              short     serviceName_ind;
              short     policyName_ind;
              short     status_ind;

              EXEC SQL END DECLARE SECTION;
```

```
 /* Host-Variable und Anzeiger initialisieren  */
 strcpy(collection,"sales_ord");
 strcpy(serviceName,"myService");
 strcpy(policyName,"myPolicy");
 override[0] = '\0';
 overrideType = NO_OVERRIDE;
 max_row = 500;
 num_row = 0;
 status[0] = '\0';
 serviceName_ind = 0;
 policyName_ind = 0;
 collection_ind = 0;
 maxrow_ind = 0;
 numrow_ind = -1;
 ovtype_ind=0;
 ov_ind=-1;
 status_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxmqRetrieve(:serviceName:serviceName_ind,
       :policyName:policyName_ind,
       :collection:collection_ind,
       :overrideType:ovtype_ind,
       :override:ov_ind,
       :max_row:maxrow_ind,
       :num_row:numrow_ind,
       :status:status_ind);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqRetrieveCLOB

Zweck:

Die gespeicherte Prozedur dxxmqRetrieveCLOB dient als Mittel zum Abrufen zerlegter XML-Dokumente. Als Eingabe erhält dxxmqRetrieveCLOB einen Puffer mit dem Namen einer aktivierten XML-Objektgruppe, dem MQ/AMI-Service und den Richtliniennamen. Die gespeicherte Prozedur sendet das zusammengesetzte XML-Dokument an eine MQ-Warteschlange, und sie gibt die Anzahl an Zeilen zurück, die an die Warteschlange gesendet wurden, sowie eine Statusnachricht. Die gespeicherte Prozedur dxxmqRetrieveCLOB ermöglicht, dass dieselbe DAD-Datei sowohl für die Zusammensetzung als auch für die Zerlegung verwendet werden kann. Diese gespeicherte Prozedur wird für die Enterprise Server Edition (ESE) nicht unterstützt.

Zur Unterstützung dynamischer Abfragen verwendet dxxmqRetrieveCLOB 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.

Die Anforderungen der DAD-Datei für dxxmqRetrieveCLOB sind die gleichen wie die Anforderungen für dxxmqGenCLOB. Der einzige Unterschied liegt darin, dass die DAD-Datei kein Eingabeparameter für dxxmqRetrieveCLOB ist; stattdessen ist der Name einer aktivierten XML-Objektgruppe der erforderliche Parameter.

Syntax:

```
dxxmqRetrieveCLOB(varchar(48)    serviceName,       /*Eingabe*/
                  varchar(48)    policyName,        /*Eingabe*/
                  varchar(80)    collectionName,    /*Eingabe*/
                  integer        overrideType,      /*Eingabe*/
                  varchar(1024)  override,          /*Eingabe*/
                  integer        maxrows,           /*Eingabe*/
                  integer        numrows,           /*Ausgabe*/
                  char(20)       status)            /*Ausgabe*/
```

Parameter:

Tabelle 79. Parameter für dxxmqRetrieveCLOB

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
collectionName	Der Name einer aktivierten Objektgruppe.	IN
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

Tabelle 79. Parameter für dxxmqRetrieveCLOB (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
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. Die maximale Größe ist 1024 Byte. Für die Zeichenfolge *overrideType* ist erforderlich, dass die RDB_node-Zuordnung in der DAD-Datei verwendet wird.	IN
maxRows	Die maximale Anzahl von Zeilen in der Ergebnistabelle.	IN
numRows	Die tatsächliche Anzahl generierter Zeilen in der Ergebnistabelle.	OUT
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqShred

Zweck:

Zerlegt auf der Basis einer DAD-Dateizuordnung ein ankommendes XML-Dokument aus einer Nachrichtenwarteschlange und speichert den Inhalt der XML-Elemente und -Attribute in den angegebenen DB2 UDB-Tabellen.

Damit dxxmqShred() 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.

Für die gespeicherte Prozedur ist erforderlich, dass die in der Verknüpfungsbedingung, in der DAD, angegebenen Spalten den Primär-/Fremdschlüsselbeziehungen in den vorhandenen Tabellen entsprechen. Die im RDB_node des Root-element_node angegebenen Spalten mit den Verknüpfungsbedingungen müssen in den Tabellen vorhanden sein.

Syntax:

```
dxxmqShred(varchar(48)   serviceName,    /* Eingabe */
           varchar(48)   policyName,     /* Eingabe */
           varchar(80)   dadFileName,    /* Eingabe */
           varchar(10)   status)         /* Ausgabe */
```

Parameter:

Tabelle 80. Parameter für dxxmqShred()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
dadFileName	Der Name der DAD-Datei. Die maximale Größe ist 80 Byte.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Das folgende Fragment zeigt ein Beispiel für einen Aufruf von dxxmqShred().

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         char        serviceName[48];    /* Name des MQ/AMI-Services */
         char        policyName[48];     /* Name der MQ/AMI-Richtlinie */
         char        dadFileName[80];    /* Name der DAD-Datei */
         char        status[20];         /* Statuscode oder Nachricht */
         short       serviceName_ind;
         short       policyName_ind;
         short       dadFileName_ind;
         short       status_ind;
         EXEC SQL END DECLARE SECTION;

              /* Host-Variable und Anzeiger initialisieren  */
         strcpy(dadFileName,"e:/dxx/samples/dad/getstart_xcollection.dad");
         strcpy(serviceName, "myService");
         strcpy(policyName, "myPolicy");
         status[0]='\0';
         serviceName_ind=0;
         policyName_ind=0;
         dadFileName_ind=0;
         status_ind=-1;

              /* Gespeicherte Prozedur aufrufen */
         EXEC SQL CALL dxxmqShred(:serviceName:serviceName_ind,
                                  :policyName:policyName_ind,
                                  :dadFileName:dadFileName_ind,
                                  :status:status_ind);
```

Zugehörige Referenzen:

- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqShredAll

Zweck:

Zerlegt auf der Basis einer DAD-Dateizuordnung alle ankommenden XML-Dokumente aus einer Nachrichtenwarteschlange. Der Inhalt der XML-Elemente und -Attribute wird in den angegebenen DB2 UDB-Tabellen gespeichert.

Damit dxxmqShredAll() 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. Für die gespeicherte Prozedur ist erforderlich, dass die in der Verknüpfungsbedingung, in der DAD, angegebenen Spalten den Primär-/Fremdschlüsselbeziehungen in den vorhandenen Tabellen entsprechen. Die im RDB_node des Root-element_node angegebenen Spalten mit den Verknüpfungsbedingungen müssen in den Tabellen vorhanden sein.

Syntax:

```
dxxmqShredAll(varchar(48)   serviceName,    /* Eingabe */
              varchar(48)   policyName,     /* Eingabe */
              varchar(80)   dadFileName,    /* Eingabe */
              varchar(20)   status)         /* Ausgabe */
```

Parameter:

Tabelle 81. Parameter für dxxmqShredAll()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
dadFileName	Der Name der DAD-Datei. Die maximale Größe ist 80 Byte.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Das folgende Fragment zeigt ein Beispiel für einen Aufruf von dxxmqShredAll().

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
      EXEC SQL BEGIN DECLARE SECTION;
        char        serviceName[48];    /* Name des MQ/AMI-Services */
        char        policyName[48];     /* Name der MQ/AMI-Richtlinie */
        char        dadFileName[80];    /* Name der DAD-Datei */
        char        status[20];         /* Statuscode oder Nachricht */
        short       serviceName_ind;
        short       policyName_ind;
        short       dadFileName_ind;
        short       status_ind;
        EXEC SQL END DECLARE SECTION;

              /* Host-Variable und Anzeiger initialisieren  */
        strcpy(dadFileName,"e:/dxx/samples/dad/getstart_xcollection.dad");
        strcpy(serviceName, "myService");
        strcpy(policyName, "myPolicy");
        status[0]=\0;
        serviceName_ind=0;
```

```
policyName_ind=0;
dadFileName_ind=0;
status_ind=-1;

  /* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxmqShredAll(:serviceName:serviceName_ind,
                  :policyName:policyName_ind,
                  :dadFileName:dadFileName_ind,
                  :status:status_ind);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqShredCLOB

Zweck:

Zerlegt auf der Basis einer DAD-Dateizuordnung ein ankommendes XML-Dokument aus einer Nachrichtenwarteschlange und speichert den Inhalt der XML-Elemente und -Attribute in den angegebenen DB2 UDB-Tabellen. Die ankommende Dokumentart ist XMLCLOB.

Für dxxmqShredCLOB müssen alle in der DAD-Datei angegebenen Tabellen vorhanden sein, und alle in der DAD-Datei angegebenen Spalten und Datentypen müssen mit den vorhandenen Tabellen konsistent sein. Für diese gespeicherte Prozedur ist erforderlich, dass die in der Verknüpfungsbedingung der DAD angegebenen Spalten den Primär-/Fremdschlüsselbeziehungen in den vorhandenen Tabellen entsprechen. Die im RDB_node des Root-element_node angegebenen Spalten mit den Verknüpfungsbedingungen müssen in den Tabellen vorhanden sein.

Syntax:

```
dxxmqShredCLOB(varchar(48)   serviceName,    /* Eingabe */
               varchar(48)   policyName,     /* Eingabe */
               varchar(80)   dadFileName,    /* Eingabe */
               varchar(10)   status)         /* Ausgabe */
```

Parameter:

Tabelle 82. Parameter für dxxmqShredCLOB

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN

Tabelle 82. Parameter für dxxmqShredCLOB (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
dadFileName	Der Name der DAD-Datei. Die maximale Größe ist 80 Byte.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqShredAllCLOB

Zweck:

Zerlegt auf der Basis einer DAD-Dateizuordnung ein ankommendes XML-Dokument aus einer Nachrichtenwarteschlange und speichert den Inhalt der XML-Elemente und -Attribute in den angegebenen DB2 UDB-Tabellen.

Für dxxmqShredAllCLOB müssen alle in der DAD-Datei angegebenen Tabellen vorhanden sein, und alle in der DAD-Datei angegebenen Spalten und Datentypen müssen mit den vorhandenen Tabellen konsistent sein. Für diese gespeicherte Prozedur ist erforderlich, dass die in der Verknüpfungsbedingung der DAD angegebenen Spalten den Primär-/Fremdschlüsselbeziehungen in den vorhandenen Tabellen entsprechen. Die im RDB_node des Root-element_node angegebenen Spalten mit den Verknüpfungsbedingungen müssen in den Tabellen vorhanden sein.

Syntax:

```
dxxmqShredCLOB(varchar(48)   serviceName,     /* Eingabe */
               varchar(48)   policyName,      /* Eingabe */
               varchar(80)   dadFileName,     /* Eingabe */
               varchar(10)   status)          /* Ausgabe */
```

Parameter:

Tabelle 83. Parameter für dxxmqShredAllCLOB

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
dadFileName	Der Name der DAD-Datei. Die maximale Größe ist 80 Byte.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Zugehörige Referenzen:

- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqInsert

Zweck:

Zerlegt ein ankommendes XML-Dokument aus der Nachrichtenwarteschlange und speichert die Daten in neuen oder vorhandenen Datenbanktabellen. dxxmqInsert verwendet an Stelle eines DAD-Dateinamens einen Objektgruppennamen, um zu bestimmen, wie die Daten gespeichert werden sollen.

Syntax:

```
dxxmqInsert(varchar(48)  serviceName,      /* Eingabe */
            varchar(48)   policyName,      /* Eingabe */
            varchar(80)   collectionName,  /* Eingabe */
            varchar(20)   status)          /* Ausgabe */
```

Parameter:

Tabelle 84. Parameter für dxxmqInsert()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
collectionName	Der Name einer aktivierten XML-Objektgruppe. Die maximale Größe ist 80 Byte.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Im folgenden Beispielfragment ruft der Aufruf dxxmqInsert() das Eingabe-XML-Dokument `order1.xml` aus der Nachrichtenwarteschlange ab, die durch *serviceName* definiert ist, zerlegt das Dokument und fügt entsprechend der in der DAD-Datei angegebenen Zuordnung, mit der sie aktiviert wurde, Daten in die Objektgruppentabellen SALES_ORDER ein.

```
#include "dxx.h"
#include "dxxrc.h"
```

```
EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char          serviceName[48];
              char          policyName[48];
              char          collection[80];     /* Name einer XML-Objektgruppe */
              char          status[10];

              short                 serviceName_ind;
              short                 policyName_ind;
              short                 collection_ind;
              short                 status_ind;
              EXEC SQL END DECLARE SECTION;

              /* Host-Variable und Anzeiger initialisieren  */
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
              strcpy(collection,"sales_ord")
              status[0]=\0;
              serviceName_ind = 0;
              policyName_ind = 0;
              collection_ind = 0;
              status_ind = -1;

              /* Gespeicherte Prozedur aufrufen */
              EXEC SQL CALL dxxmqInsert(:serviceName:serviceName_ind,
                              :policyName:policyName_ind,
                              :collection:collection_ind,
                              :status:status_ind);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqInsertCLOB

Zweck:

Zerlegt ein ankommendes XML-Dokument aus der Nachrichtenwarteschlange und speichert die Daten in neuen oder vorhandenen Datenbanktabellen. dxxmqInsert-CLOB verwendet an Stelle eines DAD-Dateinamens einen Objektgruppennamen, um zu bestimmen, wie die Daten gespeichert werden sollen. Die ankommende Dokumentart ist XMLCLOB.

Syntax:

```
dxxmqInsertCLOB(varchar(48)  serviceName,  /* Eingabe */
            varchar(48)   policyName,      /* Eingabe */
            varchar(80)   collectionName,  /* Eingabe */
            varchar(20)   status)          /* Ausgabe */
```

Parameter:

Tabelle 85. Parameter für dxxmqInsertCLOB()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
collectionName	Der Name einer aktivierten XML-Objektgruppe.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Im folgenden Beispielfragment ruft der Aufruf dxxmqInsertCLOB() das Eingabe-XML-Dokument `order1.xml` aus der Nachrichtenwarteschlange ab, die durch *serviceName* definiert ist, zerlegt das Dokument und fügt entsprechend der in der DAD-Datei angegebenen Zuordnung, mit der sie aktiviert wurde, Daten in die Objektgruppentabellen SALES_ORDER ein.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char          serviceName[48];
              char          policyName[48];
              char          collection[48];     /* Name einer XML-Objektgruppe */
              char          status[10];

              short                 serviceName_ind;
              short                 policyName_ind;
              short                 collection_ind;
```

```
short                status_ind;
EXEC SQL END DECLARE SECTION;

/* Host-Variable und Anzeiger initialisieren  */
strcpy(serviceName, "myService");
strcpy(policyName, "myPolicy");
strcpy(collection,"sales_ord")
status[0] = \0;
serviceName_ind = 0;
policyName_ind = 0;
collection_ind = 0;
status_ind = -1;

/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxmqInsertCLOB(:serviceName:serviceName_ind;
                :policyName:policyName_ind,
                :collection:collection_ind,
                :status:status_ind);
```

Zugehörige Referenzen:

- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqInsertAll

Zweck:

Zerlegt alle ankommenden XML-Dokumente aus der Nachrichtenwarteschlange und speichert die Daten in neuen oder vorhandenen Datenbanktabellen. dxxmqInsertAll verwendet an Stelle eines DAD-Dateinamens einen Objektgruppennamen, um zu bestimmen, wie die Daten gespeichert werden sollen.

Syntax:

```
dxxmqInsertAll(varchar(48)  serviceName,      /* Eingabe */
               varchar(48)   policyName,      /* Eingabe */
               varchar(48)   collectionName,  /* Eingabe */
               varchar(20)   status)          /* Ausgabe */
```

Parameter:

Tabelle 86. Parameter für dxxmqInsertAll()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN

Tabelle 86. Parameter für dxxmqInsertAll() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN
collectionName	Der Name einer aktivierten XML-Objektgruppe. Die maximale Größe ist 80 Byte.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Im folgenden Beispielfragment ruft der Aufruf dxxmqInsertAll alle Eingabe-XML-Dokumente aus der Nachrichtenwarteschlange ab, die durch *serviceName* definiert ist, zerlegt die Dokumente und fügt entsprechend der in der DAD-Datei angegebenen Zuordnung, mit der sie aktiviert wurde, Daten in die Objektgruppentabellen SALES_ORDER ein.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char          serviceName[48];
              char          policyName[48];
              char          collection[80];     /* Name einer XML-Objektgruppe */
              char          status[10];

              short                 serviceName_ind;
              short                 policyName_ind;
              short                 collection_ind;
              short                 status_ind;
              EXEC SQL END DECLARE SECTION;

              /* Host-Variable und Anzeiger initialisieren  */
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
              strcpy(collection,"sales_ord");
              status[0]='\0';
              serviceName_ind = 0;
              policyName_ind = 0;
              collection_ind = 0;
              status_ind = -1;
```

```
/* Gespeicherte Prozedur aufrufen */
EXEC SQL CALL dxxmqInsertAll(:serviceName:serviceName_ind,
                             :policyName:policyName_ind,
                             :collection:collection_ind,
                             :status:status_ind);
```

Zugehörige Konzepte:

- „Übersicht: Gespeicherte Prozeduren und Funktionen von XML Extender für MQSeries" auf Seite 231

Zugehörige Referenzen:

- „Syntaxdiagramme lesen" auf Seite ix
- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Gespeicherte Prozedur dxxmqInsertAllCLOB

Zweck:

Zerlegt alle ankommenden XML-Dokumente aus der Nachrichtenwarteschlange und speichert die Daten in neuen oder vorhandenen Datenbanktabellen. dxxmqInsertAllCLOB verwendet an Stelle eines DAD-Dateinamens einen Objektgruppennamen, um zu bestimmen, wie die Daten gespeichert werden sollen.

Syntax:

```
dxxmqInsertAllCLOB(varchar(48)  serviceName,  /* Eingabe */
                   varchar(48)  policyName,   /* Eingabe */
                   varchar(48)  collectionName,  /* Eingabe */
                   varchar(20)  status)       /* Ausgabe */
```

Parameter:

Tabelle 87. Parameter für dxxmqInsertAllCLOB()

Parameter	Beschreibung	IN/OUT-Parameter
serviceName	Eine Zeichenfolge mit der logischen MQSeries-Zieladresse, an die die Nachricht gesendet werden soll. Wenn der *serviceName* aufgelistet ist, bezieht er sich auf einen Service, der in der Repository-Datei AMT.XML definiert ist. Der DB2.DEFAULT.SERVICE wird verwendet, wenn der *serviceName* nicht angegeben ist. Die maximale Größe von *serviceName* ist 48 Byte.	IN
policyName	Eine Zeichenfolge mit der MQSeries-AMI-Servicerichtlinie, die für die Bearbeitung von Nachrichten verwendet wird. Wenn angegeben, bezieht sich *policyName* auf eine Richtlinie, die in der Repository-Datei AMT.XML definiert ist. Wenn *policyName* nicht angegeben ist, wird die Standardrichtlinie DB2.DEFAULT.POLICY verwendet. Die maximale Größe von *policyName* ist 48 Byte.	IN

Tabelle 87. Parameter für dxxmqInsertAllCLOB() (Forts.)

Parameter	Beschreibung	IN/OUT-Parameter
collectionName	Der Name einer aktivierten XML-Objektgruppe.	IN
status	Text und Codes, die angeben, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde, mögliche Fehlercodes, die generiert wurden, und die Anzahl an XML-Dokumenten, die von der Nachrichtenwarteschlange empfangen oder dorthin gesendet wurden.	OUT

Beispiele:

Im folgenden Beispielfragment ruft der Aufruf dxxmqInsertAllCLOB alle Eingabe-XML-Dokumente aus der Nachrichtenwarteschlange ab, die durch *serviceName* definiert ist, zerlegt die Dokumente und fügt entsprechend der in der DAD-Datei angegebenen Zuordnung, mit der sie aktiviert wurde, Daten in die Objektgruppentabellen SALES_ORDER ein.

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char          serviceName[48];
              char          policyName[48];
              char          collection[48];     /* Name einer XML-Objektgruppe */
              char          status[10];

              short                 serviceName_ind;
              short                 policyName_ind;
              short                 collection_ind;
              short                 status_ind;
              EXEC SQL END DECLARE SECTION;

              /* Host-Variable und Anzeiger initialisieren  */
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
              strcpy(collection,"sales_ord")
              status[0] = '\0';
              serviceName_ind = 0;
              policyName_ind = 0;
              collection_ind = 0;
              status_ind = -1;

              /* Gespeicherte Prozedur aufrufen */
              EXEC SQL CALL dxxmqInsertAllCLOB(:serviceName:serviceName_ind;
                              :policyName:policyName_ind,
                              :collection:collection_ind,
                              :status:status_ind);
```

Zugehörige Referenzen:

- Anhang C, „Grenzwerte für XML Extender", auf Seite 329

Kapitel 12. Extensible Stylesheet Language Transformation (XSLT)

HTML-Dokument mit Hilfe einer XSLT-Formatvorlage erstellen

Die Extensible Stylesheet Language Transformation (XSLT) besteht aus einer Reihe von Formatierungssteuerzeichen, die verwendet werden können, um für jedes der Elemente innerhalb eines XML-Dokuments Formatierungsregeln anzuwenden. XSL wendet verschiedene Darstellungsregeln auf den Inhalt eines XML-Dokuments an, die auf den Elementen, die auftreten, basieren. Vom Entwurf her sind XSLT-Formatvorlagen reguläre XML-Dokumente.

Ursprünglich wurde XSLT für das Seitenlayout erstellt, wird aber jetzt auf unterschiedliche Weise eingesetzt. Beispielsweise kann es als vielseitig einsetzbares Umsetzungstool verwendet werden, als System für die Reorganisation von Dokumentinhalten, oder um mehrere Ergebnisse, z. B. HTML, WAP und SVG, aus einer einzigen Quelle zu generieren.

XSLT ist eine kritische Brücke zwischen der XML-Verarbeitung und gängigeren Programmiersprachen wie HTML. XSLT ermöglicht Ihnen die Umsetzung einer XML-Struktur in andere Datentypen durch Entfernen oder Ersetzen der XML-Befehle. Es ermöglicht Ihnen außerdem, die Reihenfolge von Informationen zu ändern, einige besondere Informationen zu extrahieren oder die Informationen zu sortieren.

Voraussetzungen:

Um ein HTML-Dokument unter Verwendung einer Formatvorlage zu erstellen, müssen Sie die folgenden Tasks ausführen:

1. Erstellen Sie eine XML-Datei in der Ergebnistabelle.
2. Erstellen Sie eine Formatvorlage.

Sie können Ihre HTML-Datei mit Hilfe von XSLTransformToFile oder XSLTransformToClob erstellen. Diese Ausgabedatei kann entweder auf den DB2 UDB-Server oder von der Befehlszeile aus in einen Texteditor geschrieben werden.

Vorgehensweise:

Um Ihre HTML-Datei auf dem DB2 UDB-Server zu erstellen, geben Sie folgende Syntax ein:

```
SELECT XSLTransformToFile( CAST(doc AS CLOB(4k)),
    '$dxx_install_verz$\samples\db2xml\xslt\getstart.xsl',
         0,
   '$dxx_install_verz$\samples\db2xml\html\getstart.html')
 FROM RESULT_TAB
```

Dabei ist *$dxx_install_verz$* das Verzeichnis, in dem Sie DB2 XML Extender installiert haben.

Um Ihre HTML-Datei von der Befehlszeile aus zu erstellen, öffnen Sie einen Texteditor, und geben Sie folgenden Befehl ein:

```
getstart_xslt.cmd
```

Zugehörige Referenzen:

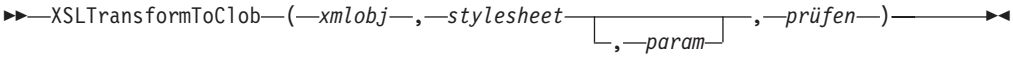
- „Gespeicherte Prozedur XSLTransformToClob()" auf Seite 282
- „Gespeicherte Prozedur XSLTransformToFile()" auf Seite 283

Gespeicherte Prozedur XSLTransformToClob()

Zweck:

XSLTransformToClob() liest ein XML-Dokument als CLOB-Querverweis und eine Formatvorlage als CLOB oder liest aus einer Datei und gibt das Dokument als CLOB zurück.

Syntax:



Parameter:

Parameter	Datentyp	Beschreibung
xmlobj	CLOB	Das XML-Dokument
stylesheet	CLOB, VARCHAR	Die Formatvorlage. Der Standort und Name der Eingabedatei für die Formatvorlage.
param	CLOB VARCHAR	Das XML-Parameterdokument. Der Standort und Name der XML-Parameterdatei.
prüfen	INTEGER	Die Prüfung des xmlobj aktivieren (1) oder inaktivieren (0).

Ergebnisse:
Im Erfolgsfall gibt XSLTransformToClob() Daten vom Typ CLOB zurück.
Beispiele:
Die folgenden Beispiele erstellen eine Beispieltabelle und speichern die beiden Eingabedateien in der Datenbank: `getstart.xml` und `getstart.xsl`. Die Datenbank muss für XML Extender aktiviert sein.

```
CREATE TABLE xslt_tab(xmlobj CLOB(4k), stylesheet CLOB(4k))
INSERT INTO xslt_tab(xmlobj, stylesheet) VALUES(
 DB2XML.XMLCLOBFromFile('c:\dxx_install_verz\samples\db2xml\xml\getstart.xml
'),
DB2XML.XMLCLOBFromFile('c:\dxx_install_verz\samples\db2xml\xslt\getstart.xsl
'))
```

Beispiel 1: Das folgende Beispiel setzt ein XML-Dokument in ein HTML-Dokument um, wobei die erstellte Tabelle verwendet wird:

```
SELECT XSLTransformToClob(xmlobj, stylesheet)
FROM xslt_tab
```

Beispiel 2: Dieses Beispiel setzt ein XML-Dokument in ein HTML-Dokument um, wobei eine Datei mit einer Formatvorlage verwendet wird:

```
SELECT XSLTransformToClob( xmlobj,
  c:\dxx_install_verz\samples\db2xml\xslt\getstart.xsl
')
  FROM xslt_tab
```

Beispiel 3: In diesem Beispiel wird die Ausgabe durch zusätzliche Parameter geändert. Das XML-Parameterdokument muss den Namensbereich definieren. Die Parameter müssen in das Element <*param*> eingeschlossen werden. Der entsprechende Wert kann auch in einem Attribut *value* oder im Inhalt des Elements <*param*> angegeben werden.

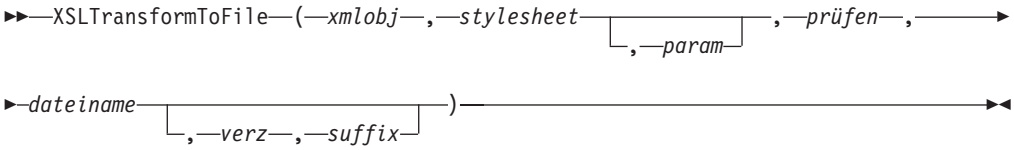
```
c:\dxx_install_verz\samples\db2xml\xml\getstart_xslt_param.xml:
<?xml version="1.0"?>
<params xmlns="http://www.ibm.com.XSLtransformParameters">
   <param name="noShipments" value="true"/>
   <param name="headline">The customers...</param>
</params>

SELECT XSLTranfsormToClob( xmlobj, stylesheet, param, 1)
FROM xslt_tab
```

Gespeicherte Prozedur XSLTransformToFile()

Zweck:

Liest ein XML-Dokument als CLOB und eine Formatvorlage als CLOB oder aus einer Datei ein. Die benutzerdefinierte Funktion (UDF) XSLTransformToFile() schreibt anschließend die Ergebnisse aus der Formatvorlage und dem XML-Dokument in eine Datei. Wenn als Parameter ein Verzeichnis und eine Dateierweiterung angegeben werden, erstellt die UDF eine Datei mit einem eindeutigen Dateinamen in diesem Verzeichnis.

Syntax:



Parameter:

Tabelle 88. Parameter für XSLTransformToFile()

Parameter	Datentyp	Beschreibung
xmlobj	CLOB	Das XML-Dokument
stylesheet	CLOB VARCHAR	Die Formatvorlage. Der Standort und Name der Eingabedatei für die Formatvorlage.
param	VARCHAR VARCH	Das XML-Parameterdokument. Der Standort und Name der XML-Parameterdatei.

Tabelle 88. Parameter für XSLTransformToFile() (Forts.)

Parameter	Datentyp	Beschreibung
prüfen	INTEGER	Die Prüfung des xmlobj aktivieren (1) oder inaktivieren (0).
dateiname	VARCHAR	Der Name der Ausgabedatei.
verz	VARCHAR	Das Verzeichnis der Ausgabedatei.
suffix	VARCHAR	Das Suffix der Ausgabedatei.

Ergebnisse:
XSLTransformToFile() gibt einen Typ VARCHAR für den Namen der geschriebenen Datei zurück.

Beispiele:

Das folgende Beispiel erstellt eine Beispieltabelle und speichert zwei Dateien, `getstart.xml` und `getstart.xsl`, in der Tabelle. Um die Beispieltabelle erstellen zu können, muss die DB2 UDB-Datenbank für XML Extender aktiviert sein.

```
CREATE TABLE xslt_tab(xmlobj CLOB(4k), stylesheet CLOB(4k))

INSERT INTO xslt_tab(xmlobj, stylesheet) VALUES(
DB2XML.XMLCLOBFromFile('$dxx_install_verz$\samples\db2xml\xml\getstart.xml
'),
DB2XML.XMLCLOBFromFile('$dxx_install_verz$\samples\db2xml\xslt\getstart.xsl
'))
```

Dabei ist *$dxx_install_verz$* das Verzeichnis, in dem Sie DB2 XML Extender installiert haben.

Beispiel 1: Dieses Beispiel setzt das XML-Dokument in ein HTML-Dokument um und schreibt das erstellte Dokument in die angegebene Datei:

```
SELECT XSLTransformFile( xmlobj, stylesheet,
   '$dxx_install_verz$samples\db2xml\html\getstart.html

  FROM xslt_tab
```

Dabei ist *$dxx_install_verz$* das Verzeichnis, in dem Sie DB2 XML Extender installiert haben.

Beispiel 2: Dieses Beispiel schreibt ein HTML-Dokument in eine Datei, wobei eine Datei mit einer Formatvorlage verwendet wird. Die Prüfung wird aktiviert, aber das Ergebnis bleibt gleich. Diese Funktion ist notwendig, um Standardwerte aus einem XML-Schema in den Umsetzungsprozess einzuschließen. Es sind keine Parameter angegeben. Der Dateiname wird von der UDF generiert.

```
SELECT XSLTransformToFile( xmlobj,
   '$dxx_install_verz$\samples\db2xml\xslt\getstart.xsl',
   '$dxx_install_verz$\samples\db2xml\html\getstart.html')
FROM xslt_tab
```

Dabei ist *$dxx_install_verz$* das Verzeichnis, in dem Sie DB2 XML Extender installiert haben.

Beispiel 3: In diesem Beispiel wird die Ausgabe durch zusätzliche Parameter geändert. Das XML-Parameterdokument muss den Namensbereich definieren. Die Parameter müssen in das Element <*param*> eingeschlossen werden. Der entsprechende Wert kann auch in einem Attribut *value* oder im Inhalt des Elements <*param*> angegeben werden.

```
$dxx_install_verz$\samples\db2xml\xml\getstart_xslt_param.xml:', 'html')
<?xml version="1.0"?>
<params xmlns="http://www.ibm.com.XSLtransformParameters">
   <param name="noShipments" value="true"/>
   <param name="headline">The customers...</param>
</params>
```

Dabei ist *$dxx_install_verz$* das Verzeichnis, in dem Sie DB2 XML Extender installiert haben.

Beispiel 4: Dieses Beispiel schreibt ein HTML-Dokument in eine Datei, wobei eine Datei mit einer Formatvorlage verwendet wird, und speichert den Dateinamen für jede Zeile in einer zusätzlichen Spalte in der Tabelle.

```
UPDATE TABLE xslt_tab ADD COLUMN filename VARCHAR(512)
UPDATE TABLE xslt_tab SET filename =
  XSLTransformToFile(xmlobj,stylesheet, param, 1,
    '$dxx_install_verz$samples\db2xml\html
         ', 'html')
  FROM xslt_tab
```

Dabei ist *$dxx_install_verz$* das Verzeichnis, in dem Sie DB2 XML Extender installiert haben.

Kapitel 13. Tabellen zur Verwaltungsunterstützung von XML Extender

Beim Aktivieren einer Datenbank werden eine DTD-Repositorytabelle (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.

DTD-Referenztabelle

XML Extender dient auch als XML DTD-Repository. Wenn eine Datenbank für XML aktiviert wurde, wird die DTD-Repositorytabelle DTD_REF erstellt. Jede Zeile dieser Tabelle stellt eine DTD mit zusätzlichen Metadaten-Informationen dar. Sie können auf diese Tabelle zugreifen und eigene 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 89 aufgeführten Spalten enthalten.

Tabelle 89. Tabelle DTD_REF

Spaltenname	Datentyp	Beschreibung
DTDID	VARCHAR(128)	Der Primärschlüssel (eindeutig und nicht NULL). Er wird zur Kennzeichnung der DTD verwendet. Wenn die DTD in der DAD-Datei angegeben ist, muss die DAD-Datei entsprechend dem durch die 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)	Der Verfasser der DTD. Diese Informationen sind optional.
CREATOR	VARCHAR(128)	Die Benutzer-ID, über die die erste Einfügung vorgenommen wird. Diese Spalte ist optional.
UPDATOR	VARCHAR(128)	Die Benutzer-ID, über die die erste Aktualisierung vorgenommen wird. Diese Spalte ist optional.

Eine DTD kann von der Anwendung nur geändert werden, wenn USAGE_COUNT Null ist.

XML-Verwendungstabelle (XML_USAGE)

Die Tabelle XML_USAGE 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 wird beim Aktivieren der Datenbank erstellt. Die Spalten der Tabelle XML_USAGE werden in Tabelle 90 aufgeführt.

Tabelle 90. Tabelle XML_USAGE

Spaltenname	Beschreibung
table_schema	Für eine XML-Spalte der Schemaname der Benutzertabelle, die eine XML-Spalte enthält. Für eine XML-Objektgruppe der Wert DXX_COLL als Standardschemaname.
table_name	Für eine XML-Spalte der Name der Benutzertabelle, die eine XML-Spalte enthält. Für eine XML-Objektgruppe 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.
DTDID	Eine Zeichenfolge, die eine Zuordnung zwischen einer DTD, die in eine DTD_REF eingefügt ist, und einer DTD, die in einer DAD-Datei angegeben ist, herstellt. Dieser Wert muss mit dem Wert des DTDID-Elements in der DAD übereinstimmen. Diese Spalte ist der Fremdschlüssel.
DAD	Der Inhalt der DAD-Datei, die der XML-Spalte oder der XML-Objektgruppe zugeordnet ist.
access_mode	Gibt den verwendeten Zugriffsmodus an: 1 für XML-Objektgruppe, 0 für XML-Spalte.
default_view	Speichert den Namen der Standardsicht, falls eine Standardsicht vorhanden ist.
trigger_suffix	Keine Nullzeichenfolge. Für eindeutige Auslösernamen.
validation	Hat den Wert 1 (Überprüfung) oder den Wert 0 (keine Überprüfung).

Sie dürfen in der Tabelle XML_USAGE keine Einträge hinzufügen, ändern oder löschen. Sie ist nur für den internen Gebrauch von XML Extender konzipiert.

Kapitel 14. Fehlerbehebung

Fehlerbehebung bei XML Extender

Alle eingebetteten SQL-Anweisungen und Aufrufe der DB2 UDB-Befehlszeilenschnittstelle (CLI-Aufrufe) in Ihrem Programm einschließlich der Aufrufe von benutzerdefinierten Funktionen (UDFs) von DB2 UDB XML Extender generieren Codes, die angeben, ob die eingebettete SQL-Anweisung oder der DB2 UDB-CLI-Aufruf erfolgreich ausgeführt wurden.

Ihr Programm kann Informationen abrufen, die diese Codes ergänzen, einschließlich 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 IBM Unterstützungsfunktion weiterleiten. XML Extender enthält eine Tracefunktion, die die Aktivitäten von XML Extender aufzeichnet. Die Trace-Informationen können hilfreiche Daten für die IBM Unterstützungsfunktion enthalten. Verwenden Sie die Tracefunktion nur unter Anleitung der IBM Unterstützungsfunktion.

In diesem Kapitel werden die Tracefunktion, Fehlercodes und Nachrichten beschrieben.

Zugehörige Referenzen:

- „SQLSTATE-Codes und zugeordnete Nachrichtennummern für XML Extender" auf Seite 291
- „XML Extender-Nachrichten" auf Seite 297
- „Trace stoppen" auf Seite 290
- „Trace für XML Extender starten" auf Seite 289

Trace für XML Extender starten

Zweck:

Zeichnet die Aktivitäten des XML Extender-Servers auf. Verwenden Sie zum Starten des Traces die Option `on` mit **dxxtrc**, zusammen mit dem Namen eines bestehenden Verzeichnisses, das die Tracedatei enthalten soll. Wenn der Trace eingeschaltet ist, wird die Datei `dxx`*`INSTANCE`*`.trc` in das angegebene Verzeichnis gestellt. *INSTANCE* ist der Wert von DB2INSTANCE. Jedes DB2 UDB-Exemplar hat seine eigene Protokolldatei. Die Tracedatei ist in der Größe nicht eingeschränkt.

Syntax:

So starten Sie den Trace:

Parameter:

Tabelle 91. Trace-Parameter

Parameter	Beschreibung
trace_verzeichnis	Name eines bestehenden Pfades und Verzeichnisse, in das die Datei `dxx`*`INSTANCE`*`.trc` gestellt wird. Erforderlich, kein Standardwert.

Beispiele:
Das folgende Beispiel zeigt, wie der Trace für ein Exemplar db2inst1 gestartet wird. Die Tracedatei `dxxdb2inst1.trc` wird in das Verzeichnis `/home/db2inst1/dxx_install_verz/log` gestellt.

```
dxxtrc on /home/db2inst1/dxx_install_verz/log
```

Trace stoppen

Zweck:

Schaltet den Trace aus. Trace-Informationen werden nicht länger protokolliert.

Empfehlung: Da beim Ausführen des Traces die Größe der Traceprotokolldatei nicht begrenzt ist und dadurch die Leistung beeinträchtigt werden kann, schalten Sie den Trace in einer Produktionsumgebung aus.

Syntax:

So stoppen Sie den Trace:

Beispiele:
Dieses Beispiel zeigt, dass die Trace-Einrichtung ausgeschaltet ist.

```
dxxtrc off
```

UDF-Rückkehrcodes von XML Extender

Eingebettete SQL-Anweisungen geben Codes in den Feldern SQLCODE, SQLWARN und SQLSTATE der SQLCA-Struktur zurück. Diese Struktur ist in einer Include-Datei SQLCA definiert. (Weitere Informationen zur SQLCA-Struktur und der Include-Datei 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 eine XML Extender-UDF eine Warnungs- oder Fehlerbedingung feststellt, leitet sie die entsprechenden Informationen an DB2 UDB weiter, wo sie in die SQLCODE-Nachricht eingebunden werden.

Eingebettete SQL-Anweisungen und DB2 UDB-CLI-Aufrufe, die die DB2 XML Extender-UDFs aufrufen, geben eventuell SQLCODE-Nachrichten und SQLSTATE-Werte zurück, die für diese UDFs eindeutig sind. DB2 UDB gibt diese Werte jedoch auf die gleiche Weise zurück wie für andere eingebettete SQL-Anweisungen oder andere DB2 UDB-CLI-Aufrufe. Sie greifen daher auf diese Werte auf die gleiche Weise zu wie eingebettete SQL-Anweisungen oder DB2 UDB-CLI-Aufrufe, die keine DB2 UDB XML Extender-UDFs starten.

Rückkehrcodes von gespeicherten Prozeduren von XML Extender

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 symbolische Konstante mit dem Rückkehrcode abgleicht.

`dxx_install_verz/include/dxxrc.h`

Zugehörige Referenzen:

- „SQLSTATE-Codes und zugeordnete Nachrichtennummern für XML Extender" auf Seite 291

SQLSTATE-Codes und zugeordnete Nachrichtennummern für XML Extender

Tabelle 92. SQLSTATE-Codes und Nachrichtennummern

SQLSTATE	Nachrichten-nummer	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	XML Extender kann die angegebene Datei nicht öffnen.
38X01	DXXA072E	XML Extender hat versucht, die Datenbank vor dem Aktivieren automatisch zu binden, konnte jedoch die Bindedateien nicht finden.
	DXXC001E	XML Extender konnte die angegebene Datei nicht finden.
38X02	DXXC002E	XML Extender kann keine Daten in der angegebenen Datei lesen.

Tabelle 92. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichten-nummer	Beschreibung
38X03	DXXC003E	XML Extender kann keine Daten in die Datei schreiben.
	DXXC011E	XML Extender kann keine Daten in die Trace-Steuerdatei schreiben.
38X04	DXXC004E	XML Extender konnte die angegebene Zeigereinheit nicht bedienen.
38X05	DXXC005E	Die Dateigröße ist größer als die XMLVarchar-Größe, und XML Extender kann nicht alle Daten aus der Datei importieren.
38X06	DXXC006E	Die Dateigröße ist größer als die XMLCLOB-Größe, und XML Extender kann nicht alle Daten aus der Datei importieren.
38X07	DXXC007E	Die Anzahl der Byte im LOB-Querverweis 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	XML Extender konnte keinen Speicher vom Betriebssystem zuordnen.
38X11	DXXA009E	Diese gespeicherte Prozedur gilt nur für eine XML-Spalte.
38X12	DXXA010E	Bei dem Versuch, die Spalte zu aktivieren, konnte XML Extender die DTD-ID nicht finden. Diese DTD-ID ist die für die DTD in der DAD-Datei angegebene Kennung.
	DXXQ060E	Beim Versuch, die Spalte zu aktivieren, konnte XML Extender die Schema-ID nicht finden. Die Schema-ID entspricht dem Wert des Speicherpositionsattributs des Befehls nonamespacelocation, der sich innerhalb des Befehls schemabindings in der DAD-Datei befindet.
38X14	DXXD000E	Es wurde versucht, ein ungültiges Dokument in einer Tabelle zu speichern. Überprüfung ist fehlgeschlagen.
38X15	DXXA056E	Das Prüfungselement in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Tabelle 92. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichten-nummer	Beschreibung
	DXXA057E	Das name-Attribut einer Neben-tabelle in der DAD-Datei ist fehler-haft 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.
	DXXA061E	Das multi_occurrence-Attribut einer Spalte in der DAD-Datei ist fehler-haft oder nicht vorhanden.
	DXXQ000E	Ein verbindliches Element ist in der DAD-Datei nicht vorhanden.
	DXXQ056E	Das angegebene Element/Attribut kann keiner Spalte zugeordnet wer-den, die als Teil eines Fremd-schlüssels angegeben wurde. Die Datenwerte für Fremdschlüssel wer-den anhand der Datenwerte der Primärschlüssel ermittelt. Eine Zuordnung des angegebenen Elements/Attributs im XML-Doku-ment zu einer Tabelle und Spalte ist nicht erforderlich.
	DXXQ057E	Die Befehle schemabindings und dtdid dürfen nicht gleichzeitig in der DAD-Datei enthalten sein.
	DXXQ058E	Der Befehl nonamespacelocation im Befehl schemabindings fehlt in der DAD-Datei.
	DXXQ059E	Für die Schemaprüfung darf sich der Befehl doctype in der DAD nicht innerhalb des Befehls XCollection befinden.
	DXXQ062E	Diese Fehlerbedingung wird norma-lerweise durch das Fehlen der Angabe multi_occurrence = YES im übergeordneten element_node des angegebenen Elements oder Attri-buts verursacht.

Tabelle 92. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichten-nummer	Beschreibung
	DXXQ063E	Der Wert des Attributs multi_occurrence für den angegebenen element_node in der DAD-Datei ist fehlerhaft oder nicht vorhanden. Der Wert muss 'yes' oder 'no' sein, wobei die Groß-/Kleinschreibung nicht beachtet werden muss.
	DXXQ064E	Eine in der Verknüpfungsbedingung angegebene Schlüsselspalte wurde keinem Element- oder Attributknoten zugeordnet.
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	XML Extender hat einen internen Fehler festgestellt.
	DXXG006E	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.
38X24	DXXQ007E	Das Element attribute_node in der DAD-Datei hat kein Spaltenelement oder keinen RDB_node.
38X25	DXXQ008E	Ein Element text_node in der DAD-Datei hat kein Spaltenelement.

Tabelle 92. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichten-nummer	Beschreibung
38X26	DXXQ009E	Die angegebene Ergebnistabelle konnte im Systemkatalog nicht gefunden werden.
38X27	DXXQ010E DXXQ040E	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 von XML Extender generiertes XML-Dokument ist zu groß für die Spalte der Ergebnistabelle.
	DXXQ040E	Der angegebene Elementname in der DAD-Datei (Document Access Definition = Dokumentzugriffs-definition) ist falsch.
38X28	DXXQ012E	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 Unterelemente müssen den im Anfangselement definierten Tabellen entsprechen. Der Tabellenname in diesem RDB_node ist im Anfangs-element nicht enthalten.
38X29	DXXQ013E	Die Elementtabelle oder -spalte muss einen Namen in der DAD-Datei haben.
	DXXQ015E	Die Bedingung im condition-Ele-ment in der DAD-Datei hat ein ungültiges Format.
	DXXQ061E	Das Format der Zeichenfolge-darstellung ist ungültig. Falls es sich bei der Zeichenfolge um einen Datums-, Zeit- oder Zeitmarkenwert handelt, entspricht die Syntax nicht seinem Datentyp.
38X30	DXXQ014E	Ein Element element_node in der DAD-Datei hat kein name-Attribut.
	DXXQ018E	Die Klausel ORDER BY fehlt in der SQL-Anweisung in einer DAD-Da-tei, die eine Zuordnung zwischen SQL und XML herstellt.
38X31	DXXQ019E	Das objids-Element hat kein Spaltenelement in der DAD-Datei, die die Zuordnung zwischen SQL und XML herstellt.

Tabelle 92. SQLSTATE-Codes und Nachrichtennummern (Forts.)

SQLSTATE	Nachrichten-nummer	Beschreibung
38x33	DXXG005E	Dieser Parameter wird in diesem Release nicht unterstützt. Er wird in späteren Versionen unterstützt.
38x34	DXXG000E	Es wurde ein ungültiger Name angegeben.
38X36	DXXA073E	Die Datenbank war nicht gebunden, als der Benutzer versuchte, sie zu aktivieren.
38X37	DXXG007E	Die Ländereinstellung zum Server-betriebssystem ist nicht konsistent mit der DB2 UDB-Codepage.
38X38	DXXG008E	Die Ländereinstellung zum Server-betriebssystem kann in der Code-page-Tabelle nicht gefunden werden.
38X41	DXXQ048E	Bei der Verarbeitung der Formatvor-lage wurde ein interner Fehler zurückgegeben. Das XML-Doku-ment oder die Formatvorlage ist möglicherweise nicht gültig.
38X42	DXXQ049E	Die angegebene Ausgabedatei ist im Verzeichnis bereits vorhanden.
38X43	DXXQ050E	Die UDF konnte keinen eindeutigen Dateinamen für das Ausgabe-dokument im angegebenen Ver-zeichnis erstellen, da sie nicht darauf zugreifen konnte. Alle Datei-namen, die generiert werden kön-nen, sind in Gebrauch oder das Verzeichnis ist möglicherweise nicht vorhanden.
38X44	DXXQ051E	Ein oder mehrere Ein- oder Aus-gabeparameter haben keinen gülti-gen Wert.
38X45	DXXQ055E	Während der Konvertierungs-operation ist ein ICU-Fehler aufge-treten.

XML Extender-Nachrichten

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 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 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 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 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 von 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: 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: 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 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, müssen Sie sie nicht 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 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 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 XML Extender die DAD-Datei nicht finden.

Benutzeraktion: Stellen Sie sicher, dass Sie den richtigen Datenbank-, Tabellen- und Spaltennamen angegeben haben.

DXXA024E XML Extender ist beim Zugriff auf die Systemkatalogtabellen auf einen internen Fehler gestoßen.

Erläuterung: 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 XML Extender die Standardsicht nicht freigeben.

Benutzeraktion: Stellen Sie sicher, dass die Administrator-ID für XML Extender die erforderlichen Berechtigungen zum Freigeben der Standardsicht hat.

DXXA026E Die Nebentabelle <*nebentabelle*> kann nicht freigegeben (Drop) werden.

Erläuterung: Bei dem Versuch, eine Spalte zu inaktivieren, konnte XML Extender die angegebene Tabelle nicht freigeben.

Benutzeraktion: Stellen Sie sicher, dass die Administrator-ID für 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:

- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein Auslöser mit diesem Namen ist nicht vorhanden.

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:

- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein Auslöser mit diesem Namen ist nicht vorhanden.

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:

- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein Auslöser mit diesem Namen ist nicht vorhanden.

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:

- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein Auslöser mit diesem Namen ist nicht vorhanden.

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 XML Extender den Wert von DXXROO-

T_ID in der Anwendungstabelle nicht auf NULL setzen.

Benutzeraktion: Stellen Sie sicher, dass die Administrator-ID für 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 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 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 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 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 Nebentabelle <*nebentabelle*> konnte nicht erstellt werden.

Erläuterung: Bei dem Versuch, eine Spalte zu aktivieren, konnte XML Extender die angegebene Nebentabelle nicht erstellen.

Benutzeraktion: Stellen Sie sicher, dass die Administrator-ID für XML Extender die erforderlichen Berechtigungen zum Erstellen der Nebentabelle hat.

DXXA047E Die Spalte konnte nicht aktiviert werden.

Erläuterung: XML Extender konnte eine Spalte nicht aktivieren, da ein interner Auslöser fehlgeschlagen ist. Mögliche Ursachen:

- Die DAD-Datei enthält falsche Syntax.
- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein weiterer Auslöser mit demselben Namen ist vorhanden.

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:

- Die DAD-Datei enthält falsche Syntax.
- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein weiterer Auslöser mit demselben Namen ist vorhanden.

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:

- Die DAD-Datei enthält falsche Syntax.
- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein weiterer Auslöser mit demselben Namen ist vorhanden.

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:

- Die DAD-Datei enthält falsche Syntax.
- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein weiterer Auslöser mit demselben Namen ist vorhanden.

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:

- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein Auslöser mit diesem Namen ist nicht vorhanden.

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:

- Die DAD-Datei enthält falsche Syntax.
- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein weiterer Auslöser mit demselben Namen ist vorhanden.

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:

- Die DAD-Datei enthält falsche Syntax.
- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein weiterer Auslöser mit demselben Namen ist vorhanden.

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:

- Die DAD-Datei enthält falsche Syntax.
- Dem System steht nicht mehr ausreichend Speicherplatz zur Verfügung.
- Ein weiterer Auslöser mit demselben Namen ist vorhanden.

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 Nebentabellenname <*nebentabelle*> in der DAD ist ungültig.

Erläuterung: Das name-Attribut einer Nebentabelle in der DAD-Datei ist fehlerhaft oder nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, dass das name-Attribut einer Nebentabelle 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-Attribut <*standortpfad*> 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: 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: 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. Die gespeicherte Prozedur erwartet einen STRING-Parameter.

Erläuterung: Die gespeicherte Prozedur erwartet einen STRING-Parameter.

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 von 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 iSeries-Benutzer-ID handelt.

DXXA077E Die Lizenzberechtigung ist nicht gültig. Das Fehlerprotokoll des Servers enthält weitere Details hierzu.

Erläuterung: Die Softwarelizenz ist abgelaufen oder nicht vorhanden.

Benutzeraktion: Wenden Sie sich an Ihren Serviceanbieter, um eine neue Softwarelizenz zu erhalten.

DXXC000E Die angegebene Datei konnte nicht geöffnet werden.

Erläuterung: 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: 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: 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: 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-Querverweis konnte nicht bearbeitet werden: <*querverweis_rc*>.

Erläuterung: XML Extender konnte den angegebenen Querverweis nicht bearbeiten.

Benutzeraktion: Stellen Sie sicher, dass der LOB-Querverweis richtig gesetzt ist.

DXXC005E Die Größe der Eingabedatei übersteigt die Größe von XMLVarchar.

Erläuterung: Die Dateigröße ist größer als die XML-Varchar-Größe, und XML Extender kann nicht alle Daten aus der Datei importieren.

Benutzeraktion: Verwenden Sie den Spaltentyp XML-CLOB.

DXXC006E Die Eingabedatei übersteigt die DB2 UDB-LOB-Begrenzung.

Erläuterung: Die Dateigröße ist größer als die XML-CLOB-Größe, und 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-Querverweis abgerufen werden.

Erläuterung: Die Anzahl der Byte im LOB-Querverweis entspricht nicht der Dateigröße.

Benutzeraktion: Stellen Sie sicher, dass der LOB-Querverweis 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: 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 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: 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.

DXXC013E Die Ergebnisse der Extraktions-UDF überschreiten die Größenbeschränkung für den UDF-Rückgabetyp.

Erläuterung: Die Daten, die durch eine Extraktions-UDF zurückgegeben werden, müssen innerhalb der Größenbeschränkung des Rückgabetyps der UDF liegen. Eine Definition finden Sie im Handbuch DB2 UDB XML Extender Verwaltung und Programmierung. Beispielsweise dürfen die Ergebnisse von extractVarchar nicht größer als 4000 Byte (einschließlich abschließendem NULL-Wert) sein.

Benutzeraktion: Verwenden Sie eine Extraktions-UDF, deren Größenbeschränkung für den Rückgabetyp größer ist: 254 Byte für extractChar(), 4 KB für extractVarchar() und 2 GB für extractClob().

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 auftritt.

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: 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: 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 Interer Fehler CLISTATE=<*clistatus*>**, RC=**<*cli_rc*>**, build** <*build_ID*>**, Datei** <*dateiname*>**, Zeile** <*zeilennummer*> **CLIMSG=**<*CLI_nachricht*>**.**

Erläuterung: 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 alle relevanten DADs, XML-Dokumente und Tabellendefinitionen mit.

DXXG007E Ländereinstellung <*ländereinstellung*> **ist nicht konsistent mit der DB2 UDB-Codepage** <*codepage*>**.**

Erläuterung: Die Ländereinstellung zum Serverbetriebssystem ist nicht konsistent mit der DB2 UDB-Codepage.

Benutzeraktion: Korrigieren Sie die Ländereinstellung zum Serverbetriebssystem, und starten Sie DB2 erneut.

DXXG008E Ländereinstellung <*ländereinstellung*> **wird nicht unterstützt.**

Erläuterung: Die Ländereinstellung zum Serverbetriebssystem kann in der Codepagetabelle nicht gefunden werden.

Benutzeraktion: Korrigieren Sie die Ländereinstellung zum Serverbetriebssystem, und starten Sie DB2 erneut.

DXXG017E Das Limit für *XML_Extender_konstante* wurde in Build *build_ID*, Datei *dateiname* und Zeile *zeilennummer* überschritten.

Erläuterung: Prüfen Sie im Handbuch 'XML Extender Verwaltung und Programmierung', ob Ihre Anwendung einen Wert in der Grenzwerttabelle überschritten hat. Wenn kein Limit überschritten wurde, wenden Sie sich an Ihren Softwareserviceanbieter. Geben Sie beim Melden des Fehlers alle Ausgabenachrichten, Tracedateien und Informationen an, wie der Fehler reproduziert werden kann, z. B. Eingabe-DADs, XML-Dokumente und Tabellendefinitionen.

Benutzeraktion: Korrigieren Sie die Ländereinstellung zum Serverbetriebssystem, und starten Sie DB2 erneut.

DXXM001W Ein DB2 UDB-Fehler ist aufgetreten.

Erläuterung: DB2 hat den angegebenen Fehler festgestellt.

Benutzeraktion: Weitere Erläuterungen finden Sie in den Begleitnachrichten und in der Dokumentation 'DB2 UDB Messages and Codes' für Ihr Betriebssystem.

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.

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 Ebene einen einspaltigen Kandidatenschlüssel gibt. Wenn in einer Tabelle kein solcher Schlüssel vorhanden ist, sollte die Abfrage mit einem Tabellenausdruck und der integrierten DB2 UDB-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: 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 zu 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 von 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 Element objids 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 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 UDB SQL-Kennung.

Benutzeraktion: Korrigieren Sie die Zeichenfolge in der DAD, so dass sie den Regeln für DB2 UDB SQL-Kennungen entspricht.

DXXQ034E Ungültiges Bedingungselement im Anfangs-RDB_node (RDB-Knoten) der DAD: <*bedingung*>

Erläuterung: Das Bedingungselement muss eine gültige WHERE-Klausel aus Verknüpfungsbedingungen, die über die Verknüpfung AND verbunden sind, sein.

Benutzeraktion: Siehe die Dokumentation zu XML Extender für die richtige Syntax der Verknüpfungsbedingung in einer DAD.

DXXQ035E Ungültige Verknüpfungsbedingung im Anfangs-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 zu 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 untergeordneten Elemente.

DXXQ038E Die SQL-Anweisung ist zu lang: SQL-_Anweisung

Erläuterung: Die im <*SQL_stmt*>-Element der DAD angegebene SQL-Anweisung überschreitet die zulässige Anzahl an Byte.

Benutzeraktion: Verringern Sie die Länge der SQL-Anweisung auf einen Wert kleiner-gleich 32765 Byte für Windows und UNIX, bzw. 16380 Byte für OS/390 und iSeries.

DXXQ039E Für eine Tabelle in der DAD-Datei wurden zu viele Spalten angegeben.

Erläuterung: Die DAD-Datei, die für die Zerlegung oder RDB-Zusammensetzung verwendet wird, kann insgesamt maximal aus 100 text_node- und attribute-_node-Elementen bestehen, die eindeutige Spalten innerhalb derselben Tabelle angeben.

Benutzeraktion: Verringern Sie die Summe der text-_node- und attribute_node-Elemente, die sich auf eindeutige Spalten innerhalb derselben Tabelle beziehen, auf höchstens 100.

DXXQ040E Der Elementname <*elementname*> **in der DAD-Datei ist ungültig.**

Erläuterung: Der angegebene Elementname in der DAD-Datei (Document Access Definition = Dokumentzugriffsdefinition) ist falsch.

Benutzeraktion: Stellen Sie sicher, dass der Elementname in der DAD-Datei korrekt eingegeben ist. Siehe die DTD für die DAD-Datei.

DXXQ041W Das XML-Dokument wurde erfolgreich generiert. Ein oder mehrere der angegebenen Überschreibungspfade sind ungültig und werden ignoriert.

Erläuterung: Geben Sie nur einen Überschreibungspfad an.

Benutzeraktion: Stellen Sie sicher, dass der Elementname in der DAD-Datei korrekt eingegeben ist. Siehe die DTD für die DAD-Datei.

DXXQ043E Attribut <*attributname*> **konnte unter Element** <*elementname*> **nicht gefunden werden.**

Erläuterung: Das Attribut <*attributname*> war im Element <*elementname*> oder einem der untergeordneten Elemente nicht vorhanden.

Benutzeraktion: Stellen Sie sicher, dass das Attribut im XML-Dokument an allen Positionen erscheint, die in der DAD angefordert werden.

DXXQ044E Element <*elementname*> **hat kein übergeordnetes Element** <*vorfahre*>**.**

Erläuterung: Gemäß der DAD ist <*vorfahre*> ein übergeordnetes Element von <*elementname*>. Im XML-Dokument haben ein oder mehrere Elemente <*elementname*> kein solches übergeordnetes Element.

Benutzeraktion: Stellen Sie sicher, dass die Verschachtelung von Elementen im XML-Dokument den Angaben in der entsprechenden DAD entspricht.

DXXQ045E Unterverzeichnisstruktur unter Element <*elementname*> enthält mehrere Attribute mit dem Namen <*attributname*>.

Erläuterung: Eine Unterverzeichnisstruktur unter <*elementname*> im XML-Dokument enthält mehrere Exemplare des Attributs <*attributname*>, die gemäß der DAD in dieselbe Zeile zerlegt werden sollen. Elemente oder Attribute, die zerlegt werden sollen, müssen eindeutige Namen haben.

Benutzeraktion: Stellen Sie sicher, dass das Element oder Attribut in der Unterverzeichnisstruktur einen eindeutigen Namen hat.

DXXQ046W DTD-ID nicht in DAD gefunden.

Erläuterung: In der DAD ist VALIDATION auf YES gesetzt, aber das DTDID-Element 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.

DXXQ047E Parser-Fehler in Zeile <*zeilennummer*>, Spalte <*spaltennummer*>: *nachricht*

Erläuterung: Der Parser konnte das Dokument aufgrund des berichteten Fehlers nicht syntaktisch analysieren.

Benutzeraktion: Korrigieren Sie den Fehler im Dokument, gegebenenfalls unter Verwendung der XML-Spezifikationen.

DXXQ048E Interner Fehler - siehe Tracedatei.

Erläuterung: Bei der Verarbeitung der Formatvorlage wurde ein interner Fehler zurückgegeben. Das XML-Dokument oder die Formatvorlage ist möglicherweise nicht gültig.

Benutzeraktion: Stellen Sie sicher, dass das XML-Dokument und die Formatvorlage gültig sind.

DXXQ049E Die Ausgabedatei ist bereits vorhanden.

Erläuterung: Die angegebene Ausgabedatei ist im Verzeichnis bereits vorhanden.

Benutzeraktion: Ändern Sie den Ausgabepfad oder Dateinamen für das Ausgabedokument in einen eindeutigen Namen oder löschen Sie die vorhandene Datei.

DXXQ050E Ein eindeutiger Dateiname kann nicht erstellt werden.

Erläuterung: Die UDF konnte keinen eindeutigen Dateinamen für das Ausgabedokument im angegebenen Verzeichnis erstellen, da sie nicht darauf zugreifen konnte, alle Dateinamen, die generiert werden können, im Gebrauch sind oder das Verzeichnis möglicherweise nicht vorhanden ist.

Benutzeraktion: Stellen Sie sicher, dass die UDF Zugriff auf das angegebene Verzeichnis hat, oder wechseln Sie zu einem Verzeichnis mit verfügbaren Dateinamen.

DXXQ051E Keine Ein- oder Ausgabedaten.

Erläuterung: Ein oder mehrere Ein- oder Ausgabeparameter haben keinen gültigen Wert.

Benutzeraktion: Überprüfen Sie die Anweisung, ob erforderliche Parameter fehlen.

DXXQ052E Beim Zugriff auf die Tabelle DB2XML.XML_USAGE ist ein Fehler aufgetreten.

Erläuterung: Entweder wurde die Datenbank nicht aktiviert, oder die Tabelle DB2XML.XML_USAGE wurde freigegeben.

Benutzeraktion: Stellen Sie sicher, dass die Datenbank aktiviert und die Tabelle DB2XML.XML_USAGE im Zugriff ist.

DXXQ053E Eine SQL-Anweisung ist fehlgeschlagen: *nachricht*

Erläuterung: Eine SQL-Anweisung, die während der XML Extender-Verarbeitung generiert wurde, konnte nicht ausgeführt werden.

Benutzeraktion: Weitere Einzelheiten finden Sie im Trace. Wenn Sie die Fehlerbedingung nicht korrigieren können, 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.

DXXQ054E Ungültiger Eingabeparameter: *parameter*

Erläuterung: Der angegebene Eingabeparameter für eine gespeicherte Prozedur oder UDF ist ungültig.

Benutzeraktion: Prüfen Sie die Signatur der entsprechenden gespeicherten Prozedur oder UDF und stellen Sie sicher, dass der tatsächliche Eingabeparameter korrekt ist.

DXXQ055E ICU-Fehler: *icu_fehler*

Erläuterung: ICU-Fehler bei der Konvertierung aufgetreten.

Benutzeraktion: Berichten Sie Ihrem Softwareservice-Provider von diesem Fehler. Fügen Sie Tracedatei, Fehlernachricht und Anweisungen zur Rekonstruktion des Fehlers bei.

DXXQ056E Element/Attribut *xmlname* kann nicht der Spalte zugeordnet werden, die als Teil des Fremdschlüssels bestimmt wurde (Spalte *spalte* in Tabelle *tabelle*).

Erläuterung: Das angegebene Element bzw. das angegebene Attribut kann keiner Spalte zugeordnet werden, die als Teil eines Fremdschlüssels angegeben wurde. Datenwerte für Fremdschlüssel werden durch die der Primärschlüssel festgelegt. Eine Zuordnung des im XML-Dokument angegebenen Elements bzw. des angegebenen Attributs zu einer Tabelle oder Spalte ist nicht notwendig.

Benutzeraktion: Entfernen Sie die Zuordnung des RDB_node zur angegebenen Spalte und Tabelle in der DAD.

DXXQ057E Die Befehle schemabindings und dtdid dürfen nicht gleichzeitig in der DAD-Datei enthalten sein.

Erläuterung: Die Befehle schemabindings und dtdid dürfen nicht gleichzeitig in der DAD-Datei enthalten sein.

Benutzeraktion: Vergewissern Sie sich, dass nur der Befehl schemabindings oder der Befehl dtdid in der DAD-Datei enthalten ist, aber nicht beide zusammen.

DXXQ058E Der Befehl nonamespacelocation im Befehl schemabindings fehlt in der DAD-Datei.

Erläuterung: Der Befehl nonamespacelocation im Befehl schemabindings fehlt in der DAD-Datei.

Benutzeraktion: Fügen Sie dem Befehl schemabindings den Befehl nonamespacelocation hinzu.

DXXQ059E Für die Schemaprüfung darf sich der Befehl doctype in der DAD nicht innerhalb des Befehls XCollection befinden.

Erläuterung: Für die Schemaprüfung darf sich der Befehl doctype in der DAD nicht innerhalb des Befehls XCollection befinden.

Benutzeraktion: Entfernen Sie für die Schemaprüfung den Befehl doctype aus dem Befehl Xcollection.

DXXQ060E Der Versuch, die Schema-ID *schemaid* zu finden, ist fehlgeschlagen.

Erläuterung: Beim Versuch, die Spalte zu aktivieren, konnte XML Extender die Schema-ID nicht finden. Die Schema-ID entspricht dem Wert des Speicherpositionsattributs des Befehls nonamespacelocation, der sich innerhalb des Befehls schemabindings in der DAD-Datei befindet.

Benutzeraktion: Vergewissern Sie sich, dass der richtige Wert für die Schema-ID in der DAD-Datei angegeben ist.

DXXQ061E Das Format der Zeichenfolge ist ungültig.

Erläuterung: Das Format der Zeichenfolgedarstellung ist ungültig. Falls es sich bei der Zeichenfolge um einen Datums-, Zeit- oder Zeitmarkenwert handelt, entspricht die Syntax nicht seinem Datentyp.

Benutzeraktion: Vergewissern Sie sich, dass das Format des Datums-, Zeit- oder Zeitmarkenwerts dem Format seines Datentyps entspricht.

DXXQ062E Es sind keine Zeilen der Ergebnismenge für *tabelle* verfügbar, um einen XML-Wert für *element* zu erzeugen.

Erläuterung: Diese Fehlerbedingung wird normalerweise durch das Fehlen der Angabe multi_occurrence = YES im übergeordneten element_node des angegebenen Elements oder Attributs verursacht.

Benutzeraktion: Vergewissern Sie sich in der DAD, dass der Wert des Attributs multi_occurrence im übergeordneten element_node das mehrfache Vorkommen in untergeordneten Elementknoten korrekt wiedergibt.

DXXQ063E Der Wert des Attributs für mehrfaches Vorkommen (multi_occurrence) für *elementname* in der DAD-Datei ist ungültig.

Erläuterung: Der Wert des Attributs multi_occurrence für den angegebenen element_node in der DAD-Datei ist fehlerhaft oder nicht vorhanden. Der Wert muss 'yes' oder 'no' sein, wobei die Groß-/Kleinschreibung nicht beachtet werden muss.

Benutzeraktion: Stellen Sie sicher, dass das Attribut multi_occurrence in der DAD-Datei korrekt angegeben ist.

DXXQ064E Die Spalte *spalte* wurde in der Fremdtabelle *tabelle* nicht gefunden.

Erläuterung: Eine in der Verknüpfungsbedingung angegebene Schlüsselspalte wurde keinem Element- oder Attributknoten zugeordnet.

Benutzeraktion: Stellen Sie sicher, dass die in der DAD-Datei angegebene Verknüpfungsbedingung korrekt ist und alle Schlüsselspalten Element- oder Attributknoten zugeordnet sind.

DXXQ065I Alle Auslöser, die sich auf für XML aktivierte Spalten beziehen, wurden erfolgreich erneut generiert.

Erläuterung: Dies ist eine Informationsnachricht.

Benutzeraktion: Keine Aktion erforderlich.

DXXQ066E Der Primärschlüssel für die Tabelle *tabellenname* ist nicht vorhanden.

Erläuterung: XML Extender konnte den Primärschlüssel für die Tabelle *tabellenname* nicht ermitteln. Stellen Sie sicher, dass er Primärschlüssel für die Tabelle nach Aktivierung der Spalte für XML nicht gelöscht wurde.

Benutzeraktion: Ändern Sie die Tabelle so, dass der Primärschlüssel hinzugefügt wird, der als Root-ID angegeben wurde, als die Spalte für XML aktiviert wurde.

DXXQ067E Der Versuch, *aktion* auszuführen, ist fehlgeschlagen.

Erläuterung: Beim Versuch, *aktion* auszuführen, ist ein SQL-Fehler aufgetreten.

Benutzeraktion: Wenden Sie sich an Ihren Softwareserviceanbieter. Geben Sie beim Weitermelden des Fehlers die XML Extender-Tracedatei an.

Anhang A. Beispiele

Dieser Anhang zeigt die mit den Beispielen in diesem Handbuch verwendeten Objekte.

Beispiel für XML-DTD

Die folgende DTD wird für das Dokument `getstart.xml` verwendet, auf das in diesem Handbuch durchgängig verwiesen 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 15. Beispiel-XML-DTD: `getstart.dtd`

Beispiel für 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 "dxx_install_verz/samples/db2xml/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 16. Beispiel-XML- Dokument: getstart.xml

Dokumentzugriffsdefinitionsdateien

Die folgenden Abschnitte enthalten Dokumentzugriffsdefinitionsdateien (Document Access Definition, DAD), die über XML-Spalten- oder XML-Objektgruppenzugriffsmodi relationalen DB2-Tabellen XML-Daten zuordnen.

- „Beispiel-DAD-Datei: XML-Spalte"
- „Beispiel-DAD-Datei: XML-Objektgruppe: SQL-Zuordnung" auf Seite 315 zeigt eine DAD-Datei für eine XML-Objektgruppe mit SQL-Zuordnung.
- „Beispiel-DAD-Datei: XML: RDB_node-Zuordnung" auf Seite 316 zeigt eine DAD-Datei für eine XML-Objektgruppe mit RDB_node-Zuordnung.

Beispiel-DAD-Datei: XML-Spalte

Diese DAD-Datei enthält die Zuordnung für eine XML-Spalte und definiert die Tabelle, die Nebentabellen und Spalten, die die XML-Daten enthalten sollen.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM  "dxx_install_verz/samples/db2xml/dtd/dad.dtd">
<DAD>
   <dtdid> "dxx_install_verz/samples/db2xml/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 17. Beispiel-DAD-Datei für eine XML-Spalte: getstart_xcolumn.dad

Beispiel-DAD-Datei: XML-Objektgruppe: SQL-Zuordnung

Diese DAD-Datei enthält eine SQL-Anweisung, die die DB2 UDB-Tabellen, -Spalten und -Bedingungen angibt, die die XML-Daten enthalten sollen.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "dxx_install_verz/samples/db2xml/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
               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 "dxx_install_verz/samples/db2xml/dtd/getstart.dtd
"</doctype>
```

Abbildung 18. Beispiel-DAD-Datei für eine XML-Objektgruppe mit SQL-Zuordnung: order_sql.dad (Teil 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_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>
     </element_node>
   </element_node>
</root_node>
</Xcollection>
</DAD>
```

Abbildung 18. Beispiel-DAD-Datei für eine XML-Objektgruppe mit SQL-Zuordnung: order_sql.dad (Teil 2 von 2)

Beispiel-DAD-Datei: XML: RDB_node-Zuordnung

Diese DAD-Datei verwendet <RDB_node>-Elemente zum Definieren der DB2 UDB-Tabellen, -Spalten und -Bedingungen, die die XML-Daten enthalten sollen.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "SQLLIB/samples/db2xml/dtd/dad.dtd>
<DAD>
 <dtdid>E:\dtd\lineItem.dtd</dtdid>
 <validation>YES</validation>
 <Xcollection>
 <prolog>?xml version="1.0"?</prolog>
 <doctype>!DOCTYPE Order SYSTEM
                  "SQLLIB/samples/db2xml/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">

       <element_node name="Name">
                    <text_node>
                      <RDB_node>
                    <table name="order_tab"/>
                   <column name="customer_name"/>
                       </RDB_node>
                    </text_node>
          </element_node>
       <element_node name="Email">
                    <text_node>
                      <RDB_node>
                    <table name="order_tab"/>
                    <column name="customer_email"/>
                       </RDB_node>
                    </text_node>
          </element_node>
          </element_node>
     <element_node name="Part">
         <attribute_node name="Key">
                      <RDB_node>
                       <table name="part_tab"/>
                    <column name="part_key"/>
                      </RDB_node>
         </attribute_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>

          <element_node name="Quantity">
                    <text_node>
                      <RDB_node>
                       <table name="part_tab"/>
                       <column name="qty"/>
                      </RDB_node>
                    </text_node>
          </element_node>
         <element_node name="Shipment" multi_occurrence="YES">
              <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> <!-- Ende des Elements Shipment-->
       </element_node> <!-- Ende des Elements Part -->
    </element_node> <!-- Ende des Elements Order -->
</root_node>

</Xcollection>

</DAD>
```

Anhang B. Überlegungen zur Codepage

XML-Dokumente und andere zugehörige Dateien müssen für jeden Client oder Server, der auf die Dateien zugreift, korrekt codiert sein. XML Extender geht bei der Verarbeitung einer Datei von einigen Annahmen aus; Sie müssen verstehen, wie er die Konvertierung von Codepages ausführt. Die wichtigsten Punkte dabei sind:

- Sicherstellen, dass die tatsächliche Codepage des Clients, der ein XML-Dokument von DB2 UDB abruft, mit der Codierung des Dokuments übereinstimmt.
- Sicherstellen, dass bei der Verarbeitung des Dokuments durch einen XML-Parser die Codierungsdeklaration des XML-Dokuments auch mit der tatsächlichen Codierung des Dokuments konsistent ist.

In den folgenden Abschnitten werden die Sachverhalte zu diesen Überlegungen erläutert, wie Sie mit eventuell auftretenden Problemen umgehen können und wie XML Extender und DB2 UDB Codepages unterstützen, wenn Dokumente vom Client an den Server und an die Datenbank übergeben werden.

Terminologie für XML-Codepages

In den Abschnitten zu XML-Codepages werden die folgenden Termini verwendet:

Dokumentcodierung
Die Codepage eines XML-Dokuments.

Dokumentcodierungsdeklaration
Der Name der in der XML-Deklaration angegebenen Codepage. Beispielsweise gibt die folgende Codierungsdeklaration `ibm-1047` an:

```
<?xml version="1.0" encoding="ibm-1047"?>
```

Konsistentes Dokument
Ein Dokument, in dem die Codepage mit der Codierungsdeklaration übereinstimmt.

Inkonsistentes Dokument
Ein Dokument, in dem die Codepage nicht mit der Codierungsdeklaration übereinstimmt.

Registervariable (Umgebungsvariable) DB2CODEPAGE
Gibt die Codepage der Daten an, die von einer Datenbankclientanwendung an DB2 UDB übergeben werden. DB2 UDB ruft die Client-Codepage aus der Ländereinstellung des Clientbetriebssystems ab, wenn diese Variable nicht gesetzt ist. Für DB2 überschreibt dieser Wert die Ländereinstellung des Clientbetriebssystems, 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 Clientbetriebssystems.

Server-Codepage oder Codepage der Ländereinstellung des Serverbetriebssystems
Die Ländereinstellung des Betriebssystems, auf dem die DB2 UDB-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 Betriebssystems auf dem Server als Standardwert.

Annahmen zur DB2- und XML Extender-Codepage

Wenn DB2 UDB ein XML-Dokument sendet oder empfängt, wird die Codierungsdeklaration nicht geprüft. Stattdessen wird die Codepage für den Client geprüft, um festzustellen, ob sie der Codepage der Datenbank entspricht. Falls die Codepages unterschiedlich sind, setzt DB2 UDB die Daten im XML-Dokument um, damit es zu einer Übereinstimmung mit der Codepage folgender Komponenten kommt:

- Der Codepage der Datenbank beim Importieren des Dokuments oder eines Teils davon in eine Datenbanktabelle.
- Der Codepage der Datenbank beim Zerlegen eines Dokuments in eine oder mehrere Datenbanktabellen.
- Der Codepage des Clients beim Exportieren des Dokuments aus der Datenbank und beim Übergeben des Dokuments an den Client.
- Der Codepage des Servers beim Verarbeiten einer Datei mit einer UDF, die Daten in eine Datei im Dateisystem des Servers zurückgibt.

Annahmen zum Importieren eines XML-Dokuments

Wenn ein XML-Dokument in die Datenbank importiert wird, wird es normalerweise zur Speicherung in einer XML-Spalte oder zum Zerlegen für eine XML-Objektgruppe importiert. Dabei werden Element- und Attributinhalt als DB2 UDB-Daten gespeichert. Wenn ein Dokument importiert wurde, konvertiert DB2 UDB die Dokumentcodierung in die Codierung der Datenbank. DB2 UDB geht davon aus, dass für das Dokument die Codepage verwendet wird, die in der Spalte „Ausgangs-Codepage" in der Tabelle unten angegeben ist. In Tabelle 93 sind die Konvertierungen zusammengefasst, die DB2 UDB beim Importieren eines XML-Dokuments vornimmt.

Tabelle 93. UDFs und gespeicherte Prozeduren verwenden, wenn die XML-Datei in die Datenbank importiert wird

Task	Quellen-Codepage für die Konvertierung	Ziel-Codepage für die Konvertierung	Kommentare
DTD-Datei in DTD_REF-Tabelle einfügen	Client-Codepage	Datenbank-Codepage	
Mit gespeicherten Prozeduren eine Spalte aktivieren oder eine Objektgruppe aktivieren bzw. Verwaltungsbefehle verwenden, die DAD-Dateien importieren	Client-Codepage (die Codepage, die während der Installation zum Binden von DXXADMIN verwendet wird)	Datenbank-Codepage	

Tabelle 93. UDFs und gespeicherte Prozeduren verwenden, wenn die XML-Datei in die Datenbank importiert wird (Forts.)

Task	Quellen-Codepage für die Konvertierung	Ziel-Codepage für die Konvertierung	Kommentare
Benutzerdefinierte Funktionen verwenden: • XMLVarchar FromFile() • XMLCLOB FromFile() • Content(): Abrufen von XMLFILE in ein CLOB	Server-Codepage	Datenbank-Codepage	Die Datenbank-Codepage wird in die Client-Codepage konvertiert, wenn die Daten für den Client dargestellt werden.
Gespeicherten Prozeduren für die Zerlegung verwenden	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.

Annahmen zum Exportieren eines XML-Dokuments

Wenn ein XML-Dokument aus der Datenbank exportiert wird, wird es aufgrund einer Clientanforderung exportiert, eines der folgenden Objekte darzustellen:

- Ein XML-Dokument aus einer XML-Spalte
- Die Abfrageergebnisse von XML-Dokumenten in einer XML-Spalte
- Ein zusammengesetztes XML-Dokument aus einer XML-Objektgruppe

Wenn ein Dokument exportiert wird, konvertiert DB2 UDB die Dokumentcodierung in die Codierung des Clients oder Servers, in Abhängigkeit davon, wo die Anforderung generiert wurde und wohin die Daten übergeben werden sollen. In Tabelle 94 sind die Konvertierungen zusammengefasst, die DB2 UDB beim Exportieren eines XML-Dokuments vornimmt.

Tabelle 94. UDFs und gespeicherte Prozeduren verwenden, wenn die XML-Datei aus der Datenbank exportiert wird

Task	DB2 konvertiert die Datei von ... in ...	Kommentare
Benutzerdefinierte Funktionen verwenden: • XMLFileFromVarchar() • XMLFileFromCLOB() • Content(): Abrufen von XMLVARCHAR in eine externe Serverdatei	Datenbankcodepage in die Servercodepage	

Tabelle 94. UDFs und gespeicherte Prozeduren verwenden, wenn die XML-Datei aus der Datenbank exportiert wird (Forts.)

Task	DB2 konvertiert die Datei von ... in ...	Kommentare
XML-Dokumente mit einer gespeicherten Prozedur zusammensetzen, die in einer Ergebnistabelle gespeichert sind, die abgefragt und exportiert werden kann.	Datenbankcodepage in Client-Codepage, wenn die Ergebnismenge dem Client dargestellt wird	• Beim Zusammensetzen von Dokumenten kopiert XML Extender die vom Befehl in der DAD angegebene Codierungsdeklaration in das neu erstellte Dokument. Sie muss bei der Darstellung mit der Client-Codepage übereinstimmen.

Überlegungen zur Codierungsdeklaration für XML Extender

Die *Codierungsdeklaration* gibt die Codepage der Codierung des XML-Dokuments an. Sie erscheint in der XML-Deklarationsanweisung. Beim Verwenden von XML Extender muss unbedingt sichergestellt werden, dass die Codierung des Dokuments der Codepage des Clients bzw. Servers entspricht. Dabei spielt die Speicherposition der Datei eine Rolle.

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. Wenn Sie die in Tabelle 95 auf Seite 323 aufgelisteten Codierungen verwenden, kann Ihre Anwendung über verschiedene IBM Betriebssysteme hinweg portiert werden. 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.
- **Kategorie** gibt das Betriebssystem an, für das DB2 UDB die angegebene Codepage unterstützt.
- **Codepage** zeigt die von IBM definierte Codepage zu der angegebenen Codierung an.

Tabelle 95. Von XML Extender unterstützte Codierungsdeklarationen

Kategorie	Codierung	Codepage
Unicode	UTF-8	1208
	UTF-16	1200
ASCII	iso-8859-1	819
	ibm-1252	1252
	iso-8859-2	912
	iso-8859-5	915
	iso-8859-6	1089
	iso-8859-7	813
	iso-8859-8	916
	iso-8859-9	920
MBCS	gb2312	1386
	ibm-932, shift_jis78	932
	Shift_JIS	943
	IBM-eucCN	1383
	ibm-1388	1388
	IBM-eucJP, EUC-JP	954, 33722
	ibm-930	930
	ibm-939	939
	ibm-1390	1390
	ibm-1399	1399
	ibm-5026	5026
	ibm-5035	5035
	euc-tw, IBM-eucTW	964
	ibm-937	937
	euc-kr, IBM-eucKR	970
	big5	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" auf Seite 324 sind die Folgen inkompatibler Codierungen beschrieben. Unter der folgenden Webadresse 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.

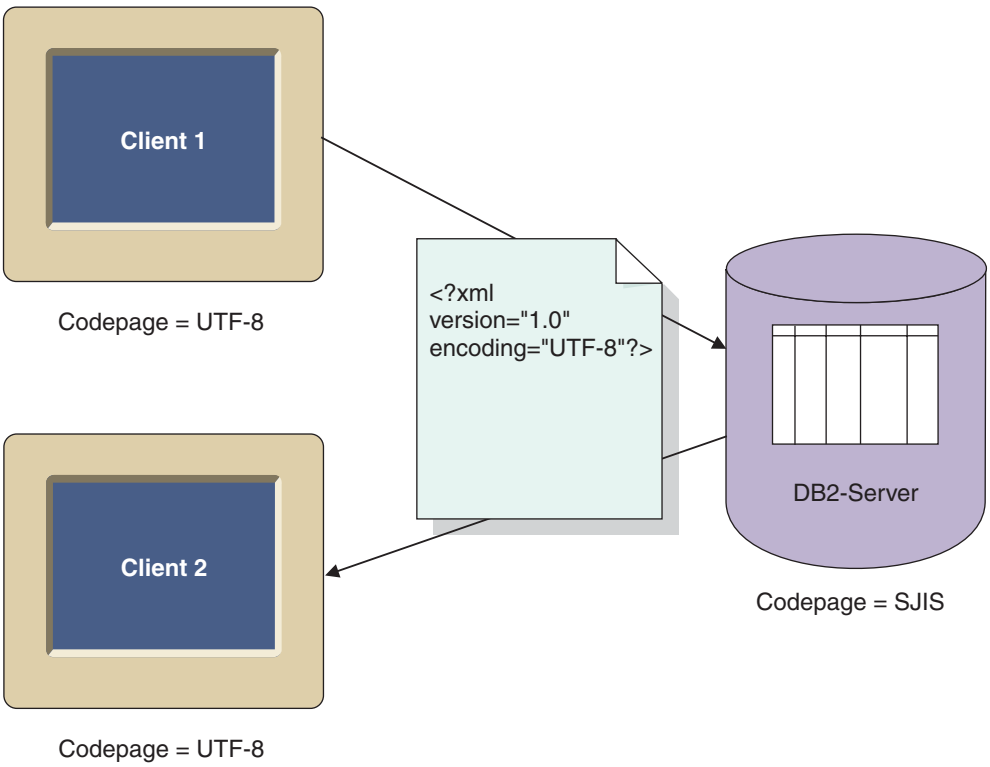


Abbildung 19. Die Clients haben übereinstimmende Codepages

Die Verwendung unterschiedlicher Codepages kann zu folgenden Situationen führen:

- Es können Umsetzungen durchgeführt werden, bei denen Daten verloren gehen, insbesondere wenn die Quellen-Codepage Unicode ist und die Ziel-Codepage eine andere Codepage. Unicode enthält den vollständigen Zeichensatz. 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 verloren gehen.
- 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 auf Seite 325 zeigt eine Umgebung, in der die Codepages der Clients nicht konsistent sind.

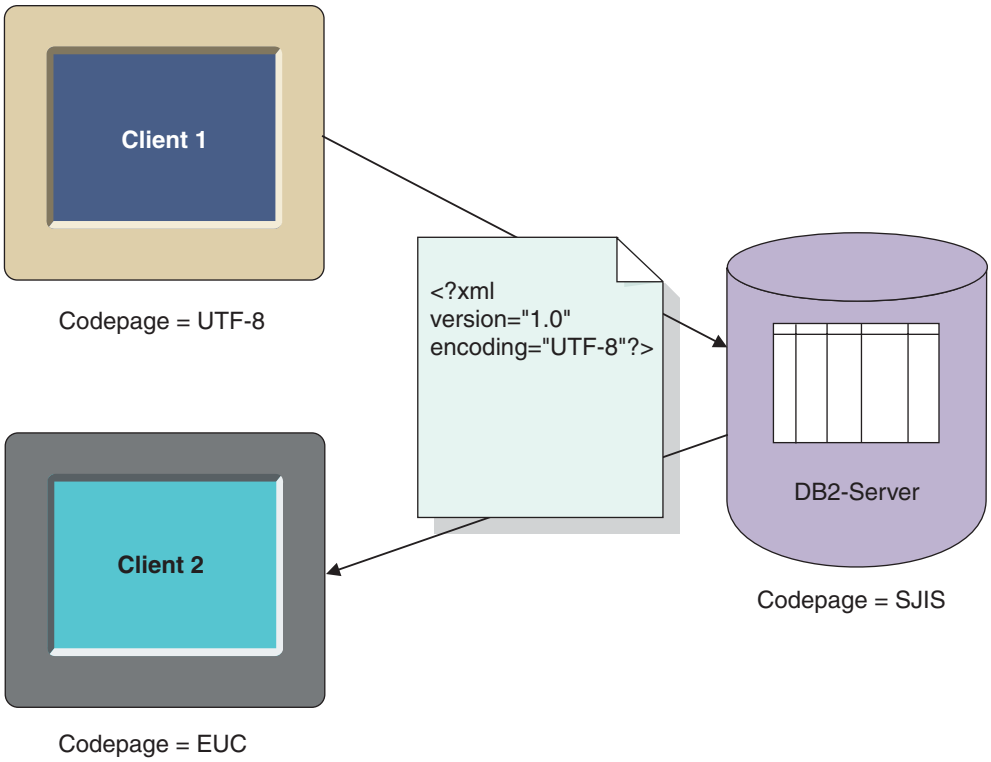


Abbildung 20. Die Codepages der Clients stimmen nicht überein.

Client 2 empfängt das Dokument in EUC, das Dokument hat jedoch die Codierungsdeklaration UTF-8.

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

XML Extender verarbeitet XML-Dokumente in folgenden Fällen:

- Beim Speichern und Abrufen von XML-Spaltendaten mit der XML-Spalten-speicherungs- 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 UDB-Konvertierung und mit einem vom Server importierten Dokument. Die Dokumentcodierungsdeklaration ist UTF-8, der Server ist UTF-8 und die Datenbank ist UTF-8. DB2 UDB muss das Dokument nicht konvertieren, da die Server-Codepage und die Datenbankcodepage identisch sind. Die Codierung und die Deklaration sind konsistent.

1. Das Dokument wird über die UDF XMLClobfromFile in DB2 UDB importiert.
2. Das Dokument wird auf den Server extrahiert.

Szenario 2: Dieses Szenario ist eine Konfiguration mit konsistenten Codierungen, DB2 UDB-Konvertierung und mit einem vom Server importierten und auf den Client exportierten Dokument. Die Dokumentcodierung und -deklaration ist SJIS, die Client- und Server-Codepages ist jeweils SJIS und die Datenbank-Codepage ist UTF-8.

1. Das Dokument wird mit der UDF XMLClobfromfile vom Server in DB2 UDB importiert. DB2 konvertiert das Dokument von SJIS und speichert es in UTF-8. Die Codierungsdeklaration und die Codierung sind in der Datenbank nicht konsistent.
2. Ein Client, der SJIS verwendet, fordert das Dokument zur Darstellung im Webbrowser an. DB2 UDB 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 UDB-Konvertierung und mit einem vom Server importierten und auf einen Client exportierten Dokument. Die Dokumentcodierungsdeklaration ist SJIS für das ankommende Dokument. Die Server-Codepage ist SJISibm-1047, und der Client und die Datenbank verwenden UTF-8.

1. Das Dokument wird über eine Speicherungs-UDF in die Datenbank importiert. DB2 UDB konvertiert das Dokument von SJIS in UTF-8. Die Codierung und die Deklaration sind inkonsistent.
2. Ein Client mit einer Codepage UTF-8 fordert das Dokument zur Darstellung in einem Webbrowser an. DB2 führt keine Konvertierung durch, da die Client- und die Datenbank-Codepages identisch sind. Die Dokumentcodierung und Deklaration sind inkonsistent, da die Deklaration SJIS angibt 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 UDB-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.

Das Dokument wird über die UDF XMLClobfromFile in DB2 UDB importiert. DB2 UDB 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 UDB ermöglicht es dem Client jedoch, die Codepage mit `db2set DB2CODEPAGE=1208` 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 mit der UDF Content(), die eine XML-Datei zurückgibt, von DB2 UDB exportiert. DB2 UDB 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.

Empfehlungen zur Vermeidung inkonsistenter XML-Dokumente

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 Datenverlust oder unbrauchbare XML-Dokumente 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 (unter der Annahme, dass XML Extender die Datei in der Server-Codepage in das Dateisystem auf dem Server exportiert hat):
  - 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 von XML Extender eine der folgenden Techniken aus (unter der Annahme, dass der Client die Ergebnistabelle abfragt, 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 eines Windows NT-Clients: 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 XMLFromFile zum Importieren der XML-Dokumente müssen Dokumente in der Codepage entsprechend der Ländereinstellung des Serverbetriebssystems 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 UDB 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 UDF 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 Serverbetriebssystems codiert; dies kann nicht UTF-8 sein.

Anhang C. Grenzwerte für XML Extender

In diesem Abschnitt werden die Grenzwerte für folgende Komponenten beschrieben:

- XML Extender-Objekte
- Werte, die von benutzerdefinierten Funktionen zurückgegeben werden
- Parameter für gespeicherte Prozeduren
- Spalten in Tabellen zur Verwaltungsunterstützung
- Zusammensetzung und Zerlegung

Die folgende Tabelle beschreibt die Grenzwerte für XML Extender-Objekte.

Tabelle 96. Grenzwerte für XML Extender-Objekte

Objekt	Grenzwert
Maximale Anzahl an Zeilen in einer Tabelle in einer XML-Objektgruppe für Zerlegung	10240 Zeilen aus jedem zerlegten XML-Dokument
Maximale Anzahl an Byte in einem XML-Dateipfadnamen, der als Parameterwert angegeben wird	512 Byte
Länge des Elementes sql_stmt in einer DAD-Datei für SQL-Zusammensetzung	Windows- und UNIX-Betriebssysteme: 32.765 Byte. OS/390- und iSeries-Betriebssysteme: 16.380 Byte.
Maximale Anzahl an Spalten für eine Tabelle, angegeben für eine Tabelle in der DAD-Datei für die RDB_node-Zerlegung	500 Spalten (Spalten für eine Tabelle) werden durch text_node- und attribute_node-Elemente in einer DAD-Datei angegeben.

Die folgende Tabelle beschreibt die Grenzwerte für Werte, die durch benutzerdefinierte Funktionen von XML Extender zurückgegeben werden.

Tabelle 97. Grenzwerte für Werte von benutzerdefinierten Funktionen (UDFs)

Von der UDF zurückgegebene Werte	Grenzwert
Maximale Anzahl an Byte, die von der UDF extractCHAR zurückgegeben werden	254 Byte
Maximale Anzahl an Byte, die von der UDF extractCLOB zurückgegeben werden	2 Gigabyte
Maximale Anzahl an Byte, die von der UDF extractVARCHAR zurückgegeben werden	4 Kilobyte

Die folgende Tabelle beschreibt die Grenzwerte für Parameter von gespeicherten Prozeduren von XML Extender.

Tabelle 98. Grenzwerte für Parameter von gespeicherten Prozeduren

Parameter für gespeicherte Prozedur	**Grenzwert**
Maximale Größe eines CLOB für ein XML-Dokument[1]	1 MB
Maximale Größe eines CLOB einer DAD (Document Access Definition, Dokumentzugriffsdefinition)[1]	100 KB
Maximale Größe von *collectionName*	30 Byte
Maximale Größe von *colName*	30 Byte
Maximale Größe von *dbName*	8 Byte
Maximale Größe von *defaultView*	128 Byte
Maximale Größe von *rootID*	30 Byte
Maximale Größe von *resultTabName*	18 Byte
Maximale Größe von *tablespace*	8 Byte
Maximale Größe von *tbName*[2]	18 Byte
Maximale Größe von *resultColumn*	30 Byte
Maximale Größe von *validColumn*	30 Byte
Maximale Größe von *varchar_value*	32672 Byte

Anmerkungen:

1. Diese Größe kann für dxxGenXMLClob und dxxRetrieveXMLCLOB geändert werden.
2. Wenn der Wert des Parameters *tbName* über einen Schemanamen angegeben ist, darf der gesamte Name (einschließlich des Trennzeichens) nicht länger als 128 Zeichen sein.

Die folgende Tabelle beschreibt die Grenzwerte für die Tabelle DB2XML.DTD_REF.

Tabelle 99. Grenzwerte für XML Extender

Spalten der Tabelle DB2XML.DTD_REF	**Grenzwert**
Größe der Spalte AUTHOR	128 Byte
Größe der Spalte CREATOR	128 Byte
Größe der Spalte UPDATOR	128 Byte
Größe der Spalte DTDID	128 Byte
Größe der Spalte CLOB	100 KB

Namen können erweitert werden, wenn DB2 UDB sie von der Client-Codepage in die Datenbankcodepage konvertiert. Ein Name kann auf dem Client innerhalb der Größenbegrenzung liegen und dennoch den Grenzwert überschreiten, wenn die gespeicherte Prozedur den konvertierten Namen erhält.

Die folgende Tabelle beschreibt die Grenzwerte für die Zusammensetzung und Zerlegung.

Tabelle 100. Grenzwerte von XML Extender für Zusammensetzung und Zerlegung

Objekt	Grenzwert
Maximale Anzahl an Zeilen, die in eine Tabelle in einer XML-Objektgruppe für Zerlegung eingefügt werden	10240 Zeilen aus jedem zerlegten XML-Dokument
Maximale Länge des Namensattributs in elements_node- oder attribute_node-Elementen innerhalb einer DAD	63 Byte
Maximale Anzahl an Byte in einem XMLFile-Pfadnamen, der als Parameterwert angegeben wird	512 Byte

Umgebungsvariable DB2DXX_MIN_TMPFILE_SIZE:

XML Extender stellt große Dokumente in temporäre Dateien, damit während der Verarbeitung nicht zu viel Speicherplatz belegt wird. Auf Systemen mit großem physischem Speicher kann das Verschieben von Dokumenten in temporäre Dateien umgangen werden, wodurch die Menge der Eingabe- und Ausgabeaktivitäten reduziert wird. Für die Verarbeitung von Dokumenten, die kleiner als der angegebene Wert sind, veranlasst die Umgebungsvariable DB2DXX_MIN_TMPFILE_SIZE XML Extender zur Verwendung von Speicherpuffern an Stelle von temporären Dateien. Die Variable kann nur für den Server und nicht für einen Client verwendet werden. Wenn mehrere physische Knoten an einer Partition mit mehreren Knoten teilhaben, ist die Variable möglicherweise auf jedem Knoten unterschiedlich gesetzt und gibt genau die Speicherkapazität jeder einzelnen Maschine wieder. Wenn die Umgebungsvariable nicht gesetzt ist, werden Dokumente, die größer als 128 KB sind, während der Verarbeitung automatisch in temporären Dateien gespeichert. Dokumente, die kleiner als 128 KB sind, werden hingegen im Hauptspeicher verarbeitet.

Anhang D. UDT- und UDF-Namen für XML Extender

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 und -UDTs ist DB2XML. In dieser Dokumentation wird nur auf den Funktionsnamen verwiesen.

Wenn Sie den Schemanamen zum Funktionspfad hinzufügen, können Sie UDTs und UDFs ohne Schemanamen angeben. Der Funktionspfad ist eine geordnete Liste von Schemanamen. DB2 UDB verwendet die Reihenfolge von Schemanamen in der Liste, um Verweise auf Funktionen und UDTs aufzulösen. Sie können den Funktionspfad durch Angabe der SQL-Anweisung SET CURRENT FUNCTION PATH angeben. Durch diese Anweisung wird der Funktionspfad im speziellen Register CURRENT FUNCTION PATH festgelegt.

Empfehlung: Fügen Sie den Schemanamen DB2XML zum Funktionspfad hinzu. Durch Hinzufügen dieses Schemanamens können Sie XML Extender-UDF- und -UDT-Namen ohne das Qualifizierungsmerkmal DB2XML eingeben. Das folgende Beispiel zeigt, wie Sie das Schema DB2XML zum Funktionspfad hinzufügen:

```
SET CURRENT FUNCTION PATH = DB2XML, CURRENT FUNCTION PATH
```

Einschränkung: Fügen Sie DB2XML nicht als erstes Schema im Funktionspfad hinzu, wenn Sie sich unter der Benutzer-ID DB2XML anmelden. DB2XML wird automatisch als erstes Schema definiert, wenn Sie sich als DB2XML anmelden. Wenn Sie DB2XML als erstes Schema im Funktionspfad hinzufügen, empfangen Sie eine Fehlerbedingung, da der Funktionspfad mit zwei Schemata DB2XML beginnt.

XML Extender-Glossar

A

Abfrageobjekt. Ein Objekt, das die Merkmale, Merkmalswerte und Merkmalsgewichtungen für eine QBIC-Abfrage angibt. Das Objekt kann benannt und zur späteren Verwendung in einer QBIC-Abfrage gespeichert werden. Gegensatz zu Abfragezeichenfolge.

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.

Aktivieren. Das Vorbereiten einer Datenbank, einer Texttabelle oder einer Textspalte zur Verwendung durch XML Extender.

Analysieren. Berechnen numerischer Werte für die Merkmale eines Bilds und zum Hinzufügen der Werte zu einem QBIC-Katalog.

Anfangs-element_node. (Anfangselementknoten) Eine Darstellung des Stammelements des XML-Dokuments in der DAD.

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) Die DB2 UDB-Extender bieten APIs zum Anfordern von benutzerdefinierten Funktionen, Verwaltungsoperationen, Anzeigeoperationen und Überwachungfunktionen für die Änderung von Videoszenen. DB2 Net Search Extender bietet APIs zum Anfordern von benutzerdefinierten Funktionen, Verwaltungsoperationen und Serviceoperationen zum Abrufen von Informationen. 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.

Auslöser. Ein Mechanismus, der einer *Protokolltabelle* automatisch Informationen über die zu indexierenden Dokumenten hinzufügt, sobald ein Dokument hinzugefügt, geändert oder aus einer Textspalte gelöscht wird.

Auslöser. Die Definition einer Gruppe von auszuführenden Aktionen beim Ändern einer Tabelle. Auslöser können verwendet werden, um Aktionen wie beispielsweise das Überprüfen von Eingabedaten, das automatische Generieren eines Werts für neu eingefügte Zeilen, das Lesen aus anderen Tabellen für Querverweise oder das Schreiben in andere Tabellen zu Prüfzwecken verwendet werden. Auslöser werden häufig zur Integritätsprüfung oder zum Erzwingen von Geschäftsregeln verwendet.

B

B-Baumstruktur-Indexierung. Das native Indexierungsschema der DB2 UDB-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.

Befehlszeilenprozessor. Ein Programm mit dem Namen DB2TX, das folgende Aufgaben erfüllt:

Ermöglicht die Eingabe von Net Search Extender-Befehlen

Verarbeitet die Befehle

Zeigt das Ergebnis an

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.

Benutzerdefinierte Funktion (User-Defined Function, UDF). Ein von einem DB2-Benutzer erstellte SQL-Funktion, im Gegensatz zu einer von DB2 bereitgestellten SQL-Funktion. Net Search Extender bietet Suchfunktionen in Form von UDFs, wie beispielsweise CONTAINS.

Benutzerdefinierte Funktion (User-Defined Function, UDF). Eine Funktion, die von einem Benutzer für DB2 definiert wird. Sobald die Funktion definiert ist, kann sie in SQL-Abfragen und Videoobjekten verwendet werden. UDFs können beispielsweise erstellt werden, um das Komprimierungsformat für Videodaten abzurufen oder die Abtastrate von Audiodaten zurückzugeben. Auf diese Weise kann das Verhalten von Objekten eines bestimmten Typs definiert werden.

Benutzerdefinierter eindeutiger Typ (User-defined Distinct Type, UDT). Ein von einem DB2-Benutzer erstellter Datentyp, im Gegensatz zu einem von DB2 UDB bereitgestellten Datentyp, wie beispielsweise LONG VARCHAR.

Benutzerdefinierter Typ (User-Defined Type, UDT). Ein nicht nativer Datentyp im Datenbankmanager; er wurde von einem Benutzer erstellt. Siehe *eindeutiger Typ*.

Benutzerdefinierter Typ (User-Defined Type, UDT). Ein Datentyp, der von einem Benutzer für DB2 definiert wird. UDTs werden verwendet, um LOBs voneinander zu unterscheiden. Es kann beispielsweise ein UDT für Bildobjekte erstellt werden und ein anderer für Audioobjekte. Bild- und Audioobjekte werden zwar als BLOBs gespeichert, aber als Typen behandelt, die sich von BLOBs und voneinander unterscheiden.

Benutzertabelle. Eine Tabelle, die für eine Anwendung erstellt wurde und von dieser verwendet wird.

Bild. In diesem Zusammenhang eine elektronische Darstellung eines Bilds.

Boolesche Suche. Eine Suche, bei der ein oder mehrere Suchbegriffe mit Booleschen Operatoren kombiniert werden.

Browser. Eine Net Search Extender-Funktion, die das Anzeigen von Text auf einem Computerbildschirm ermöglicht-.Siehe *Web-Browser*.

C

CCSID. Kennung für codierten Zeichensatz (Coded Character Set Identifier).

CLOB. Großes Zeichenobjekt (Character Large Object).

Codepage. Eine Zuordnung von Schriftzeichen und Steuerfunktionen zu allen Codepunkten. Beispiel: Die Zuordnung der Zeichen und Bedeutungen zu den 256 Codepunkten bei einem 8-Bit-Code.

D

DAD. Siehe *Dokumentzugriffsdefinition*.

Dateireferenzvariable. Eine Programmierungsvariable, die beim Verschieben eines LOB in und von einer Client-Workstation hilfreich ist.

Datenaustausch. Die gemeinsame Benutzung von Daten über verschiedene Anwendungen hinweg. XML unterstützt den Austausch von Daten, ohne dass diese zunächst in ein geeignetes Format umgewandelt werden müssen.

Datenbank-Partitions-Server. Verwaltet eine *Datenbank-Partition*. Ein Datenbank-Partitions-Server besteht aus einem Datenbankmanager und der Objektgruppe von Daten und Systemressourcen, die er verwaltet. Normalerweise ist jeder Maschine ein Datenbank-Partitions-Server zugeordnet.

Datenbankpartition. Ein Teil der Datenbank, der aus eigenen Benutzerdaten, Indizes, Konfigurationsdateien und Transaktionsprotokollen besteht. Manchmal auch als Knoten oder Datenbankknoten bezeichnet.

Datenquelle. Ein lokaler oder ferner relationaler oder nicht relationaler Datenmanager, der Datenzugriffe über einen ODBC-Treiber unterstützt, der wiederum die ODBC-APIs unterstützt.

Datenstrom. Von einer API-Funktion zurückgegebene Informationen aus Text (mindestens ein Abschnitt) mit dem gesuchten Begriff sowie Informationen zur Hervorhebung des gefundenen Begriffs in diesem Text.

Datentyp. Ein Attribut von Spalten und Literalen.

DBCLOB. Großes Doppelbyte-Zeichenobjekt (Double-byte Character Large Object).

DBCS. Unterstützung von Doppelbytezeichen.

Dokument. Siehe *Textdokument*.

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_REF-Tabelle. DTD-Referenztabelle.

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-Repository. Eine DB2 UDB-Tabelle mit dem Namen DTD_REF. Jede Zeile dieser Tabelle stellt eine DTD mit zusätzlichen Metadateninformationen dar.

Durchsuchen. Anzeigen von Text auf einem Computerbildschirm.

E

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.

Erweitern. Das Hinzufügen weiterer aus einem Thesaurus abgeleiteten Begriffe zu einem Suchbegriff.

Escape-Zeichen. Ein Zeichen, das angibt, dass das folgende Zeichen nicht als *Platzhalterzeichen* interpretiert werden soll.

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.

Extensible 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. Ein Textdokument in Form einer im Dateisystem des Betriebssystems gespeicherten Datei und nicht in Form einer Zelle in einer Tabelle unter der Steuerung von DB2. Eine Datei, die in einem von DB2 aus gesehen externen Dateisystem vorhanden ist.

F

Fremdschlüssel. Ein Schlüssel, der Teil der Definition einer referenziellen Integritätsbedingung ist und aus einer oder mehreren Spalten einer abhängigen Tabelle besteht.

Funktion. Siehe *Zugriffsfunktion*.

G

Gebundene Suche. Eine Suche in koreanischen Dokumenten, bei der die Wortgrenzen berücksichtigt werden.

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.

Gigabyte (GB). Eine Milliarde (10^9) Byte. Bei der Angabe der Speicherkapazität 1 073 741 824 Byte.

Großes Binärobjekt (Binary Large Object, BLOB). Eine binäre Zeichenfolge, deren Länge bis zu 2 GB betragen kann. Bild-, Audio- und Videoobjekte werden in der DB2-Datenbank als BLOB gespeichert.

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 einer DB2 UDB-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 UDB-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.

H

Host-Variable. Eine Variable in einem Anwendungsprogramm, auf die in einer eingebetteten SQL-Anweisung verwiesen werden kann. Host-Variablen sind der primäre Mechanismus zur Übertragung von Daten zwischen einer Datenbank und den Arbeitsbereichen von Anwendungsprogrammen.

I

Inaktivieren. Das Wiederherstellen einer Datenbank, einer Texttabelle oder einer Textspalte in den Zustand vor seiner Aktivierung für XML Extender. Hierzu werden die beim Aktivierungsprozess erstellten Elemente entfernt.

Index. Das Extrahieren signifikanter Begriffe aus dem Text und das Speichern dieser Begriffe in einem *Textindex*. 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.

J

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.

K

Katalogsicht. Eine von Net Search Extender erstellte Systemtabelle zu Verwaltungszwecken. Eine Katalogsicht enthält Informationen zu den Tabellen und Spalten, die zur Verwendung mit Net Search Extender aktiviert wurden.

Kilobyte (KB). Eintausend (10^3) Byte.Bei der Angabe der Speicherkapazität 1024 Byte.

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.

Korrekt formatiertes Dokument. Ein XML-Dokument, das keine DTD enthält. Obwohl es die XML-Spezifikation erfüllt, muss ein Dokument mit einer gültigen DTD außerdem korrekt formatiert sein.

L

Linguistischer Index. Ein *Textindex* mit Begriffen, die durch eine linguistische Verarbeitung auf ihre Basis reduziert wurden. Der Begriff "Mäuse" wird beispielsweise als "Maus" indexiert. Siehe auch *präziser Index*, *Ngram-Index* und *dualer Index*.

LOB. Großes Objekt.

LOB-Zeigereinheit. Ein kleiner (4 Byte), in einer Host-Variablen gespeicherter Wert, der in einem Programm zum Verweisen auf ein wesentlich größeres LOB in einer DB2 UDB-Datenbank verwendet werden kann. Durch die Verwendung einer LOB-Zeigereinheit kann ein Benutzer das LOB bearbeiten, als wäre es in einer regulären Host-Variablen gespeichert; das LOB muss also nicht zwischen der Anwendung auf der Clientmaschine und dem Datenbankserver transportiert werden.

Logischer Knoten. Ein Knoten auf einem Prozessor, wenn diesem Prozessor mehrere Knoten zugeordnet sind.

Lokales Dateisystem. Ein in DB2 vorhandenes Dateisystem.

M

Megabyte (MB). Eine Million (10^6) Byte. Bei der Angabe der Speicherkapazität 1 048 576 Byte.

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*.

N

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

O

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*.

P

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 referenziellen Integritätsbedingung.

Prozedur. Siehe *gespeicherte Prozedur*.

Q

QBIC-Katalog. Ein Repository, das Daten zu den visuellen Merkmalen von Bildern enthält.

R

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. 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 Nebentabellen der Anwendungstabelle zuordnet.

S

SBCS. Unterstützung von Einzelbytezeichensätzen.

Schema. Eine Objektgruppe von Datenbankobjekten wie beispielsweise Tabellen, Sichten, Indizes oder Auslösern. Ein Schema bietet eine logische Klassifizierung von Datenbankobjekten.

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 UDB-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. 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 Nebentabellen 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 Zuordnung zwischen einem XML-Element bzw. -Attribut und einer DB2 UDB-Spalte beim Definieren des Indexierungsschemas für die XML-Spalte verwendet. Darüber hinaus wird der Standortpfad von Net Search 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 Werte von in der Anweisung angegebenen Hostvariablen ändern können.

Struktureller Textindex. Indexieren von Textschlüsseln entsprechend der Baumstruktur des XML-Dokuments mit DB2 UDB Net Search Extender.

Suchargument. Die bei der Durchführung einer Suche angegebenen Bedingungen; sie bestehen aus einem oder mehreren Begriffen und Suchparametern.

Szenenkatalog. Eine Datenbanktabelle oder Datei, in der Daten zu Szenen wie beispielsweise die Nummer des Start- und Enderahmens der Szene in einem Videoclip gespeichert werden. Ein Benutzer kann eine Sicht einer Tabelle über eine SQL-Abfrage aufrufen oder auf die Daten in der Datei zugreifen.

T

Tabellen zur Verwaltungsunterstützung. Eine Tabelle, mit der ein DB2 UDB-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.

Tabellen zur Verwaltungsunterstützung. Eine der Tabellen, mit denen ein DB2 UDB-Extender Benutzeranfragen zu Bild-, Audio- oder Videoobjekten 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. Auch als *Metadatentabelle* bezeichnet.

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.

Terabyte. Eine Billion (10^{12}) Byte. Zehn hoch 12 Byte. Bei der Angabe der Speicherkapazität 1 099 511 627 776 Byte.

text_node. (Textknoten) Eine Darstellung des CDATA-Texts eines Elements.

Texttabelle. Eine DB2 UDB-Tabelle, die *Textspalten* enthält.

Trace. Speichern von Informationen in einer Datei, die später zum Suchen der Ursache eines Fehlers verwendet werden kann.

U

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, wie beispielsweise http://www.ibm.com/software /data/db2/extenders.

Überladene Funktion. Ein Funktionsname, für den mehrere Funktionsexemplare vorhanden sind.

V

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 UDB-Sicht, die Tabellen miteinander verknüpft.

Verwaltung. Die Task zum Vorbereiten von Textdokumenten zum Suchen, Verwalten von Indizes und zum Abrufen von Statusinformationen.

Video. Bezieht sich auf den sichtbaren Teil der aufgezeichneten Informationen.

Videoclip. Ein Abschnitt eines Films oder Videobands.

Videoindex. Eine Datei, über die der Video Extender eine bestimmte *Szene* oder einen Rahmen in einem Videoclip findet.

W

Web-Browser. Ein Clientprogramm, das Anfragen an einen Web-Server startet und die vom Server zurückgegebenen Informationen anzeigt.

wildcard character. sinnloser Eintrag, da im Dt. beides 'Platzhalterzeichen' See *masking character*

X

XML. eXtensible Markup Language.

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-Attribut. Ein beliebiges durch ATTLIST im XML-Element in der DTD angegebenes Attribut. 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. 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-Spalte. Eine Spalte in der Anwendungstabelle, die für die UDTs von XML Extender aktiviert wurde.

XML-Tabelle. Eine Anwendungstabelle, die eine oder mehrere XML Extender-Spalten enthält.

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

XML-UDT. Ein benutzerdefinierter DB2 UDB-Typ, der von 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.

Z

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 UDB-Datenbank über zwei wesentliche Zugriffs- und Speichermethoden zu: XML-Spalten und XML-Objektgruppen. Siehe auch *XML-Spalte* und *XML-Objektgruppe*.

Zugriffsfunktion. Eine vom Benutzer bereitgestellte Funktion, die den Datentyp des in einer Spalte gespeicherten Texts in einen Typ umwandelt, der von Net Search Extender verarbeitet werden kann.

Zuordnungsschema. Eine Definition, die festlegt, wie XML-Daten in einer relationalen Datenbank dargestellt werden. Das Zuordnungsschema wird in der DAD-Datei angegeben. 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.

Bemerkungen

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. An Stelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Dienstleistungen in Verbindung mit Fremdprodukten und Fremddienstleistungen liegt beim Kunden, soweit nicht ausdrücklich solche Verbindungen erwähnt sind.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Europe, Director of Licensing, 92066 Paris La Defense Cedex, France.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Garantie, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

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 zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten der IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden, Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält 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 werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

Folgende Namen sind in gewissen Ländern Marken der International Business Machines Corporation und wurden in mindestens einem der Dokumente in der DB2 UDB-Dokumentationsbibliothek verwendet:

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
IBM System AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
IBM System /390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Folgende Namen sind in gewissen Ländern Marken oder eingetragene Marken anderer Unternehmen und wurden in mindestens einem der Dokumente in der DB2 UDB-Dokumentationsbibliothek verwendet.

Microsoft, Windows, Windows NT und das Windows-Logo sind in gewissen Ländern Marken der Microsoft Corporation.

Intel und Pentium sind in gewissen Ländern Marken der Intel Corporation.

Java und alle auf Java basierenden Marken sind in gewissen Ländern Marken von Sun Microsystems, Inc.

UNIX ist in gewissen Ländern eine eingetragene Marke von The Open Group.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken anderer Unternehmen sein.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Kontaktaufnahme mit IBM

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0190 7 72243 erreichen Sie die DB2 Helpline, wo Sie Antworten zu DB2-spezifischen Problemen erhalten.

Informationen zur nächsten IBM Niederlassung in Ihrem Land oder Ihrer Region finden Sie im IBM Verzeichnis für weltweite Kontakte, das Sie im Web unter http://www.ibm.com/planetwide abrufen können.

Produktinformationen

Informationen zu DB2 Universal Database-Produkten erhalten Sie telefonisch oder im World Wide Web unter http://www.ibm.com/software/data/db2/udb.

Diese Site enthält die neuesten Informationen zur technischen Bibliothek, zum Bestellen von Büchern, zu Produktdownloads, Newsgroups, FixPaks, Neuerungen und Links auf verfügbare Webressourcen.

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180 5 5090 können Sie Handbücher telefonisch bestellen.

Informationen dazu, wie Sie sich mit IBM in Verbindung setzen können, finden Sie auf der globalen IBM Internet-Seite unter folgender Adresse:
www.ibm.com/planetwide

SC12-3062-01