IBM® DB2® Universal Database

XML Extender
Administración y programación

Versión 7

IBM® DB2® Universal Database

XML Extender
Administración y programación

Versión 7

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general incluida en el apartado “Avisos” en la página 297. .

Esta publicación es la traducción del original inglés *IBM DB2 Universal Database XML Extender Administration and Programming Version 7*.

Contenido

Tablas

Acerca de este manual

La presente sección describe lo siguiente:

A quién va dirigido este manual

Este manual está pensado para las personas siguientes:

- Personas que trabajan con datos XML en aplicaciones DB2 y que están familiarizados con los conceptos de XML. Los lectores de esta publicación deben tener un conocimiento general de XML y DB2. Para aprender más acerca de XML y temas afines, consulte el sitio Web siguiente:

  http://www.w3c.org/XML

  Para aprender más acerca de DB2, consulte el sitio Web siguiente:

  http://www.ibm.com/software/data/db2/library
- Administradores de bases de datos DB2 que conocen los conceptos, herramientas y técnicas referentes a la administración de DB2.
- Programadores de aplicaciones DB2 que conocen SQL y uno o más lenguajes de programación que se pueden utilizar para aplicaciones DB2.

Cómo obtener una versión actual de este manual

Puede obtener la última versión de este manual en el sitio Web del XML Extender:

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

Cómo utilizar este manual

Este manual está estructurado de la manera siguiente:

Parte 1. Introducción

Esta parte proporciona una visión general del XML Extender e indica cómo puede utilizarlo en sus aplicaciones comerciales. Contiene un caso hipotético de iniciación que le ayuda a comenzar a utilizar el producto.

Parte 2. Administración
Esta parte describe cómo preparar y mantener una base de datos DB2 para datos XML. Lea esta parte si necesita administrar una base de datos DB2 que contiene datos XML.

Parte 3. Programación
Esta parte describe cómo gestionar sus datos XML. Lea esta parte si necesita acceder y manejar datos XML en un programa de aplicación DB2.

Parte 4. Consulta
Esta parte describe cómo utilizar los mandatos de administración del XML Extender, los tipos definidos por el usuario (UDT), las funciones definidas por el usuario (UDF) y los procedimientos almacenados. También lista los mensajes y códigos que emite el XML Extender. Lea esta parte si está familiarizado con los conceptos y tareas del XML Extender, pero necesita información sobre un tipo definido por el usuario (UDT), una función definida por el usuario (UDF), un mandato, un mensaje, tablas de metadatos, tablas de control o código.

Parte 5. Apéndices
Los apéndices describen el DTD utilizado en la definición de acceso a documento para los ejemplos y el caso hipotético de iniciación, y otros productos XML de IBM.

Novedades de este manual para DB2 UDB Versión Fixpak 2

Este manual contiene información nueva o modificada sobre:

- Puesta a punto e inicio del asistente de administración
- Cómo la UDF Update modifica documentos XML
- Cómo la UDF XMLClob devuelve resultados
- Cómo aumentar los límites de CLOB para procedimientos almacenados
- Cambios en los requisitos de DAD:
  - Inclusión de las instrucciones de proceso de la hoja de estilos cuando se componen documentos XML nuevos
  - Utilización de una condición que falta o está vacía para el primer nodo RDB cuando sólo se ha especificado una tabla.
- Cómo convertir marcadores de parámetros con JDBC
- Trabajo con páginas de códigos:
  - Declaraciones de codificaciones permitidas
  - Premisas sobre la conversión
  - Escenarios sobre la conversión
- Apéndice de los límites de parámetros de XML Extender

Si desea obtener información sobre los arreglos de todos los defectos correspondientes a este fixpak, consulte el archivo readme.

Consulte la página de soporte del sitio web de XML Extender para ver nuestras FAQ (Frequently Asked Questions- Preguntas más habituales) y la lista de problemas conocidos para obtener más información sobre arreglos y soluciones a las preguntas más habituales:
http://www.ibm.com/software/data/db2/extenders/xmlext/support.html

Inclusión de este manual en el Centro de Información de DB2 UDB Versión 7

La documentación de XML Extender puede incluirse en el Centro de Información de DB2 siguiendo estos pasos:

- Copie los archivos HTML de este manual, desde el subdirectorio `DOC` correspondiente a su idioma de la instalación del producto de XML, al subdirectorio del directorio de documentación de DB2 UDB de su idioma:

  Para Windows, el directorio del Centro de Información es `sqllib\doc\html\db2sx`

  Para UNIX, el directorio del Centro de Información es *`install directory`*`/doc/html/db2sx`
- Reinicie el Centro de Información y este manual quedará incluido en el separador **Manuales**.

Convenios de resaltado

Este manual utiliza los convenios siguientes:

Negrita	El texto en negrita indica: • Mandatos • Nombres de campos • Nombres de menús • Pulsadores
Cursiva	El texto en cursiva indica: • Parámetros variables que han de ser sustituidos por un valor • Palabras resaltadas • La primera aparición de un término del glosario
MAYÚSCULAS	Las letras mayúsculas indican: • Tipos de datos • Nombres de columnas • Nombres de tablas

`Ejemplo`	El texto de ejemplo indica: • Mensajes del sistema • Valores que escribe el usuario • Ejemplos de codificación • Nombres de directorios • Nombres de archivos • Vías de acceso

Cómo leer los diagramas de sintaxis

A lo largo de todo este manual, la sintaxis de los mandatos y de las sentencias de SQL se describe utilizando diagramas de sintaxis.

Para leer los diagramas de sintaxis, siga estas directrices:

- Lea el diagrama de sintaxis de izquierda a derecha, de arriba a abajo, siguiendo el recorrido de la línea.

  El símbolo ▸▸— indica el comienzo de una sentencia.

  El símbolo —▸ indica que la sintaxis de la sentencia continúa en la línea siguiente.

  El símbolo ▸— indica que la sentencia continúa de la línea anterior.

  El símbolo —▸◂ indica el final de una sentencia.

  Los diagramas de unidades sintácticas que no son sentencias completas comienzan con el símbolo ▸— y terminan con el símbolo —▸.
- Los elementos obligatorios de la sintaxis aparecen en la línea horizontal (la ruta principal).

  ▸▸—*elemento_obligatorio*—▸◂

- Los elementos opcionales aparecen debajo de la ruta principal.

  ▸▸—*elemento_obligatorio*—▸◂
  └*elemento_opcional*┘

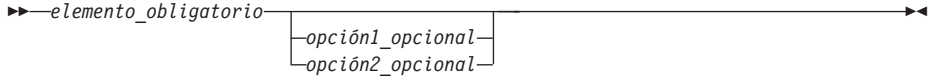
  Si un elemento opcional aparece encima de la ruta principal, ese elemento no afecta a la ejecución de la sentencia y se utiliza sólo por razones de legibilidad.

  ┌*elemento_opcional*┐
  ▸▸—*elemento_obligatorio*—▸◂

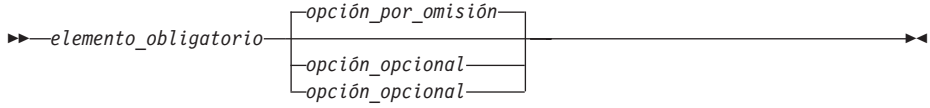
- Si puede elegir entre dos o más elementos, aparecen apilados verticalmente.

Si *debe* seleccionar uno de los elementos, un elemento de la pila aparece en la ruta principal.

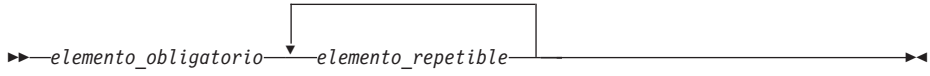
Si es opcional elegir uno de los elementos, la pila completa aparece debajo de la ruta principal.



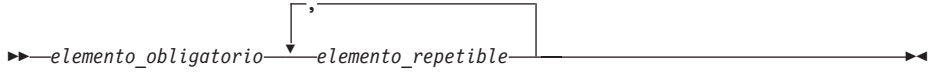
Si uno de los elementos es la opción por omisión, aparece encima de la ruta principal y las demás opciones aparecen debajo.



- Una flecha que vuelve hacia la izquierda, encima de la línea principal, indica un elemento que puede repetirse.



Si la flecha de repetición contiene un signo de puntuación, el usuario debe separar los elementos repetidos utilizando el signo de puntuación especificado.



Una flecha de repetición encima de una pila indica que se pueden repetir los elementos de la pila.

- Las palabras clave aparecen en mayúsculas (por ejemplo, `FROM`). En el XML Extender, pueden utilizarse indistintamente mayúsculas y minúsculas para las palabras clave. Los términos que no son palabras clave aparecen en letras minúsculas (por ejemplo *nombre_columna*). Estos términos representan nombres o valores que proporciona el usuario.
- Si aparecen signos de puntuación, paréntesis, operadores aritméticos u otros símbolos de esta clase, debe especificarlos como parte de la sintaxis.

Información afín

Los documentos siguientes pueden serle útiles cuando utilice el XML Extender y productos afines:

Documento	Número de pedido	Descripción
Call Level Interface Guide and Reference	SC09–2950	Este manual describe cómo escribir programas de aplicación utilizando CLI para acceder a servidores DB2.
DB2 Application Development Guide	SC09–2949	Este manual describe el proceso de desarrollo de aplicaciones y cómo codificar, compilar y ejecutar programas de aplicación que hacen uso del SQL incorporado y de las API para acceder a la base de datos.
Página de DB2 Extender	N/D	Esta página contiene información sobre los DB2 Extenders y las técnicas aplicables a los extensores. La dirección Web de la página correspondiente a los DB2 Extenders es: http://www.software.ibm.com/data/db2/extenders
DB2 Universal Database Consulta de SQL Partes 1 y 2	• Parte 1: SC10–3497 • Parte 2: SC10–3549	Estos manuales describen la sintaxis, la semántica y las reglas del lenguaje SQL. También incluyen información sobre incompatibilidades entre releases, límites del producto y vistas de catálogo.
• *DB2 Universal Database Administration Guide: Implementation* • *DB2 Universal Database Administration Guide: Performance* • *DB2 Universal Database Administration Guide: Planning*	• Implementation: SC09–2944 • Performance: SC09–2945 • Planning: SC09–2946	Estos manuales describen cómo diseñar, implementar y mantener una base de datos DB2.

Documento	Número de pedido	Descripción
DB2 Universal Database Image, Audio, and Video Extenders Administration and Programming	SC26–9929	Este manual describe cómo administrar una base de datos DB2 para datos de imagen, de audio y de vídeo. También describe cómo utilizar las API (interfaces de programación de aplicaciones) proporcionadas por los extensores para acceder y manejar estos tipos de datos.
DB2 Universal Database Text Extender Administration and Programming	SC26–9930	Este manual describe cómo administrar una base de datos DB2 para datos de tipo texto. También describe cómo utilizar las API (interfaces de programación de aplicaciones) proporcionadas por los extensores para acceder y manejar estos tipos de datos.

Parte 1. Introducción

Esta parte proporciona una visión general del XML Extender e indica cómo puede utilizarlo en sus aplicaciones comerciales.

Capítulo 1. Nociones preliminares sobre el XML Extender

La familia de productos IBM® DB2® Extenders™ proporciona soluciones de gestión de datos y metadatos para el manejo de datos tradicionales y no tradicionales. El XML Extender le ayuda a combinar la potencia de DB2 Universal Database (DB2 UDB)™ de IBM con la flexibilidad de XML.

El XML Extender de DB2 proporciona la posibilidad de almacenar documentos XML y acceder a ellos o de generar documentos XML a partir de datos relacionales existentes y fragmentar (descomponer, almacenando contenido de elementos o atributos sin codificar) documentos XML en datos relacionales. El XML Extender proporciona nuevos tipos de datos, funciones y procedimientos almacenados para gestionar los datos XML en DB2.

El XML Extender está disponible en los siguientes sistemas operativos:

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

Documentos XML

Existen muchos programas de aplicación en la industria informática, cada uno con sus cualidades y defectos. Hoy en día, el usuario puede elegir la aplicación que sea más apropiada para sus tareas. Sin embargo, debido a que a menudo se comparten datos entre aplicaciones diferentes, el usuario debe afrontar continuamente el problema de replicar, transformar, exportar o guardar los datos utilizando un formato diferente que se pueda importar a otra aplicación. Esto puede ser un problema crítico en las aplicaciones comerciales, pues muchos de estos procesos de transformación tienden a eliminar parte de los datos, o como mínimo obligan al usuario a realizar el tedioso proceso de asegurar la coherencia de los datos. Esto supone una pérdida de tiempo y dinero.

Actualmente, una de las formas de abordar este problema es que el programador de aplicaciones escriba aplicaciones *ODBC (Open Database Connectivity)* para guardar los datos en un sistema de gestión de bases de datos. A partir de este momento, los datos se pueden manejar y presentar en la forma que sea necesaria para otra aplicación. Es necesario que las aplicaciones de base de datos conviertan los datos en un formato requerido

para una aplicación determinada, pero de todos modos las aplicaciones cambian con rapidez y quedan desfasadas. Las aplicaciones que convierten datos a HTML proporcionan soluciones de presentación, pero los datos presentados prácticamente no se pueden utilizar para ningún otro propósito. Si existiera otro método que separara los datos de la presentación, este método podría utilizarse como una forma práctica de intercambio entre aplicaciones.

XML ha surgido para hacer frente a este problema. XML es un acrónimo de *eXtensible Markup Language*. Es un lenguaje ampliable en cuanto que el propio lenguaje es un metalenguaje que permite al usuario crear su propio lenguaje de acuerdo con las necesidades de su empresa. Puede utilizar XML para capturar no sólo los datos que necesita una aplicación determinada, sino también la estructura de datos. XML no es el único formato de intercambio de datos. Pero, XML ha surgido como el estándar aceptado. Las aplicaciones que se ajustan a este estándar pueden finalmente compartir datos entre ellas sin tener que transformar datos que utilizan formatos patentados.

Aplicaciones XML

Debido a que XML es ahora el estándar aceptado para el intercambio de datos, están surgiendo muchas aplicaciones que podrán sacar provecho de él.

Suponga que está utilizando una determinada aplicación de gestión de proyectos y desea compartir algunos datos de esa aplicación con una aplicación de agenda. Mediante XML, esto se puede hacer con facilidad. En el mundo interconectado de hoy en día, un proveedor de aplicaciones no podrá competir a menos que en sus aplicaciones integre programas de utilidad para el intercambio de datos con XML. Por lo tanto, en este ejemplo, la aplicación de gestión de proyectos puede exportar tareas en XML, que luego se pueden importar tal cual a la aplicación de agenda (si la información se ajusta a una DTD aceptada).

¿Por qué XML y DB2?

Aunque XML soluciona muchos problemas al proporcionar un formato estándar para el intercambio de datos, existen todavía otros problemas para superar. Cuando se crea una aplicación de datos para una empresa, es necesario plantearse cuestiones tales como las siguientes:

- ¿Con qué frecuencia es necesario replicar los datos?
- ¿Qué clase de información es necesario compartir entre aplicaciones?
- ¿Con qué rapidez se puede buscar la información necesaria?
- ¿Cómo hacer que una acción determinada, tal como la adición de una nueva entrada, desencadene un intercambio automático de datos entre todas las aplicaciones?

Esta clase de cuestiones sólo se pueden abordar utilizando un sistema de gestión de bases de datos. Mediante la incorporación directa de la información y meta-información de XML en la base de datos, se pueden obtener directamente (y más rápidamente) los resultados XML que las demás aplicaciones del usuario necesitan para su fin determinado. Es aquí donde el XML Extender puede ayudarle. Con el XML Extender, puede sacar provecho de la potencia de DB2 en muchas aplicaciones XML.

La colocación de documentos XML estructurados en una base de datos DB2 permite combinar información XML estructurada con datos relacionales tradicionales. Basándose en la aplicación, puede elegir si desea almacenar documentos XML completos en DB2, en forma de tipo de datos no tradicional definido por el usuario, o si desea correlacionar el contenido XML como datos tradicionales en tablas relacionales. Para los tipos de datos XML no tradicionales, el XML Extender añade la posibilidad de buscar diversos tipos de datos para valores de elementos o atributos de XML, además de la búsqueda estructural de texto proporcionada por el DB2 UDB Text Extender.

Mediante el XML Extender, la aplicación del usuario puede:

- Almacenar documentos XML completos como *datos de columna* en una tabla de aplicación o externamente como archivo local y al mismo tiempo extraer los valores deseados de elementos o atributos XML para formar *tablas auxiliares* con fines de búsqueda. Utilizando el método de columna XML, se puede:
  - Realizar búsquedas rápidas para elementos o atributos XML de *tipos de datos* generales de SQL que se han extraído para formar tablas auxiliares y se han indexado
  - Actualizar el contenido de un *elemento XML* o el valor de un *atributo XML*
  - Extraer elementos o atributos XML de forma dinámica utilizando consultas SQL
  - Validar documentos XML durante la inserción y actualización
  - Realizar una búsqueda de texto estructural con Text Extender
- Componer o descomponer el contenido de documentos XML con una o más tablas relacionales, utilizando el método de almacenamiento y acceso de colección XML

Integración de XML en DB2

El XML Extender proporciona las siguientes características para ayudarle a gestionar y a explotar datos XML con DB2:

- Las herramientas de administración le ayudan a gestionar la integración de datos XML en tablas relacionales
- Métodos de almacenamiento y de utilización para los datos XML.

- Un depósito DTD para almacenar las DTD que se utilizan para validar datos XML: DTD_REF
- Un esquema de correlación llamado archivo DAD (Definición de acceso a documento) para correlacionar documentos XML con datos relacionales

Herramientas de administración

Las herramientas de administración del XML Extender le ayudan a habilitar la base de datos y las columnas de tabla para XML y a correlacionar datos de XML con estructuras relacionales de DB2. El XML Extender proporciona varias herramientas de administración para el usuario, que le permiten desarrollar una aplicación para realizar tareas de administración o simplemente utilizar un asistente (programa de ayuda). Puede utilizar las herramientas siguientes para realizar tareas de administración para el XML Extender:

- Los asistentes de administración del XML Extender proporcionan una interfaz gráfica del usuario para tareas de administración.
- El mandato **dxxadm** proporciona una opción de línea de mandatos para tareas de administración.
- Los procedimientos almacenados de administración del XML Extender proporcionan opciones de desarrollo de aplicaciones para tareas de administración.

Métodos de almacenamiento y acceso

El XML Extender proporciona dos métodos de almacenamiento y acceso para la integración de documentos XML en DB2: columna XML y colección XML. Estos métodos tienen usos muy diferentes, pero se pueden utilizar en la misma aplicación.

columna XML

Este método le ayuda a almacenar documentos XML inalterados en DB2. La columna XML es un método adecuado para archivar documentos. Los documentos se insertan en columnas que están habilitadas para XML y se pueden actualizar, recuperar y también se puede buscar en ellos. Los datos de elementos y atributos se pueden correlacionar con tablas DB2 (tablas auxiliares), que a su vez se pueden indexar para una búsqueda estructural rápida.

colección XML

Este método le ayuda a correlacionar estructuras de documento XML con tablas DB2 de modo que se puedan componer documentos XML a partir de datos de DB2, o descomponer (almacenar el contenido de elementos o atributos sin codificar) documentos XML en datos DB2. Este método es el adecuado para aplicaciones de intercambio de datos, sobretodo cuando el contenido de los documentos XML se actualiza con frecuencia.

Depósito DTD

El XML Extender proporciona un *depósito de Definición de tipo de documento (DTD)*, que es un conjunto de declaraciones para elementos y atributos XML. Cuando una base de datos está *habilitada* para XML, se crea una *tabla de referencia DTD (DTD_REF)*. Cada fila de esta tabla representa una DTD, junto con información adicional de metadatos. Los usuarios pueden acceder a esta tabla para insertar sus propias DTD. Las *DTD* de la tabla DTD_REF se utilizan para validar documentos XML.

Definiciones de acceso a documento (DAD)

La forma en que los documentos XML estructurados deben manejarse se especifica en una *definición de acceso a documento (DAD)*. La propia DAD es un documento con formato XML. La DAD asocia la estructura de documento XML con una base de datos cuando se utilizan columnas XML o colecciones XML. La estructura de la DAD es distinta cuando se define una columna XML y cuando se define una colección XML.

Los archivos DAD se manejan utilizando la tabla XML_USAGE, que se crea cuando se habilita una base de datos para XML.

Columna XML: Almacenamiento y recuperación de documentos estructurados

Debido a que XML contiene toda la información necesaria para crear un conjunto de documentos, habrá ocasiones en que deseará almacenar y mantener la estructura del documento tal como es actualmente.

Por ejemplo, una empresa de divulgación de noticias por la Web puede desear mantener un archivo de los artículos publicados. En este caso, el XML Extender le permite almacenar los artículos XML, completos o parciales, en una columna de una tabla DB2. Este tipo de almacenamiento de documentos XML se denomina *columna XML*, tal como se muestra en la Figura 1 en la página 8.

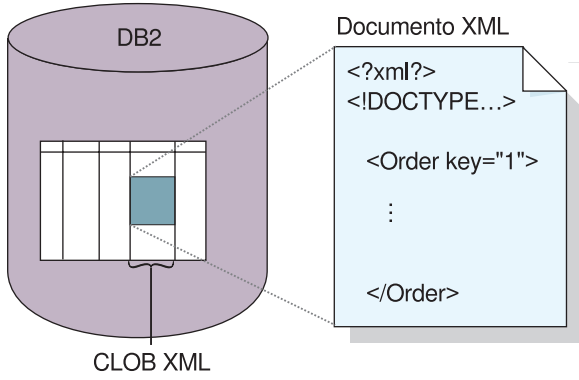


Figura 1. Almacenamiento de documentos XML estructurados en una columna de una tabla DB2

El XML Extender proporciona los siguientes tipos definidos por el usuario (UDT) para su utilización con columnas XML:

- XMLVarchar
- XMLCLOB
- XMLFILE

Todos los UDT del XML Extender tienen el prefijo db2xml, que es el *nombre de esquema* de las UDT del XML Extender de DB2. Estos tipos de datos se utilizan para identificar el tipo de almacenamiento de los documentos XML en la tabla de aplicación. El XML Extender da soporte a archivos planos preexistentes; no es necesario almacenar los documentos XML dentro de DB2. También puede almacenar documentos XML como archivos en el *sistema de archivos local*, de acuerdo con lo especificado por un nombre de archivo local.

El XML Extender de DB2 proporciona potentes *funciones definidas por el usuario (user-defined functions, UDF)* para almacenar y recuperar documentos XML en columnas XML, así como para extraer valores de elementos o atributos de XML. Una UDF es una función que se define para el sistema de gestión de bases de datos y a la que se puede hacer referencia en consultas SQL. El XML Extender proporciona los siguientes tipos de UDF:

- Almacenamiento: Almacena documentos XML inalterados en columnas habilitadas para XML de tipos de datos XML
- Extracción: Extrae documentos XML o los valores de los elementos y atributos especificados como tipos de base de datos
- Actualización: Actualiza documentos XML completos o los valores de los elementos y atributos especificados

Las funciones de extracción le permiten realizar búsquedas potentes en tipos de datos SQL generales. Además, puede utilizar el DB2 UDB Text Extender con el XML Extender para llevar a cabo búsquedas estructurales y *búsquedas*

totales en el texto de documentos XML. Esta potente capacidad de búsqueda se puede utilizar, por ejemplo, para mejorar las posibilidades de utilización de un sitio Web que publica grandes volúmenes de texto legible, tales como artículos de prensa, o de aplicaciones de intercambio de datos electrónicos *(Electronic Data Interchange, EDI)*, que contienen elementos o atributos de búsqueda frecuente.

Todas las UDF del XML Extender tienen el prefijo db2xml, que es el nombre de esquema de las UDF del XML Extender de DB2. Las UDF se aplican a los UDT de XML y se utilizan para columnas XML.

Vía de ubicación

Una *vía de ubicación* es una secuencia de *códigos XML* que identifican un elemento o atributo de XML. El XML Extender utiliza la vía de ubicación para identificar la estructura del documento XML, indicando el contexto para el elemento o atributo. Una vía con una barra inclinada (/) individual indica que el contexto es el documento completo. La vía de ubicación se utiliza en las situaciones siguientes:

- Para identificar los elementos y atributos que deben extraerse, cuando se extraen las UDF
- Para especificar el archivo de correlación entre un elemento o atributo XML y una columna DB2 cuando se define el esquema de indexación en la DAD para columnas XML
- Para identificar un elemento o atributo XML cuando se utiliza Text Extender para una búsqueda de texto estructural

La Figura 2 en la página 10 muestra un ejemplo de una vía de ubicación y su relación con la estructura del documento XML.

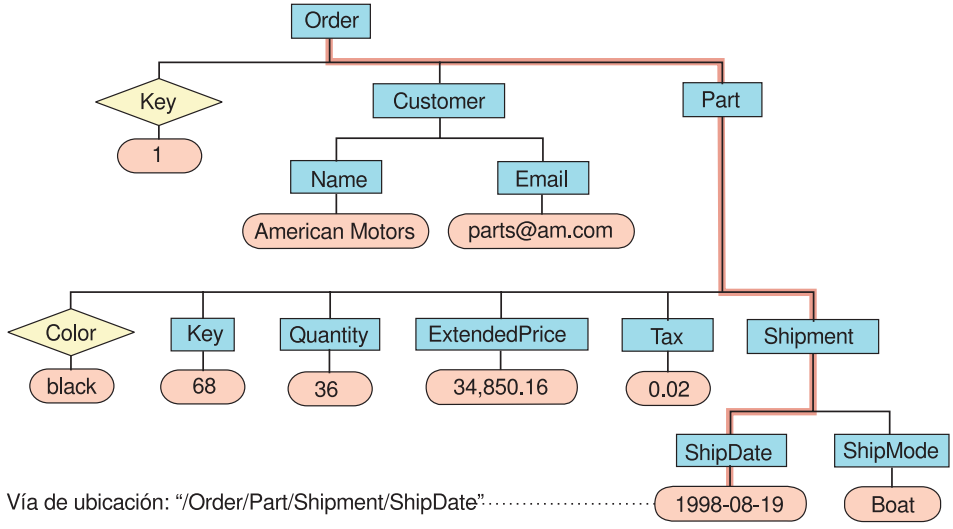


Figura 2. Almacenamiento de documentos como documentos XML estructurados en una columna de una tabla DB2

Para especificar la vía de ubicación, el XML Extender utiliza un subconjunto del *XML Stylesheet Language Transformation (XSLT)* y *XML Path Language (XPath)*. Este manual utiliza el término *vía de ubicación*, que está definido en la especificación de XPath. La vía de ubicación es una secuencia de códigos XML que identifican un elemento o atributo de XML. Este manual también utiliza la sintaxis abreviada de XSLT o XPath de la *vía de ubicación absoluta*, que se especifica en las especificaciones de XPath. La vía de acceso absoluta es la vía de acceso completa de un objeto.

XSLT es un lenguaje para transformar documentos XML en otros documentos XML. Está diseñado para ser utilizado como parte del *XML Stylesheet Language (XSL)*, que es un lenguaje de hoja de estilo para XML. Además de XSLT, XSL incluye un vocabulario XML para especificar información de formato. XSL especifica el diseño de un documento XML utilizando XSLT para describir cómo se transforma el documento en otro documento XML que hace uso del vocabulario de formato.

XPath es un lenguaje para direccionar partes de un documento XML y está diseñado para ser utilizado por XSLT. Cada vía de ubicación se puede expresar utilizando la sintaxis definida para XPath.

Para obtener más información sobre XSLT y XPath, vea las páginas Web siguientes:

- Para XSLT, vea: http://www.w3.org/TR/WD-xslt
- Para XPath, vea: http://www.w3.org/TR/xpath

Vea el apartado "Vía de ubicación" en la página 55 para conocer la sintaxis y las limitaciones.

Terminología para columnas XML

Esta sección describe conceptos y terminología de XML que se utilizan en este manual.

definición de acceso a documento (document access definition, DAD)
Para una columna XML es una correlación de estructura de documento XML con tablas auxiliares DB2 indexadas para consultas estructurales.

DXX_INSTALL
Es el directorio de instalación del XML Extender.

tabla auxiliar
Son tablas adicionales que el XML Extender crea para mejorar el rendimiento cuando se buscan elementos o atributos en una columna XML.

columna XML
Es un método para almacenar y acceder a documentos XML mediante la habilitación de una columna DB2 para tipos de datos XML y el almacenamiento de un documento XML inalterado en la columna habilitada. También hace referencia a una columna que se ha habilitado para XML, utilizando una de las herramientas de administración.

tabla XML
Es una tabla de aplicación que incluye una o más columnas que se han habilitado para XML, utilizando una de las herramientas de administración.

UDF de XML
Es una función de DB2 definida por el usuario que proporciona el XML Extender.

UDT de XML
Es un tipo DB2 definido por el usuario que proporciona el XML Extender.

Colección XML: Gestión de datos integrados

Los datos de SQL tradicionales se *descomponen* a partir de documentos XML de entrada o se utilizan para *componer* documentos XML de salida. Si tiene datos que deben compartirse con otras aplicaciones, puede que desee poder componer y descomponer los documentos XML de entrada y salida, y gestionar los datos según convenga para sacar provecho de las posibilidades relacionales de DB2. Este tipo de almacenamiento de documentos XML se denomina *colección XML*.

Un ejemplo de una colección XML se proporciona en la Figura 3 .

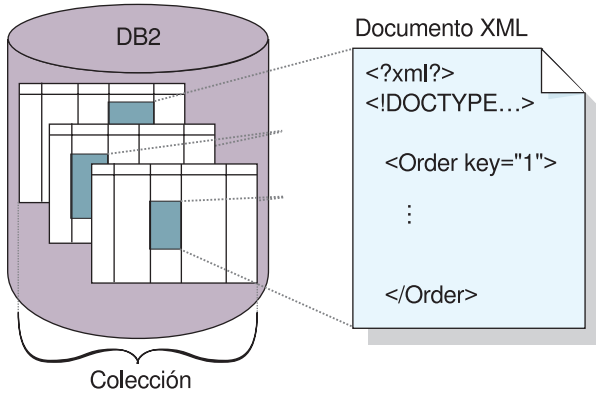


Figura 3. Almacenamiento de documentos como datos no codificados en tablas DB2

La colección XML se define en un archivo DAD, que especifica cómo se correlacionan los elementos y atributos con una o más tablas relacionales. Se puede definir un nombre de colección habilitándolo y, a continuación, utilizándolo con procedimientos almacenados para componer o descomponer documentos XML.

Cuando se define una colección en el archivo DAD, se utiliza uno de los dos tipos de esquemas de correlación, correlación SQL o correlación de nodo_RDB. La correlación SQL utiliza sentencias SELECT de SQL para definir los usos de las condiciones y las tablas de DB2 para la colección. La correlación de nodo_RDB utiliza un nodo_RDB basado en XPath para definir las tablas, columnas y condiciones.

Los procedimientos almacenados se proporcionan para componer o descomponer documentos XML. Los procedimientos almacenados utilizan el prefijo db2xml, que es el *nombre de esquema* del XML Extender. Utilice los siguientes procedimientos almacenados con colecciones XML:

- Composición:
  - dxxGenXML(): utiliza un archivo DAD de una colección XML para componer documentos XML
  - dxxRetrieveXML(): utiliza una colección XML habilitada para componer documentos XML
- Descomposición:
  - dxxShredXML(): utiliza un archivo DAD de una colección XML para descomponer documentos XML

– dxxInsertXML(): utiliza una colección XML habilitada para descomponer documentos XML

Terminología para colecciones XML

Los términos siguientes son específicos del XML Extender y aparecen a menudo en el presente manual.

composición
Es la generación de documentos XML a partir de datos relacionales existentes, tal como se define en un archivo DAD.

descomposición
Es el almacenamiento de documentos XML como datos relacionales sin codificar, tal como se se define en un archivo DAD.

definición de acceso a documento (document access definition, DAD)
Para una colección XML, es una correlación de estructuras de documento XML con estructuras de datos DB2 para componer o descomponer documentos XML.

DXX_INSTALL
Es el directorio de instalación del XML Extender.

colección XML
Es un método para almacenar y acceder a datos XML utilizando un conjunto de tablas relacionales. Los datos sin codificar se pueden componer para formar documentos XML o se pueden descomponer a partir de documentos XML. Este término también designa el conjunto de tablas utilizado en la composición o descomposición de documentos XML.

procedimientos almacenados XML
Son procedimientos almacenados para componer o descomponer documentos XML.

Capítulo 2. Iniciación al XML Extender

Este capítulo muestra cómo iniciarse en la utilización del XML Extender para acceder a datos XML y modificarlos para las aplicaciones. Siguiendo las lecciones de guía de aprendizaje que se proporcionan, puede configurar una base de datos utilizando los datos de muestra proporcionados, correlacionar datos SQL con un documento XML, almacenar documentos XML en la base de datos y, a continuación, buscar datos en documentos XML y extraerlos.

En las lecciones de administración, utilice la Ventana de mandatos de DB2 con mandatos de administración. Puede llevar a cabo estas tareas con el asistente de administración del XML Extender, que también se describe en este manual. En las lecciones de gestión de datos XML, utilizará las UDF y los procedimientos almacenados proporcionados por el XML Extender. La mayoría de ejemplos del resto del manual se basan en los datos de muestra que se utilizan en este capítulo.

Requisito: Para realizar las lecciones de este capítulo, debe tener DB2 UDB Versión 6.1 o superior. Si utiliza la Versión 6.1, debe instalar el Fixpak 2. Además, en los pasos de estas lecciones se supone que se utiliza Windows NT®.

Las lecciones son las siguientes:

- Almacenar un documento XML inalterado en una columna de tabla DB2
  - Planificar el UDT de XML en el que se va a almacenar el documento y los elementos y atributos XML que se van a buscar con frecuencia.
  - Configurar la base de datos y las tablas
  - Habilitar la base de datos para XML
  - Insertar la DTD en la tabla de depósito de DTD
  - Preparar una DAD para una columna XML
  - Añadir una columna a una tabla existente de tipo XML
  - Habilitar la nueva columna para XML
  - Crear índices en las tablas auxiliares
  - Almacenar un documento XML en la columna XML
  - Buscar en la columna XML utilizando las UDF del XML Extender
- Crear un documento XML a partir de datos existentes
  - Planificar la estructura de datos del documento XML
  - Configurar la base de datos y las tablas
  - Habilitar la base de datos para XML

- Preparar un archivo de definición de acceso a documento (DAD) para una colección XML
- Componer el documento XML a partir de datos existentes
- Recuperar el documento XML de la base de datos

• Limpiar la base de datos

Escenario para las lecciones

En estas lecciones, se supone que el usuario trabaja para ACME Auto Direct, una compañía que distribuye coches y camiones a empresas líder en el sector automovilístico. Se le han asignado dos tareas. Primero va a configurar un sistema en el que los pedidos se puedan archivar en la base de datos SALES_DB para que el departamento de ventas los consulte. La segunda tarea consiste en obtener información de una base de datos de pedidos de compras existente, SALES_DB y extraer información clave de ésta para almacenarla en documentos XML.

Lección: Almacenar un documento XML en una columna XML

Escenario

Se le ha asignado la tarea de archivar los datos de ventas para el departamento de servicio. Los datos están almacenados en documentos XML que utilizan la misma DTD. El departamento de servicio utilizará estos documentos XML cuando trabaje con pedidos y reclamaciones de los clientes.

El departamento de servicio ha suministrado una estructura recomendada para los documentos XML y ha especificado los datos de elementos que cree que se van a consultar con más frecuencia. Desean guardar los documentos XML en la tabla SALES_TAB de la base de datos SALES_DB y desearían poder buscar en ellas con rapidez. La tabla SALES_DB tendrá dos columnas con datos sobre cada venta y una tercera columna que contendrá el documento XML. Esta columna se llama ORDER.

En primer lugar, el usuario determinará los tipos de datos XML en los que se va a almacenar el documento XML, así como los elementos y atributos XML que se van a consultar con frecuencia. A continuación, configurará la base de datos SALES_DB para XML, creará la tabla SALES_TAB y habilitará la columna ORDER para almacenar el documento inalterado en DB2. También insertará una DTD para el documento XML para validación y, después, almacenará el documento como un tipo de datos XMLVARCHAR. Cuando habilite la columna, definirá las tablas auxiliares que deben indexarse para la búsqueda estructural del documento en un archivo de definición de acceso a documento (DAD), un documento XML que especifica la estructura de las tablas auxiliares. Para ver ejemplos del archivo DAD, de la DTD y del documento XML, vea el “Apéndice B. Ejemplos” en la página 275.

SALES_TAB se describe en la Tabla 1.

Tabla 1. Tabla SALES_TAB

Nombre de columna	Tipo de datos
INVOICE_NUM	CHAR(6) NOT NULL PRIMARY KEY
SALES_PERSON	VARCHAR(20)
ORDER	XMLVARCHAR

Planificación

Antes de empezar a trabajar con el XML Extender para guardar los documentos, debe comprender la estructura del documento XML para poder determinar cómo buscar en el documento. Cuando planifique cómo buscar en el documento, debe determinar:

- El tipo XML definido por el usuario en el que se va a almacenar el documento XML
- Los elementos y atributos XML que el departamento de servicio va a buscar con frecuencia, para que se puedan indexar y así mejorar el rendimiento.

Las secciones siguientes describen cómo tomar estas decisiones.

Estructura del documento XML

La estructura del documento XML para esta lección reúne información para un pedido específico estructurada con la clave de pedido como nivel superior y, después, la información de cliente, pieza y envío en el nivel siguiente. El documento XML se describe en la Figura 4 en la página 18.

Esta lección también proporciona una DTD de ejemplo para que la utilice para comprender y validar la estructura del documento XML. Puede ver el archivo DTD en el "Apéndice B. Ejemplos" en la página 275. Coincide con la estructura de la Figura 4 en la página 18.

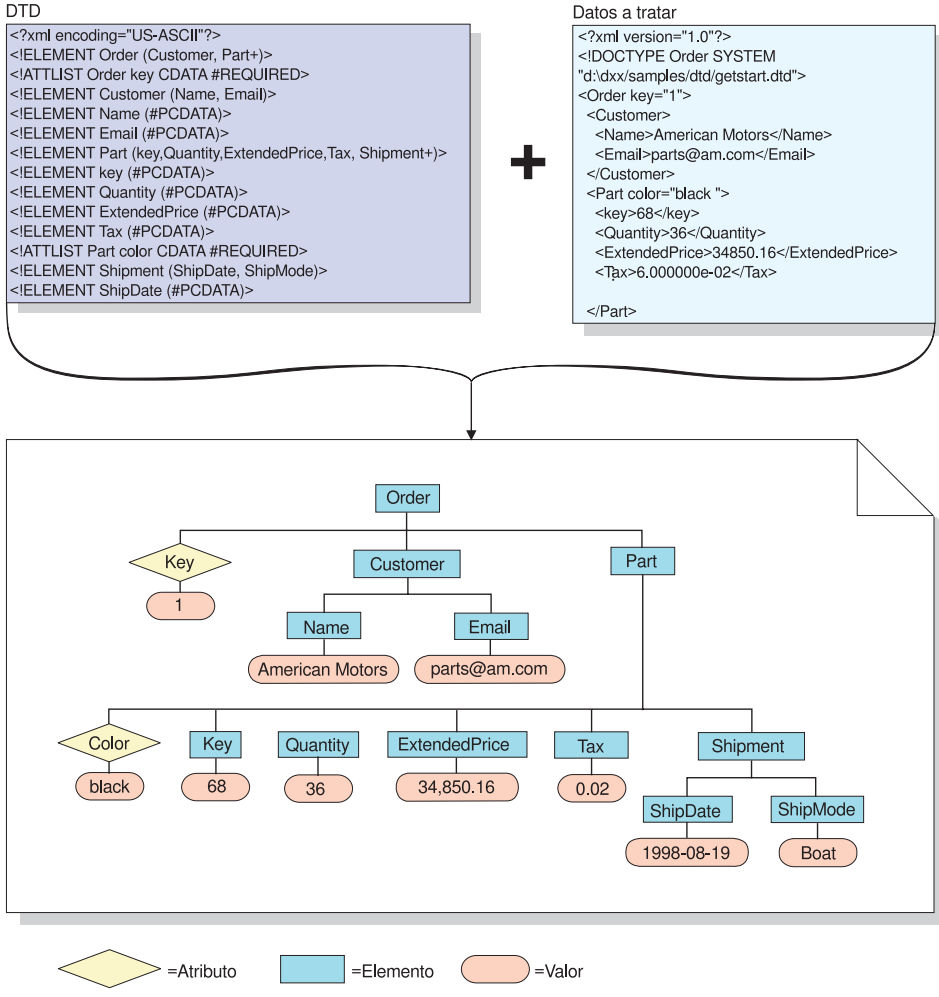


Figura 4. Estructura jerárquica de la DTD y el documento XML

Determinación del tipo de datos XML para la columna XML

El XML Extender proporciona tipos de datos de usuario XML en los que se define una columna para contener documentos XML. Estos tipos de datos son los siguientes:

- XMLVarchar: para documentos pequeños almacenados en DB2
- XMLCLOB: para documentos grandes almacenados en DB2
- XMLFILE: para documentos almacenados fuera de DB2

En esta lección, va a almacenar un documento pequeño en DB2 y, por lo tanto, utilizará el tipo de datos XMLVarchar.

Determinación de los elementos y atributos que deben buscarse

Una vez haya comprendido la estructura del documento XML y las necesidades de la aplicación, puede determinar los elementos y atributos que deben buscarse, los elementos y atributos que se buscarán o extraerán con más frecuencia o los más caros de consultar. El departamento de servicio ha indicado que va a consultar con frecuencia la clave de pedido, el nombre de cliente, el precio y la fecha de envío de un pedido, y que va a necesitar un rendimiento rápido para estas búsquedas. Esta información está contenida en los elementos y los atributos de la estructura del documento XML. La Tabla 2 describe las vías de ubicación de cada elemento y atributo.

Tabla 2. Elementos y atributos que deben buscarse

Datos	Vía de ubicación
clave de pedido	/Order/@key
cliente	/Order/Customer/Name
precio	/Order/Part/ExtendedPrice
fecha de envío	/Order/Part/Shipment/ShipDate

Correlación del documento XML con las tablas auxiliares

En esta guía de aprendizaje va a crear un archivo DAD para la columna XML, que se utiliza para almacenar el documento XML en DB2. También correlaciona el contenido de los elementos y atributos XML con las tablas auxiliares DB2 que se utilizan para la indexación, que mejora el rendimiento de las búsquedas. En la última sección, ha visto los elementos y atributos que van a buscarse. En esta sección, va a aprender más sobre cómo correlacionar los valores de los elementos y atributos con tablas DB2 que se pueden indexar.

Después de identificar los elementos y los atributos que se van a buscar, determinará cómo deberán organizarse en las tablas auxiliares, cuántas tablas existirán y las columnas que habrá en cada tabla. Por lo general, las tablas auxiliares se organizan incluyendo información similar en la misma tabla. La estructura también se determina según si la vía de ubicación de los elementos se puede repetir más de una vez en el documento. Por ejemplo, en nuestro documento, el elemento pieza se puede repetir varias veces y, por lo tanto, los elementos de precio y fecha pueden aparecer varias veces. Los elementos que pueden aparecer varias veces deben estar en sus propias tablas.

Además, también es necesario determinar los tipos base DB2 que deben utilizar los valores de los elementos y atributos. Normalmente, esto se determina fácilmente mediante el formato de los datos. Si los datos son de texto, elija VARCHAR; si los datos son de números enteros, elija INTEGER o si los datos son una fecha y desea efectuar búsquedas por rango, elija DATE.

En esta guía de aprendizaje, los elementos y los atributos se correlacionan con las siguientes tablas auxiliares:

ORDER_SIDE_TAB

Nombre de columna	Tipo de datos	Vía de ubicación	¿Aparición múltiple?
ORDER_KEY	INTEGER	/Order/@key	No
CUSTOMER	VARCHAR(16)	/Order/Customer/Name	No

PART_SIDE_TAB

Nombre de columna	Tipo de datos	Vía de ubicación	¿Aparición múltiple?
PRICE	DECIMAL(10,2)	/Order/Part/ExtendedPrice	Sí

SHIP_SIDE_TAB

Nombre de columna	Tipo de datos	Vía de ubicación	¿Aparición múltiple?
DATE	DATE	/Order/Part/Shipment/ShipDate	Sí

Para esta guía de aprendizaje, se proporciona un conjunto de scripts para que los utilice para configurar su entorno. Estos scripts se encuentran en el directorio *DXX_INSTALL*\samples\cmd (donde *DXX_INSTALL* es la unidad y el directorio en el que se ha instalado el XML Extender, por ejemplo c:\dxx\samples\cmd) y son los siguientes:

getstart_db.cmd
Crea la base de datos y rellena cuatro tablas.

getstart_prep.cmd
Enlaza la base de datos con los procedimientos almacenados del XML Extender y la CLI de DB2.

getstart_insertDTD.cmd
Inserta la DTD, que se utiliza para validar el documento XML, en la columna XML.

getstart_createTabCol.cmd
Crea una tabla de aplicación que tendrá una columna habilitada para XML.

getstart_alterTabCol.cmd
Modifica la tabla de aplicación añadiendo la columna que se habilitará para XML.

getstart_enableCol.cmd
Habilita la columna XML.

getstart_createIndex.cmd
Crea índices en las tablas auxiliares para la columna XML.

getstart_insertXML.cmd
Inserta el documento XML en la columna XML.

getstart_queryCol.cmd
Ejecuta una sentencia select en la tabla de aplicación y devuelve el documento XML.

getstart_clean.cmd
Limpia el entorno de la guía de aprendizaje.

Puesta a punto

En esta sección, se prepara la base de datos para utilizarla con el XML Extender. Va a realizar las tareas siguientes:

1. Crear la base de datos.
2. Habilitar la base de datos.

Creación de la base de datos

En esta sección se utiliza un mandato para preparar la base de datos. Este mandato crea una base de datos de ejemplo, se conecta a ella, crea las tablas que van a contener los datos y, después, inserta los datos.

Para crear la base de datos:

1. Cambie al directorio *DXX_INSTALL*\samples\cmd, donde *DXX_INSTALL* es la unidad y el directorio en los que ha instalado el XML Extender. En estas lecciones se supone que el directorio es c:\dxx. Cambie estos valores si su unidad y directorio son otros.
2. Abra la Ventana de mandatos de DB2 desde el menú Inicio de Windows NT o entre el mandato siguiente desde el indicador de mandatos de Windows NT:

   ```
   DB2CMD
   ```

3. Desde la Ventana de mandatos de DB2, ejecute el mandato siguiente:

   ```
   getstart_db.cmd
   ```

Habilitación de la base de datos

Para guardar información XML en la base de datos, debe habilitarla para el XML Extender. Cuando habilita una base de datos para XML, el XML Extender:

- Crea todos los tipos definidos por el usuario (UDT) y todas las funciones definidas por el usuario (UDF).

- Crea y rellena las tablas de control con los metadatos necesarios para el XML Extender.
- Crea el esquema db2xml y asigna los privilegios necesarios.

Para habilitar la base de datos para XML:

Desde la Ventana de mandatos de DB2, ejecute el script siguiente para habilitar las base de datos SALES_DB:

```
getstart_prep.cmd
```

Este script enlaza la base de datos con los procedimientos almacenados del XML Extender y la CLI de DB2. También ejecuta la opción del mandato **dxxadm** que habilita la base de datos:

```
dxxadm enable_db SALES_DB
```

Creación de la columna XML

El XML Extender proporciona un método para almacenar y acceder a todos los documentos XML de la base de datos, llamado columna XML. Con la utilización del método de columna XML puede almacenar el documento utilizando el tipo de archivo XMLFILE, indexar la columna en tablas auxiliares y después, consultar o buscar en el documento XML. Este método de almacenamiento es particularmente útil para las aplicaciones de archivo en las que los documentos no se actualizan con frecuencia. Para la finalidad de esta guía de aprendizaje, va a guardar en la columna XML el documento XML proporcionado.

En esta lección va a guardar el documento en la tabla SALES_TAB. Para guardar el documento:

1. Insertará la DTD para el documento XML en la tabla de referencia de DTD, DTD_REF.
2. Preparará un archivo DAD que especifique la ubicación del documento XML y las tablas auxiliares para una búsqueda estructural.
3. Añadirá una columna a la tabla SALES_TAB con un tipo XML definido por el usuario de XMLVARCHAR.
4. Habilitará la columna para XML.
5. Indexará las tablas auxiliares para una búsqueda estructural.
6. Guardará el documento utilizando una función definida por el usuario, proporcionada por el XML Extender.

Almacenamiento de la DTD en el depósito de DTD

Puede utilizar una DTD para validar datos XML en una columna XML. El XML Extender crea una tabla en una base de datos habilitada para XML, llamada DTD_REF. La tabla se conoce con el nombre de referencia de DTD y está disponible para que pueda almacenar en ella las DTD. Cuando decida

validar documentos XML, deberá almacenar la DTD en este depósito. La DTD para esta guía de aprendizaje es `c:\dxx\samples\dtd\getstart.dtd`.

Para insertar la DTD:

Desde la Ventana de mandatos de DB2, entre el siguiente mandato INSERT de SQL, todo en la misma línea:

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

También puede ejecutar el archivo de mandatos para insertar la DTD:

```
getstart_insertDTD.cmd
```

Preparación del archivo DAD

El archivo DAD para la columna XML tiene una estructura simple. La modalidad de almacenamiento se especifica como columna XML y las tablas y columnas se definen para indexación.

En los pasos siguientes, se hace referencia a los elementos de la DAD como *códigos* y a los elementos del documento XML como *elementos*. En `c:\dxx\samples\dad\getstart_xcolumn.dad` encontrará un ejemplo de archivo DAD similar al que va a crear. Tiene algunas pequeñas diferencias respecto al archivo que se genera en los pasos siguientes. Si lo utiliza para la lección, tenga en cuenta que las vías de acceso de archivo pueden ser diferentes de las de su propio entorno, el valor <validation> está establecido en NO, en vez de YES.

Para preparar el archivo DAD:

1. Abra un editor de texto y denomine el archivo como `getstart_xcolumn.dad`

   Tenga en cuenta que todos los códigos que se utilizan en el archivo DAD son sensibles a las mayúsculas y minúsculas.
2. Cree la cabecera de DAD, con las declaraciones XML y Doctype.

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

   El archivo DAD es un documento XML y requiere declaraciones XML.
3. Inserte los códigos <DAD></DAD> de apertura y cierre. Los demás códigos están incluidos dentro de estos códigos.
4. Inserte los códigos <DTDID></DTDID> de apertura y cierre para especificar el identificador DTDID que asocia la DAD con la DTD del documento XML y especifica la ubicación de DTD en el cliente.

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

Verifique que esta serie coincida con el valor que se utiliza como primer valor de parámetro cuando se inserta la DTD en la tabla de referencia de DTD en la sección "Almacenamiento de la DTD en el depósito de DTD" en la página 22. Por ejemplo, la vía de acceso que utilice para el DTDID puede ser diferente de la serie anterior si trabaja en una unidad de máquina distinta.

5. Especifique los códigos <validation></validation> de apertura y cierre para indicar que el XML Extender debe validar la estructura del documento XML utilizando la DTD que se ha insertado en la tabla de depósito de DTD.

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

   El valor de <validation> debe estar en mayúsculas.

6. Inserte los códigos <Xcolumn></Xcolumn> de apertura y cierre para definir el método de almacenamiento como columna XML. El método define que los datos XML deben almacenarse en una columna XML.

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

7. Inserte los códigos <table></table> de apertura y cierre para cada tabla auxiliar que deba generarse.

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

8. Inserte los códigos <column></column> de apertura y cierre para cada columna que deba incluirse en las tablas auxiliares. Cada código <column> tiene cuatro atributos:
   - **name:** el nombre de la columna.
   - **type:** el tipo de datos de la columna.
   - **path:** la vía de ubicación del elemento correspondiente en el documento XML, utilizando la sintaxis de XPath. Vea "Vía de ubicación" en la página 55 para obtener la sintaxis de la vía de acceso a la ubicación.
   - **multi-occurrence:** indicación sobre si la vía de ubicación del elemento puede aparecer más de una vez en la estructura del documento XM.

```
<Xcolumn>
<table name="order_side_tab">
  <column name="order_key"
          type="integer"
          path="/Order/@key"
          multi_occurrence="NO"/>
  <column name="customer"
```

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

9. Compruebe que existe un </Xcolumn> de cierre después del último código </table>.
10. Compruebe que existe un </DAD> de cierre después del código </Xcolumn>.
11. Guarde el archivo como `getstart_xcolumn.dad`.

Puede comparar el archivo que acaba de crear con el archivo de ejemplo, `c:\dxx\samples\dad\getstart_xcolumn.dad`. Este archivo es una copia de trabajo del archivo DAD necesario para habilitar la columna XML y crear las tablas auxiliares. El archivo de ejemplo contiene las sentencias path que es posible que tengan que modificarse para que coincidan con su entorno y se ejecuten satisfactoriamente.

Creación de la tabla SALES_TAB

En esta sección creará la tabla SALES_TAB. Inicialmente tiene dos columnas con la información de venta para el pedido.

Para crear la tabla:

Desde la Ventana de mandatos de DB2, entre las sentencia CREATE TABLE siguiente:

```
DB2 CONNECT TO SALES_DB

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

Alternativamente, puede ejecutar el archivo de mandatos siguiente para crear la tabla:

```
getstart_createTabCol.cmd
```

Adición de la columna de tipo XML

Ahora, añada una nueva columna a la tabla SALES_TAB. Esta columna contendrá el documento XML inalterado que ha generado previamente y deberá tener un UDT XML. El XML Extender proporciona varios tipos de datos, que se describen en el "Capítulo 8. Tipos definidos por el usuario del XML Extender" en la página 173. En esta guía de aprendizaje, va a guardar el documento como XMLVARCHAR.

Para añadir la columna de tipo XML:

Desde la Ventana de mandatos de DB2, entre la siguiente sentencia de SQL:

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

Alternativamente, puede ejecutar el archivo de mandatos siguiente para modificar la tabla:

```
getstart_alterTabCol.cmd
```

Habilitación de la columna XML

Después de crear la columna de tipo XML, habilítela para el XML Extender. Cuando habilita la columna, el XML Extender lee el archivo DAD y crea las tablas auxiliares. Antes de habilitar la columna debe:

- Determinar si desea crear una vista por omisión de la columna XML, que contenga el documento XML, y las columnas de tabla auxiliar. Puede especificar la vista por omisión cuando consulte el documento XML. En esta lección especificará la vista con el parámetro `-v`.
- Determine si desea especificar una clave primaria como *ID RAÍZ*, el nombre de columna de la clave primaria en la tabla de aplicación y un identificador exclusivo que asocia todas las tablas auxiliares con la tabla de aplicación. Si no especifica ninguna clave primaria, el XML Extender añade la columna DXXROOT_ID a la tabla de aplicación y a las tablas auxiliares. La columna ROOT_ID une la aplicación y las tablas auxiliares, lo que permite al XML Extender actualizar automáticamente las tablas auxiliares si se actualiza el documento XML. En esta lección especificará el nombre de la clave primaria en el mandato (INVOICE_NUM) con el parámetro `-r`. A continuación, el XML Extender utilizará la columna especificada como ID_RAÍZ y añadirá la columna a las tablas auxiliares.
- Determine si desea especificar un espacio de tablas o si desea utilizar el espacio de tablas por omisión. En esta lección va a utilizar el espacio de tablas por omisión.

Para habilitar la columna para XML:

Desde la Ventana de mandatos de DB2, entre el mandato siguiente:

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

Alternativamente, puede ejecutar el archivo de mandatos siguiente para habilitar la columna para XML:

```
getstart_enableCol.cmd
```

El XML Extender crea las tablas auxiliares con la columna INVOICE_NUM y crea la vista por omisión.

Importante: No modifique las tablas auxiliares de ningún modo. Sólo debe actualizar el documento XML utilizando las UDF proporcionadas por el XML Extender. El XML Extender actualizará automáticamente las tablas auxiliares cuando actualice el documento XML en la columna XML.

Visualización de las columnas y las tablas auxiliares

Cuando ha habilitado la columna XML, ha creado una vista de la columna XML y de las tablas auxiliares. Puede utilizar esta vista cuando trabaje con la columna XML.

Para ver la columna XML y las columnas de tabla auxiliar:

Desde la Ventana de mandatos de DB2, entre la siguiente sentencia SELECT de SQL:

```
DB2 SELECT * FROM SALES_ORDER_VIEW
```

La vista muestra las columnas de las tablas auxiliares, tal como se especifica en el archivo `getstart_xcolumn.dad`.

Creación de índices en las tablas auxiliares

La creación de índices en tablas auxiliares le permite realizar búsquedas estructurales rápidas del documento XML. En este paso creará índices en columnas clave de las tablas auxiliares creadas cuando habilitó la columna XML, ORDER. El departamento de servicio ha especificado las columnas que probablemente van a consultar con más frecuencia sus empleados. La Tabla 3 describe estas columnas, que va a indexar:

Tabla 3. Columnas de tabla auxiliar que se van a indexar

Columna	Tabla auxiliar
ORDER_KEY	ORDER_SIDE_TAB
CUSTOMER	ORDER_SIDE_TAB
PRICE	PART_SIDE_TAB
DATE	SHIP_SIDE_TAB

Para indexar las tablas auxiliares:

Entre los mandatos SQL siguientes desde la Ventana de mandatos de DB2:

```
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)
```

Alternativamente, puede ejecutar el archivo de mandatos siguiente para crear los índices:

```
getstart_createIndex.cmd
```

Almacenamiento del documento XML

Después de haber habilitado una columna que puede contener el documento XML y de haber indexado las tablas auxiliares, puede almacenar el documento utilizando las funciones que proporciona el XML Extender. Para almacenar datos en una columna XML puede utilizar las funciones predefinidas por omisión o las UDF del XML Extender. Dado que va a almacenar un objeto del tipo base VARCHAR en una columna del UDT XML XMLVARCHAR, utilizará la función predefinida por omisión. Vea "Almacenamiento de datos" en la página 122 para obtener más información sobre las funciones predefinidas por omisión y las UDF proporcionadas por el XML Extender.

Para almacenar el documento XML:

Importante: Abra el documento de XML `c:\dxx\samples\xml\getstart.xml`. Asegúrese de que la vía de acceso de archivos de DOCTYPE coincide con el ID de DTD especificado en la DAD y al insertar la DTD en el depósito de DTD. Para verificar que coincidan, consulte la tabla db2xml.DTD_REF y compruebe el elemento DTDID del archivo DAD. Si utiliza una unidad y una estructura de directorios distintas de las que se asignan por omisión, tal vez tenga que modificar la vía de acceso en la declaración DOCTYPE.

Desde la Ventana de mandatos de DB2, entre el siguiente mandato INSERT de SQL:

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

Cuando almacena el documento XML, el XML Extender actualiza automáticamente las tablas auxiliares.

Alternativamente, puede ejecutar el archivo de mandatos siguiente para guardar el documento:

```
getstart_insertXML.cmd
```

Para comprobar que se hayan actualizado las tablas, ejecute las siguientes sentencias SELECT para las tablas desde la Ventana de mandatos de DB2:

```
DB2 SELECT * FROM SALES_TAB

DB2 SELECT * FROM PART_SIDE_TAB

DB2 SELECT * FROM ORDER_SIDE_TAB

DB2 SELECT * FROM SHIP_SIDE_TAB
```

Búsqueda en el documento XML

Puede buscar en el documento XML con una consulta directa de las tablas auxiliares. En este paso buscará todos los pedidos con un precio que esté por encima de 2500,00.

Para consultar las tablas auxiliares:

Entre la siguiente sentencia SELECT desde la Ventana de mandatos de DB2:

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

El conjunto de resultados debe mostrar los nombres de los vendedores que han vendido un artículo con un precio por encima de 2500,00.

Alternativamente, puede ejecutar el archivo de mandatos siguientes para buscar en el documento:

```
getstart_queryCol.cmd
```

Ha completado la guía de iniciación para almacenar documentos XML en tablas DB2. Muchos de los ejemplos del manual están basados en estas lecciones.

Lección: Composición de un documento XML

Escenario de la guía de aprendizaje

Se le asignado la tarea de obtener información de una base de datos de pedidos de compra existente, SALES_DB, y extraer información clave de ésta para almacenarla en documentos XML. El departamento de servicio utilizará estos documentos XML cuando trabaje con pedidos y reclamaciones de los clientes. El departamento de servicio ha solicitado que se incluyan datos específicos y ha proporcionado una estructura para los documentos XML.

Utilizando datos existentes va a componer un documento XML, `getstart.xml`, a partir de los datos de estas tablas.

También va a planificar y crear un archivo DAD que correlacione las columnas de las tablas relacionadas con una estructura de documento XML que proporciona un registro de pedidos de compra. Debido a que este documento está formado por múltiples tablas, va a crear una colección XML, asociando estas tablas con una estructura XML y una DTD. Utilice esta DTD para definir la estructura del documento XML. También puede utilizarla para validar el documento XML compuesto en las aplicaciones.

Los datos de la base de datos existente para el documento XML se describen en las tablas siguientes. Los nombres de columna que aparecen en *cursiva* son las columnas que el departamento de servicio ha solicitado que estén en la estructura de documento XML.

ORDER_TAB

Nombre de columna	Tipo de datos
ORDER_KEY	INTEGER
CUSTOMER	VARCHAR(16)
CUSTOMER_NAME	VARCHAR(16)
CUSTOMER_EMAIL	VARCHAR(16)

PART_TAB

Nombre de columna	Tipo de datos
PART_KEY	INTEGER
COLOR	CHAR(6)
QUANTITY	INTEGER
PRICE	DECIMAL(10,2)
TAX	REAL
ORDER_KEY	INTEGER

SHIP_TAB

Nombre de columna	Tipo de datos
DATE	DATE
MODE	CHAR(6)
COMMENT	VARCHAR(128)
PART_KEY	INTEGER

Planificación

Antes de empezar a trabajar con el XML Extender para componer los documentos, debe determinar la estructura del documento XML y cómo ésta corresponde a la estructura de los datos de la base de datos. Esta sección le proporcionará una visión general de la estructura del documento XML que ha solicitado el departamento de servicio, de la DTD que va a utilizar para definir la estructura del documento y de cómo este documento se correlaciona con las columnas que contienen los datos que se utilizan para rellenar los documentos.

Determinación de la estructura del documento

La estructura del documento XML obtiene información para un pedido específico de varias tablas y crea un documento XML para dicho pedido. Cada una de estas tablas contiene información relacionada sobre el pedido y se puede unir en sus columnas clave. El departamento de servicio desea un documento que se estructure según el número de pedido como nivel superior y, a continuación, la información de cliente, pieza y envío. Quieren que la estructura del documento sea intuitiva y flexible, cuyos elementos describan los datos, en lugar de la estructura del documento. (Por ejemplo, el nombre de cliente debe estar en un elemento llamado "cliente," en lugar de un párrafo.) Según su petición, la estructura jerárquica de la DTD y del documento XML debe ser similar a la que se describe en la Figura 5 en la página 32.

Una vez designada la estructura del documento, debe crear una DTD que describa la estructura del documento XML. Esta guía de aprendizaje le proporciona un documento XML y una DTD. Puede ver el archivo DTD en el "Apéndice B. Ejemplos" en la página 275. Puede ver que coincide con la estructura de la Figura 5 en la página 32.

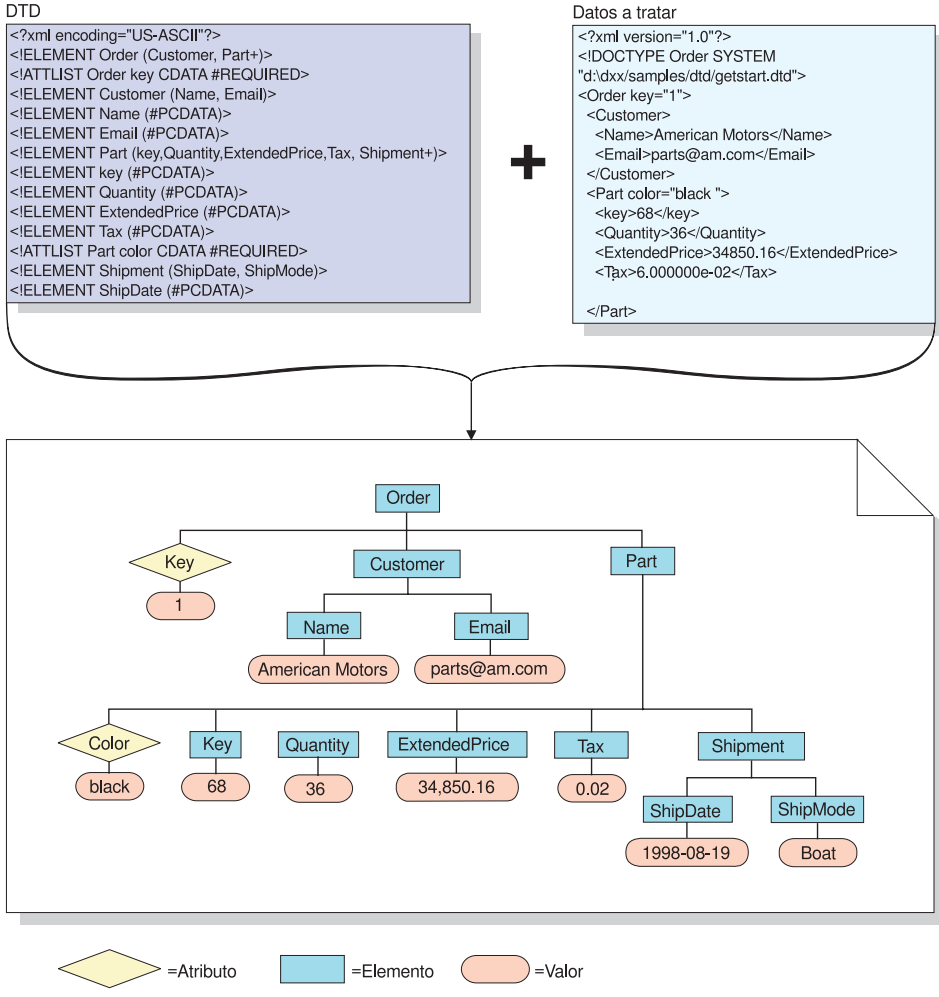


Figura 5. Estructura jerárquica de la DTD y el documento XML

Correlación de la relación de un documento XML y una base de datos

Después de designar la estructura y crear la DTD, debe indicar cómo se relaciona la estructura del documento con las tablas DB2 que utilizará para rellenar los elementos y atributos. Puede correlacionar la estructura jerárquica con columnas específicas de las tablas relacionales, como se muestra en la Figura 6 en la página 33.

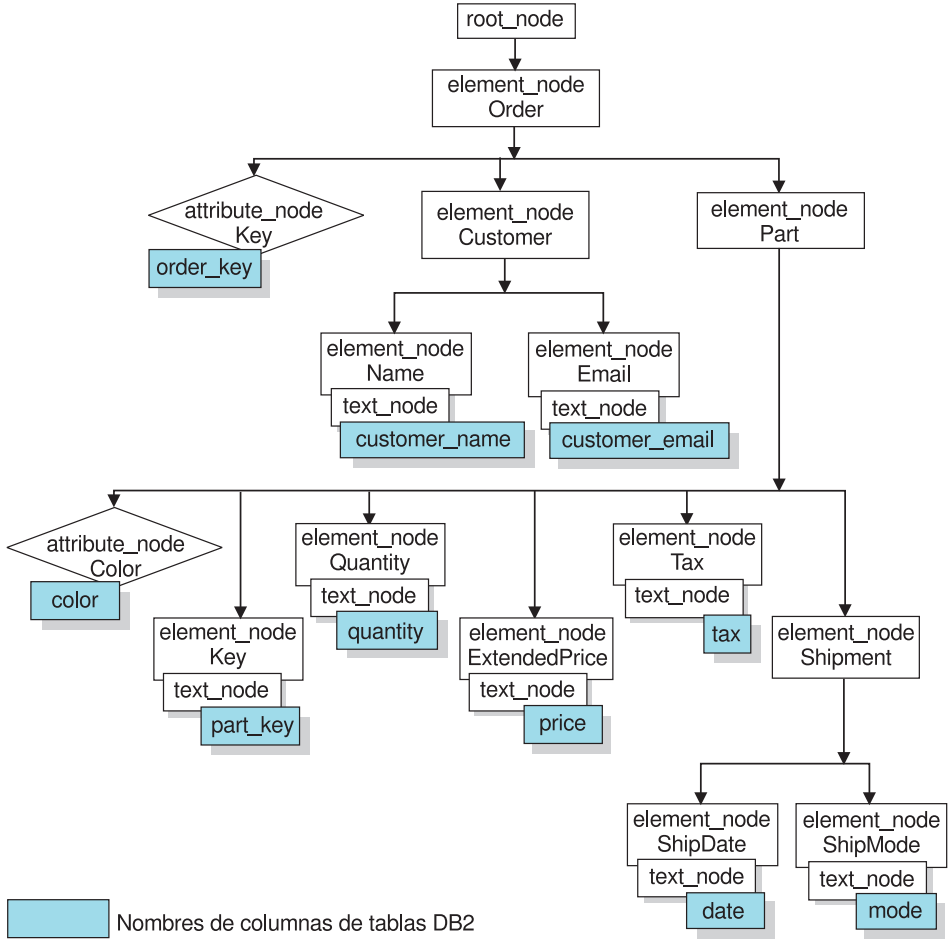


Figura 6. Documento XML correlacionado con columnas de tabla relacional

Utilice la descripción de esta relación para crear archivos DAD que definan la relación entre los datos relacionales y la estructura del documento XML.

Para crear el archivo DAD de la colección XML, debe comprender la correspondencia entre el documento XML y la estructura de la base de datos, tal como se describe en la Figura 6, para que pueda describir de qué tablas y columnas obtiene datos la estructura del documento XML para los elementos y atributos. Utilizará esta información para crear el archivo DAD para la colección XML.

Para esta guía de aprendizaje, se proporciona un conjunto de scripts para que los utilice para configurar su entorno. Estos scripts se encuentran en el directorio *DXX_INSTALL*\samples\cmd (donde *DXX_INSTALL* es la unidad y el

directorio en el que se ha instalado el XML Extender, por ejemplo `c:\dxx\samples\cmd`) y son los siguientes:

getstart_db.cmd
Crea la base de datos y rellena cuatro tablas.

getstart_prep.cmd
Enlaza la base de datos con los procedimientos almacenados del XML Extender y la CLI de DB2.

getstart_stp.cmd
Ejecuta el procedimiento almacenado para componer la colección XML.

getstart_exportXML.cmd
Exporta el documento XML desde la base de datos para utilizarlo en una aplicación.

getstart_clean.cmd
Limpia el entorno de la guía de aprendizaje.

Puesta a punto

Creación de la base de datos

En esta sección se utiliza un mandato para preparar la base de datos. Este mandato crea una base de datos de ejemplo, se conecta a ella, crea las tablas que van a contener los datos y, después, inserta los datos.

Importante: Si ha completado la lección sobre la columna XML y no ha limpiado el entorno, puede saltarse este paso. Compruebe si tiene una base de datos llamada SALES_DB.

Para crear la base de datos:

1. Cambie al directorio *DXX_INSTALL*`\samples\cmd`, donde *DXX_INSTALL* es la unidad y el directorio en los que ha instalado el XML Extender. En estas lecciones se supone que el directorio es `c:\dxx`. Cambie estos valores si su unidad y directorio son otros.
2. Abra la Ventana de mandatos de DB2 desde el menú Inicio de Windows NT o entre el mandato siguiente desde el indicador de mandatos de Windows NT:

   ```
   DB2CMD
   ```

3. Desde la Ventana de mandatos de DB2, ejecute el mandato siguiente:

   ```
   getstart_db.cmd
   ```

Habilitación de la base de datos

Para guardar información XML en la base de datos, debe habilitarla para el XML Extender. Cuando habilita una base de datos para XML, el XML Extender:

- Crea todos los tipos definidos por el usuario (UDT) y todas las funciones definidas por el usuario (UDF).
- Crea y rellena las tablas de control con los metadatos necesarios para el XML Extender.
- Crea el esquema db2xml y asigna los privilegios necesarios.

Importante: Si ha completado la lección de la columna XML y no ha limpiado el entorno, puede saltarse este paso.

Para habilitar la base de datos para XML:

Desde la Ventana de mandatos de DB2, ejecute el script siguiente para habilitar las base de datos SALES_DB:

```
getstart_prep.cmd
```

Este script enlaza la base de datos con los procedimientos almacenados del XML Extender y la CLI de DB2. También ejecuta la opción del mandato **dxxadm** que habilita la base de datos:

```
dxxadm enable_db SALES_DB
```

Creación de la colección XML: preparación del archivo DAD

Dado que los datos ya existen en varias tablas, creará una colección XML, que asocia las tablas con el documento XML. Para crear una colección XML, defina la colección mediante la preparación de un archivo DAD.

En "Planificación" en la página 31 ha determinado las columnas que están en la base de datos relacional en la que existen los datos y cómo se estructurarán las tablas para formar un documento XML. En esta sección, va a crear el esquema de correlación del archivo DAD que especifica la relación entre las tablas y la estructura del documento XML.

En los pasos siguientes, se hace referencia a los elementos de la DAD como *códigos* y a los elementos del documento XML como *elementos*. En `c:\dxx\samples\dad\getstart_xcollection.dad` encontrará un ejemplo de un archivo DAD similar al que va a crear. Tiene algunas pequeñas diferencias respecto al archivo que se genera en los pasos siguientes. Si lo utiliza para esta lección, tenga en cuenta que las vías de acceso de archivo pueden ser diferentes de las de su propio entorno.

Para crear el archivo DAD para componer un documento XML:

1. Desde el directorio `c:\dxx\samples\cmd`, abra un editor de texto y cree un archivo llamado `getstart_xcollection.dad`.
2. Cree la cabecera del archivo DAD, utilizando el texto siguiente:

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

El XML Extender supone que ha instalado el producto en `c:\dxx`. Si esto no es así, cambie este valor por la unidad y el directorio que haya especificado durante la instalación de este producto aquí y en los pasos siguientes.

3. Inserte los códigos <DAD></DAD>. Los demás códigos están incluidos dentro de estos códigos.
4. Especifique los códigos <validation> </validation> para indicar si el XML Extender valida la estructura del documento XML utilizando la DTD que ha insertado en la tabla de depósito de DTD.

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

5. Utilice los códigos <Xcollection></Xcollection> para definir el método de acceso y almacenamiento como colección XML. Los métodos de acceso y de almacenamiento definen que los datos XML se guardan en una colección de tablas DB2.

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

6. Especifique una sentencia SQL para especificar las tablas y columnas que se utilizan para la colección XML. Este método recibe el nombre correlación de SQL y es una de las dos maneras de correlacionar los datos relacionales con la estructura del documento XML. (Vea "Tipos de esquemas de correlación" en la página 63 para obtener más información sobre los esquemas de correlación.) Entre la sentencia siguiente:

   ```
   <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>
   ```

   Esta sentencia SQL utiliza las siguientes directrices cuando utiliza la correlación SQL. Consulte la Figura 6 en la página 33 para ver la estructura del documento.

   - Las columnas se especifican en orden de superior a inferior, según la jerarquía de la estructura del documento XML. Por ejemplo, las columnas para los elementos de pedido y cliente van primero, el elemento de pieza en segundo lugar y el envío en tercer lugar.
   - Las columnas para una entidad se agrupan y cada grupo tiene una columna de ID de objeto: ORDER_KEY, PART_KEY y SHIP_ID.
   - La columna de ID de objeto es la primera columna en cada grupo. Por ejemplo, O.ORDER_KEY precede a las columnas relacionadas con el atributo y p.PART_KEY precede a las columnas para el elemento Pieza.

- La tabla SHIP_TAB no tiene una única columna condicional clave y, por lo tanto, se utiliza la función incorporada de DB2 generate_unique para generar la columna SHIP_ID.
- A continuación, las columnas de ID de objeto se listan en orden de superior a inferior en una sentencia ORDER BY. Las columnas de ORDER BY no deben estar calificadas por ningún esquema ni nombre de tabla y deben coincidir con los nombres de tabla de la cláusula SELECT.

Vea "Requisitos del esquema de correlación" en la página 65 para conocer los requisitos cuando se escribe una sentencia SQL.

7. Añada la siguiente información de prólogo que debe utilizarse en el documento XML compuesto.

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

Este texto exacto es necesario para todos los archivos DAD.

8. Añada los códigos <doctype></doctype> que deben utilizarse en el documento XML que está componiendo. El código <doctype> contiene la vía de acceso a la DTD almacenada en el cliente.

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

9. Defina el elemento raíz del documento XML utilizando los códigos <root_node></root_node>. Dentro del nodo raíz, especifique los elementos y atributos que forman el documento XML.
10. Correlacione la estructura del documento XML con la estructura de la tabla relacional DB2 utilizando los tres tipos de nodos siguientes:

element_node > (nodo_de_elemento)
Especifica el elemento del documento XML. Los element_nodes pueden tener element_nodes hijo.

attribute_node (nodo_de_atributo)
Especifica el atributo de un elemento del documento XML.

text_node (nodo_de_texto)
Especifica el contenido del texto del elemento y los datos de las columnas de una tabla relacional para element_nodes de nivel inferior.

Vea "El archivo DAD" en la página 60 para obtener más información sobre estos nodos. La Figura 6 en la página 33 muestra la estructura jerárquica del documento XML y las columnas de tabla DB2, e indica qué tipos de nodos se utilizan. Los recuadros sombreados indican los nombres de columna de tabla DB2 de los que se obtendrán datos para componer el documento XML.

En los pasos siguientes añadirá cada tipo de nodo, un tipo a la vez.

a. Defina un código <element_node> para cada elemento del documento XML.

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

Tenga en cuenta que el elemento hijo <Shipment> tiene un atributo de multi_occurrence=″YES″. Este atributo se utiliza para los elementos sin un atributo, que se repiten en el documento. El elemento <Part> no utiliza el atributo multi-occurrence porque tiene un atributo de color que hace que sea exclusivo.

b. Defina un código <attribute_node> para cada atributo de documento XML. Estos atributos están anidados en su element_node. Los attribute_nodes añadidos se resaltan en negrita:

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

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

...

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

c. Para cada element_node de nivel superior, defina los códigos <text_node>, para indicar que el elemento XML contiene datos de tipo carácter que deben extraerse de DB2 cuando se compone el documento.

```
<root_node>
<element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node>
      </text_node>
    </element_node>
    <element_node name="Email">
      <text_node>
      </text_node>
    </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
    <element_node name="key">
       <text_node>
       </text_node>
    </element_node>
    <element_node name="Quantity">
      <text_node>
      </text_node>
    </element_node>
    <element_node name="ExtendedPrice">
      <text_node>
      </text_node>
    </element_node>
    <element_node name="Tax">
      <text_node>
      </text_node>
    </element_node>
    <element_node name="Shipment" multi-occurrence="YES">
      <element_node name="ShipDate">
        <text_node>
        </text_node>
      </element_node>
      <element_node name="ShipMode">
        <text_node>
        </text_node>
      </element_node>
```

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

d. Para cada element_node de nivel inferior, defina un código <column>. Estos códigos especifican de qué columna deben extraerse los datos cuando se compone el documento XML y generalmente están incluidos dentro del código <attribute_node> o <text_node>. Recuerde que las columnas que defina aquí deben estar en la cláusula <SQL_stmt> SELECT.

```
<root_node>
<element_node name="Order">
  <attribute_node name="key">
    <column name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node>
        <column name="customer_name"/>
      </text_node>
    </element_node>
    <element_node name="Email">
      <text_node>
        <column name="customer_email"/>
     </text_node>
    </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
       <column name="color"/>
    </attribute_node>
    <element_node name="key">
      <text_node>
        <column name="part_key"/>
      </text_node>
    <element_node name="Quantity">
      <text_node>
        <column name="quantity"/>
      </text_node>
    </element_node>
    <element_node name="ExtendedPrice">
      <text_node>
        <column name="price"/>
      </text_node>
    </element_node>
    <element_node name="Tax">
      <text_node>
        <column name="tax"/>
      </text_node>
    </element_node>
    <element_node name="Shipment" multi-occurrence="YES">
      <element_node name="ShipDate">
        <text_node>
```

```
            <column name="date"/>
          </text_node>
        </element_node>
        <element_node name="ShipMode">
          <text_node>
            <column name="mode"/>
          </text_node>
        </element_node>
      </element_node> <!-- end Shipment -->
    </element_node> <!-- end Part -->
  </element_node> <!-- end Order -->
  </root_node>
```

11. Compruebe que hay un código final </root_node> a continuación del último código </element_node>.
12. Compruebe que hay un código final </Xcollection> a continuación del código </root_node>.
13. Compruebe que hay un código final </DAD> a continuación del código </Xcollection>.
14. Guarde el archivo como `getstart_xcollection.dad`

Puede comparar el archivo que acaba de crear con el archivo de ejemplo `c:\dxx\samples\dad\getstart_xcollection.dad`. Este archivo es una copia de trabajo del archivo DAD necesario para componer el documento XML. El archivo de ejemplo contiene las sentencias path que es posible que tengan que modificarse para que coincidan con su entorno y se ejecuten satisfactoriamente.

En la aplicación, si va a utilizar con frecuencia una colección XML para componer documentos, puede definir un nombre de colección habilitando la colección. Cuando se habilita la colección, ésta se registra en la tabla XML_USAGE y puede mejorar el rendimiento especificando el nombre de colección (en lugar del nombre de archivo DAD) al ejecutar procedimientos almacenados. En estas lecciones, no va a habilitar la colección. Para obtener más información sobre cómo habilitar colecciones, vea "Habilitación de colecciones XML" en la página 114.

Proceso de composición del documento XML

En este paso, utilizará el procedimiento almacenado dxxGenXML() para componer el documento XML especificado por el archivo DAD. Este procedimiento almacenado devuelve el documento con un UDT XMLVARCHAR.

Para componer el documento XML:

1. Desde la Ventana de mandatos de DB2, entre el mandato siguiente para ejecutar el procedimiento almacenado:

   ```
   getstart_stp.cmd
   ```

El documento XML se ha compuesto y se guarda en la tabla RESULT_TAB.

Puede ver ejemplos de los procedimientos almacenados que pueden utilizarse en este paso en los archivos siguientes:

- `c:\dxx\samples\c\tests2x.sqc` muestra cómo llamar al procedimiento almacenado utilizando SQL intercalado y genera el archivo ejecutable `texts2x`, utilizado por `getstart_stp.cmd`.
- `c:\dxx\samples\cli\sql2xml.c` muestra cómo llamar al procedimiento almacenado utilizando la CLI.

2. Exporte el documento XML desde la tabla a un archivo utilizando la función de recuperación del XML Extender, Content():

```
DB2 CONNECT TO SALES_DB

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

Alternativamente, puede ejecutar el mandato siguiente para exportar el archivo:

```
getstart_exportXML.cmd
```

Esta lección le enseña cómo obtener uno o más documentos XML compuestos utilizando la función de conjunto de resultados del procedimiento almacenado de DB2 para que le permita extraer cada fila para obtener cada documento. Cuando obtiene cada fila de un documento, puede exportarla a un archivo, que es el modo más simple de demostrar el funcionamiento de esta característica. Para conocer maneras más eficaces de obtener datos, vea los ejemplos de CLI en `c:\dxx\samples\cli`.

Limpieza del entorno de la guía de aprendizaje

Si desea limpiar el entorno de la guía de aprendizaje, puede ejecutar el archivo `getstart_clean.cmd`. Este archivo:

- Inhabilita la columna XML, ORDER
- Elimina las tablas creadas en la guía de aprendizaje
- Suprime la DTD de la tabla de referencia de DTD

Este archivo de mandatos no inhabilita ni elimina la base de datos SALES_DB; esta base de datos sigue disponible para que la utilice el XML Extender. Puede recibir mensajes de error si no ha completado las dos lecciones de este capítulo. Puede ignorar estos errores.

Para limpiar el entorno de la guía de aprendizaje:

1. Desde la Ventana de mandatos de DB2, ejecute el mandato siguiente:

```
getstart_clean.cmd
```

2. Si desea inhabilitar la base de datos, puede ejecutar el mandato siguiente del XML Extender desde la Ventana de mandatos de DB2:

   ```
   dxxadm disable_db SALES_DB
   ```

   Este mandato elimina las tablas de control de administración DTD_REF y XML_USAGE, y además elimina los tipos definidos por el usuario y las funciones proporcionadas por el XML Extender.
3. Si desea inhabilitar la base de datos, puede ejecutar el mandato siguiente desde la Ventana de mandatos de DB2:

   ```
   db2 drop database SALES_DB
   ```

   Este mandato elimina SALES_DB.

Parte 2. Administración

Esta parte describe cómo realizar tareas administrativas para el XML Extender.

Capítulo 3. Preparación para utilizar el XML Extender: administración

Este capítulo describe los requisitos para poner a punto y preparar la utilización del XML Extender.

Requisitos para la puesta a punto

Las secciones siguientes describen los requisitos para la puesta a punto de XML Extender.

Requisitos de software

El XML Extender puede utilizarse en AIX, Windows NT y Sun Solaris.

Software necesario: El XML Extender requiere DB2 Universal Database Versión 7.1 o superior.

Software opcional:

- Para realizar la búsqueda de texto estructural, es necesario tener DB2 Universal Database Text Extender Versión 7.1 o superior
- Para la herramienta de administración de XML Extender:
  - DB2 UDB JDBC (proporcionado con DB2 UDB Versión 7.1 o superior)
  - JDK 1.1.7 o JRE 1.1.7 (proporcionado con el Centro de Control de DB2 UDB)
  - JFC 1.1 con Swing 1.1 (proporcionado con el Centro de Control de DB2 UDB)

Requisitos de instalación

Vea el archivo README correspondiente a su sistema operativo para obtener información sobre las tareas siguientes:

- Enlazar el XML Extender con la base de datos DB2 UDB.

  Por razones de seguridad, debe enlazar el XML Extender con cada base de datos. Para obtener detalles sobre cómo realizar el enlace, vea "Antes de comenzar" en la página 215 o, para ver un ejemplo, vea

  DXX_INSTALL\samples\cmd\getstart_prep.cmd
- Visualizar las instrucciones de puesta a punto en UNIX.
- Crear una base de datos para el acceso mediante XML.

Requisitos de autorización

Es necesario tener autorización DB2ADM para realizar tareas de administración.

Herramientas de administración

El XML Extender proporciona tres métodos para realizar la administración: el asistente de administración del XML Extender, el mandato de administración del XML Extender y los *procedimientos almacenados* del XML Extender.

- El asistente de administración sirve de guía para realizar las tareas de administración y es el método recomendado para efectuar esas tareas. El uso de esta herramienta está descrito en las tareas de administración del "Capítulo 4. Administración de datos XML" en la página 71.
- El mandato de administración, **dxxadm**, proporciona opciones para las diversas tareas de administración. El uso de este mandato está descrito en las tareas de administración del "Capítulo 4. Administración de datos XML" en la página 71 y del "Capítulo 7. El mandato de administración del XML Extender: **dxxadm**" en la página 159.
- Los procedimientos almacenados de administración también proporcionan opciones para diversas tareas de administración. Estos procedimientos almacenados están descritos en "Procedimientos almacenados de administración" en la página 215.

Planificación de la administración

Cuando planifique utilizar una aplicación que hace uso de documentos XML, primero debe determinar lo siguiente:

- Si compondrá documentos XML a partir de datos de la base de datos
- Si almacenará documentos XML preexistentes y si desea almacenarlos en una columna como documentos XML inalterados o bien como documentos descompuestos en datos DB2 normales.

Una vez determinado lo anterior, puede planificar el resto de las tareas de administración:

- Elija si desea validar o no los documentos XML
- Elija si desea indexar datos de columnas XML para permitir una búsqueda y recuperación rápida de los datos
- Cómo correlacionar la estructura del documento XML con tablas relacionales DB2

La forma en que utiliza el XML Extender depende de las necesidades de su programa de aplicación. Tal como se indica en el "Capítulo 1. Nociones preliminares sobre el XML Extender" en la página 3, puede componer documentos XML a partir de datos DB2 existentes y almacenar documentos XML en DB2, en forma de documentos inalterados o como datos DB2. Cada uno de estos métodos de almacenamiento y acceso tiene unos requisitos de planificación distintos. Las secciones siguientes tratan sobre cada una de estas consideraciones de planificación.

Elección de un método de acceso y almacenamiento

El XML Extender proporciona dos métodos de acceso y almacenamiento para utilizar DB2 como depósito XML: la columna XML y la colección XML. Primero debe determinar cuál de estos métodos se ajusta mejor a las necesidades de su programa de aplicación para acceder y manejar datos XML.

columna XML
Almacena y recupera documentos XML completos en forma de datos de columnas DB2. Los datos XML se representan mediante una columna XML.

colección XML
Descompone documentos XML en una colección de tablas relacionales o compone documentos XML a partir de una colección de tablas relacionales.

La naturaleza de su programa de aplicación determina el tipo de acceso y método de almacenamiento que se debe utilizar y cómo estructurar los datos XML. Los casos siguientes describen las situaciones en las que es más apropiado cada uno de los métodos de acceso y almacenamiento.

Cuándo utilizar columnas XML

Utilice columnas XML en las situaciones siguientes:

- Los documentos XML ya existen o proceden de alguna fuente externa y prefiere almacenarlos en el formato XML nativo. Desea almacenar los documentos en DB2 para asegurar su integridad y con fines de archivado y auditoría.
- Generalmente los documentos XML se leen, pero no se actualizan.
- Desea utilizar tipos de datos de nombre de archivo para almacenar los documentos XML externos a DB2 en el sistema de archivos local o remoto y utilizar DB2 para operaciones de gestión y búsqueda.
- Necesita realizar búsquedas por rangos para valores de elementos o atributos XML, y conoce qué elementos o atributos serán a menudo los argumentos de búsqueda.
- Los documentos tienen elementos con grandes bloques de texto y desea utilizar el Text Extender de DB2 para realizar búsquedas estructurales de texto y al mismo tiempo mantener intactos los documentos completos.

Cuándo utilizar colecciones XML

Utilice colecciones XML en las situaciones siguientes:

- Sus tablas relacionales contienen datos y desea componer documentos XML basados en una determinada DTD.
- Tiene documentos XML que es necesario almacenar con colecciones de datos que se correlacionen bien con tablas relacionales.

- Desea crear vistas diferentes de sus datos relacionales utilizando diferentes esquemas de correlación.
- Tiene documentos XML que proceden de otras fuentes de datos. Desea preservar los datos, pero no los códigos, y desea guardar datos puros en la base de datos. Desea poder decidir entre almacenar los datos en tablas existentes o en tablas nuevas.
- Una pequeña parte de sus documentos XML necesita actualizarse con frecuencia, y el rendimiento de la actualización es de importancia crítica.
- Necesita almacenar los datos de documentos XML completos, pero a menudo sólo necesita recuperar una parte de ellos.
- Sus documentos XML tienen más de 2 gigabytes y debe descomponerlos.

Utilice el archivo de definición de acceso a documento (archivo DAD) para asociar datos XML con tablas DB2 mediante esos dos métodos de acceso y almacenamiento. La Figura 7 muestra cómo la DAD especifica los métodos de acceso y almacenamiento.

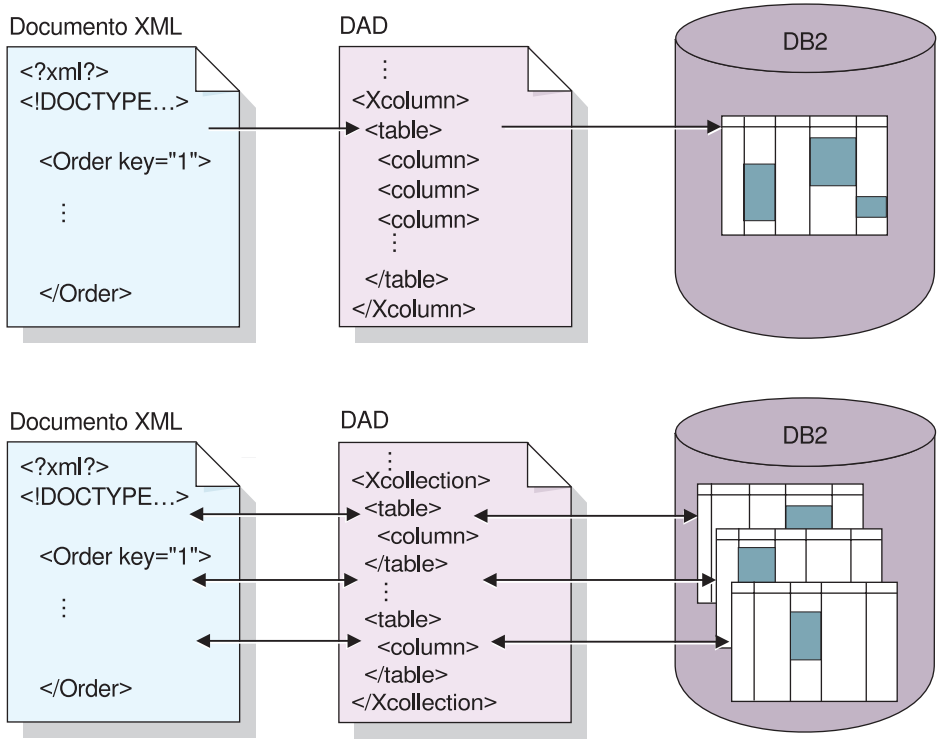


Figura 7. El archivo DAD correlaciona la estructura del documento XML con DB2 y especifica el método de acceso y almacenamiento.

El archivo DAD es una parte importante de la administración del XML Extender. Este archivo define la ubicación de archivos clave, tales como la DTD, y especifica cómo la estructura del documento XML se correlaciona con los datos DB2. Y lo que es más importante, define los métodos de acceso y almacenamiento que el usuario utiliza en el programa de aplicación.

Planificación para utilizar columnas XML

Las secciones siguientes describen las tareas de planificación para columnas XML.

Validación

Una vez elegido un método de acceso y almacenamiento, puede determinar si desea *validar* los datos. Para validar los datos XML utilice una DTD; esto asegura la validez del documento XML y permite realizar búsquedas estructuradas en los datos de XML. La DTD se almacena en el depósito de DTD o se puede almacenar en el sistema de archivos al que el servidor de DB2 tiene acceso.

Puede validar documentos de una misma columna XML utilizando varias DTD diferentes. Es decir, puede tener documentos de estructura similar, con elementos y atributos similares, que invocan definiciones DTD que son diferentes. Para hacer referencia a varias DTD, siga estas directrices:

- El ID de sistema del documento XML, contenido en la definición DOCTYPE, debe utilizar una vía de acceso completa para especificar el archivo DTD.
- Debe especificar YES para la validación en el archivo DAD.
- Como mínimo una de las DTD debe estar almacenada en la tabla DTD_REF. Todas las DTD pueden estar almacenadas en esta tabla.
- Las DTD deben tener una estructura común, con diferencias sólo en los subelementos.
- El archivo DAD debe especificar elementos o atributos que son comunes a todas las DTD referenciadas por documentos de esa columna.

Importante: Si decide validar los datos XML, debe hacer la validación antes de insertar los datos en DB2. El XML Extender no permite validar datos que ya estén insertados en DB2.

Consideraciones:

- Se recomienda validar los datos XML con una DTD, a menos que almacene los documentos XML con fines de archivado. Para realizar la validación, es necesario que tenga una DTD en el depósito del XML Extender. Vea "Almacenamiento de una DTD en el depósito de DTD" en la página 76 para conocer cómo insertar una DTD en el depósito.
- No es necesaria una DTD para almacenar ni archivar documentos XML.

- El validar los datos XML puede tener un pequeño efecto sobre el rendimiento del sistema.
- Puede utilizar varias DTD, pero sólo puede indexar elementos y atributos comunes.
- Si no opta por validar un documento, la DTD especificada por el documento XML no se procesa. Es importante que se procesen las DTD para resolver los valores por omisión de entidades y atributos incluso cuando se procesan fragmentos de documentos que no se pueden validar.

Tipos XML definidos por el usuario

Los documentos XML se almacenan en una columna XML en forma de UDT (tipo definido por el usuario) . La Tabla 4 lista los UDT disponibles.

Tabla 4. Los UDT del XML Extender

Columna de tipo definido por el usuario	Tipo de datos fuente	Descripción
XMLVARCHAR	VARCHAR(*long_varchar*)	Almacena un documento XML completo, en forma de VARCHAR, dentro de DB2.
XMLCLOB	CLOB(*long_clob*)	Almacena un documento XML completo, en forma de *CLOB*, dentro de DB2.
XMLFILE	VARCHAR(1024)	Almacena el nombre de archivo de un documento XML en DB2 y guarda el documento XML en un archivo local del servidor DB2.

Tablas auxiliares

Cuando se planifican tablas auxiliares, debe tenerse en cuenta cómo organizar las tablas, cuántas tablas se van a crear y si se va a crear una vista por omisión para las tablas auxiliares. Estas decisiones en parte se basan en varios puntos: si los elementos y atributos pueden aparecer varias veces y los requisitos de rendimiento de las consultas.

Aparición múltiple: Cuando un documento tiene vías de ubicación de aparición múltiple, el XML Extender añade una columna DXX_SEQNO del tipo INTEGER a cada tabla auxiliar para mantener un seguimiento del orden de los elementos que aparecen más de una vez. Con DXX_SEQNO, se puede recuperar una lista de los elementos utilizando el mismo orden que el documento XML original especificando ORDER BY DXX_SEQNO en una consulta SQL.

Vistas por omisión y rendimiento de la consulta: Cuando habilite una columna XML, puede especificar una vista de sólo lectura por omisión que una la tabla de aplicación con las tablas auxiliares utilizando un ID exclusivo llamado ID RAÍZ. Con la vista por omisión, puede buscar en documentos XML consultando las tablas auxiliares. Por ejemplo, si tiene la tabla de aplicación SALES_TAB y las tablas auxiliares ORDER_TAB, PART_TAB y SHIP_TAB:

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

La sentencia SQL devuelve los nombres de los vendedores de SALES_TAB que tienen pedidos guardados en la columna ORDER y en los que PRICE es mayor que 2500.00.

La ventaja de consultar la vista por omisión es que proporciona una única vista virtual de la tabla de aplicación y las tablas auxiliares. Sin embargo, cuantas más tablas auxiliares se creen, más costosa es la consulta. Por lo tanto, la creación de la vista por omisión sólo se recomienda cuando el número total de columnas de la tablas auxiliares es pequeño. Las aplicaciones pueden crear sus propias vistas, uniendo las columnas de tablas auxiliares importantes.

Índices para datos de columna XML

Una decisión importante de la planificación es si desea indexar el documento de columnas XML. Esta decisión debe tomarla basándose en la frecuencia con que necesita acceder a los datos y el grado de importancia que tiene el rendimiento en las búsquedas estructurales.

Cuando utiliza columnas XML, que contienen documentos XML completos, puede crear tablas auxiliares para contener columnas de valores de elementos o atributos XML, y luego crear índices sobre estas columnas. Debe determinar para qué elementos y atributos necesita crear el índice.

La indexación de columnas XML permite utilizar el soporte nativo de DB2 para indexar datos de consulta frecuente de tipo general, tales como INTEGER, DECIMAL o DATE. El XML Extender obtiene los valores de elementos o atributos XML a partir de documentos XML y los almacena en las *tablas auxiliares*, permitiendo al usuario crear índices sobre estas tablas auxiliares.

Puede especificar cada columna de una tabla auxiliar utilizando una vía de ubicación, la cual identifica un elemento o atributo XML y un tipo de datos de SQL. La Figura 8 en la página 54 muestra una columna XML con tablas auxiliares.

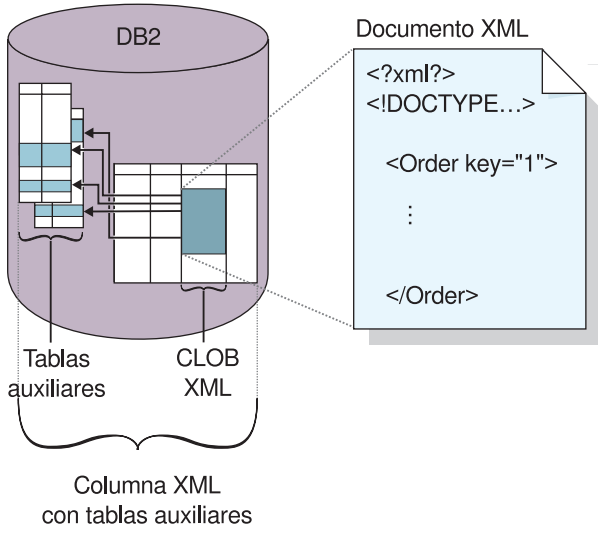


Figura 8. Columna XML con tablas auxiliares

El XML Extender llena automáticamente con datos la tabla auxiliar cuando el usuario almacena documentos XML en la columna XML.

Para poder realizar búsquedas rápidas, cree índices sobre estas columnas utilizando la técnica de la *indexación de árbol binario* de DB2. Los métodos utilizados para crear un índice varían según el sistema operativo, y el XML Extender da soporte a estos métodos.

Consideraciones:

- Para los documentos XML con elementos o atributos que aparecen varias veces (*apariciones múltiples*), debe crear una tabla auxiliar separada para cada elemento o atributo XML repetitivo, debido a la estructura compleja de los documentos XML.

  Por ejemplo, puede crear un índice para `/Order/Part/ExtendedPrice` y especificar el tipo de datos REAL para `/Order/Part/ExtendedPrice`. En este caso, el XML Extender almacena el valor de `/Order/Part/ExtendedPrice` en la columna PRICE de una tabla auxiliar.
- Puede crear varios índices sobre una columna XML. De acuerdo con el ejemplo anterior, puede crear dos columnas en dos tablas auxiliares: una para `ExtendedPrice` y otra para `ShipDate`.
- Puede asociar las tablas auxiliares con la tabla de aplicación utilizando el ID RAÍZ, el nombre de columna de la clave primaria de la tabla de aplicación y un identificador exclusivo que asocia todas las tablas auxiliares con la tabla de aplicación. También puede decidir si desea que la clave primaria

de la tabla de aplicación sea el ID RAÍZ, aunque no puede ser la clave compuesta. Este es el método recomendado.

Si la clave primaria simple no existe en la tabla de la aplicación, o por alguna razón no desea utilizarla, el XML Extender modifica la tabla de aplicación para añadir la columna DXXROOT_ID, la cual contiene un ID exclusivo que se crea en el momento de la inserción. Todas las tablas auxiliares tienen una columna DXXROOT_ID que contiene el ID exclusivo. Si se utiliza la clave primaria como ID RAÍZ, todas las tablas auxiliares tienen una columna con el mismo nombre y tipo que la columna de clave primaria de la tabla de aplicación, y se guardan los valores de las claves primarias.

- Si habilita una columna XML para el Text Extender de DB2, puede también utilizar la función de búsqueda estructural de texto del Text Extender. El Text Extender permite realizar una *búsqueda por secciones*, que amplía las prestaciones de una búsqueda de texto completa de tipo convencional, pues permite comparar palabras de búsqueda dentro de un contexto determinado del documento, que se especifica mediante vías de acceso de ubicación. El *índice de texto estructural* se puede utilizar con la función de indexación del XML Extender sobre tipos de datos generales de SQL.

Vía de ubicación

Una *vía de ubicación* es una secuencia de *códigos XML* que identifican un elemento o atributo de XML. El XML Extender utiliza la vía de ubicación en las siguientes situaciones:

- Cuando se utilizan las UDF de extracción, para identificar los elementos y atributos que se deben extraer
- Para especificar el archivo de correlación entre un elemento o atributo XML y una columna DB2 cuando se define el esquema de indexación en la DAD para columnas XML
- Para realizar búsquedas estructurales de texto con el Text Extender

La Figura 9 en la página 56 muestra un ejemplo de una vía de ubicación y su relación con la estructura del documento XML.

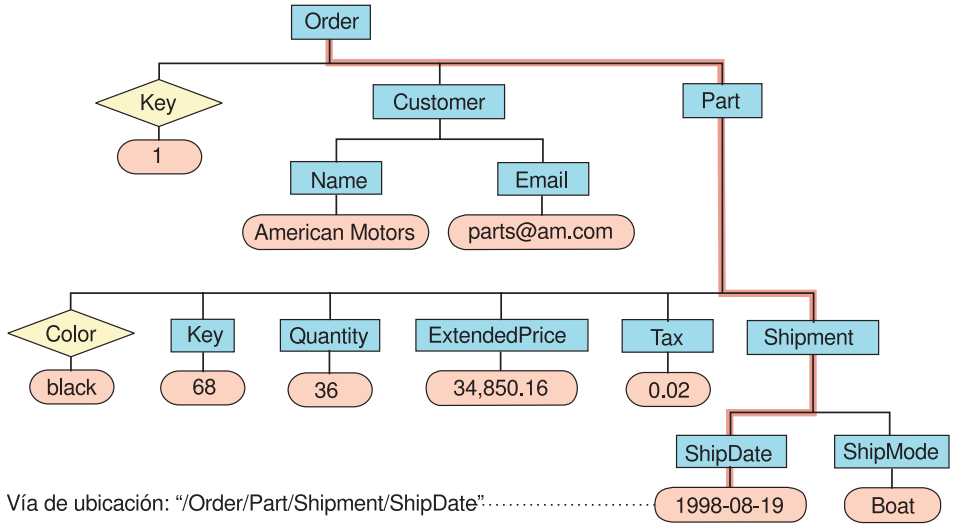


Figura 9. Almacenamiento de documentos como documentos XML estructurados en una columna de una tabla DB2

Sintaxis de la vía de ubicación: La lista siguiente describe la sintaxis de la vía de ubicación soportada por el XML Extender. Una vía con una barra inclinada (/) individual indica que el contexto es el documento completo.

1. / Representa el *elemento raíz* de XML.

2. */tag1*
Representa el elemento *tag1* contenido en la raíz

3. */tag1/tag2/..../tagn*
Representa un elemento llamado *tagn* que deriva de la cadena que comienza en la raíz y continua por *tag1*, *tag2* hasta *tagn*-1.

4. *//tagn*
Representa cualquier elemento llamado *tagn*, donde la barra inclinada doble (//) representa un número cualquiera de elementos.

5. */tag1//tagn*
Representa cualquier elemento llamado *tagn*, que deriva del elemento llamado *tag1* contenido en la raíz, donde la barra inclinada doble (//) representa un número cualquiera de elementos.

6. */tag1/tag2/@attr1*
Representa el atributo *attr1* del elemento *tag2*, que a su vez deriva del elemento *tag1* contenido en la raíz.

7. */tag1/tag2[@attr1="5"]*
Representa un elemento llamado *tag2* cuyo atributo *attr1* tiene el valor 5. *tag2* es hijo del elemento *tag1* contenido en la raíz.

8. */tag1/tag2[@attr1="5"]/.../tagn*

Representa un elemento llamado *tagn* que deriva de la cadena que comienza en la raíz y continua por *tag1*, *tag2* hasta *tagn*-1, donde el atributo *attr1* de *tag2* tiene el valor 5.

Caracteres comodín: Se puede sustituir un asterisco por un elemento de una vía de ubicación para que coincida con cualquier serie de caracteres.

Vía de ubicación simple: *Vía de ubicación simple* es la sintaxis de la vía de ubicación que se utiliza para especificar los elementos y atributos de las tablas auxiliares, definidas en el archivo DAD de la columna XML. La vía de ubicación simple está representada como secuencia de nombres de tipos de elementos que están conectados por una barra inclinada (/). Los valores de atributos están encerrados entre corchetes, a continuación del correspondiente tipo de elemento. La Tabla 5 resume la sintaxis de la vía de ubicación simple.

Tabla 5. Sintaxis de la vía de ubicación simple

Sujeto	Vía de ubicación	Descripción
Elemento XML	*/tag1/tag2/..../tagn-1/tagn*	Representa un contenido de elemento identificado por el elemento llamado *tagn* y sus padres
Atributo XML	*/tag_1/tag_2/..../tag_n-1/tag_n/@attr1*	Representa el atributo *attr1* del elemento identificado por *tagn* y sus padres

Restricciones referentes al XML Extender: El XML Extender tiene restricciones respecto al uso de la vía de ubicación al definir un elemento o atributo en la DAD. Debido a que el XML Extender utiliza una correlación de uno con uno entre un elemento o un atributo y una columna DB2, impone restricciones sobre las normas de sintaxis permitidas en el archivo DAD y en las funciones. La Tabla 6 describe las restricciones existentes para la vía de ubicación. Los números que figuran en la columna Vía de ubicación soportada remiten a las representaciones de sintaxis de "Sintaxis de la vía de ubicación" en la página 56.

Tabla 6. Restricciones del XML Extender en la utilización de la vía de ubicación

Uso de la vía de ubicación	Vía de ubicación soportada
Elemento de la correlación de DAD de columna XML para tablas auxiliares	3, 6 (vía de ubicación simple descrita en la Tabla 5)
Funciones UDF de extracción	1-8[1]
UDF de actualización	1-8[1]
UDF de búsqueda del Text Extender	1-8

Tabla 6. Restricciones del XML Extender en la utilización de la vía de ubicación (continuación)

Uso de la vía de ubicación	Vía de ubicación soportada

[1] Las UDF de extracción y actualización dan soporte a las vías de ubicación que tienen predicados con atributos, pero no elementos.

El archivo DAD

Para las columnas XML, la DAD principalmente especifica cómo se deben indexar los documentos que están contenidos en una columna XML. La DAD es un documento formateado por XML y reside en el sistema cliente. Si elige validar los documentos XML utilizando una DTD, el archivo DAD se puede asociar a esa DTD. El tipo de datos del archivo DAD es CLOB. Este archivo puede tener hasta 100 KB.

El archivo DAD correspondiente a columnas XML contiene una cabecera XML, que especifica las vías de directorios del sistema cliente para acceder al archivo DAD y DTD, y proporciona un mapa de los datos XML que se deben almacenar en tablas auxiliares para su indexación.

Para especificar el método de acceso y almacenamiento para columnas XML, utilice el código siguiente en el archivo DAD.

<Xcolumn>

Especifica que los datos XML se deben almacenar y recuperar como documentos XML completos en columnas DB2 que están habilitadas para datos XML.

Las columnas habilitadas para XML pertenecen al UDT de XML Extender. Los programas de aplicación pueden incluir la columna en cualquier *tabla de usuario*. El acceso a los datos de columnas XML se realiza principalmente mediante sentencias de SQL y las UDF del XML Extender.

Puede utilizar el asistente de administración del XML Extender o un editor para crear y actualizar la DAD.

Planificación para utilizar colecciones XML

Cuando se planifica para colecciones XML, existen distintas consideraciones a tener en cuenta para componer documentos a partir de datos de DB2, descomponer documentos XML en datos de DB2 o ambas cosas. La secciones siguientes tratan sobre temas de planificación para colecciones XML y sobre consideraciones para la composición y descomposición.

Validación

Una vez elegido un método de acceso y almacenamiento, puede decidir si desea validar los datos. Los datos de XML se validan utilizando una DTD.

Con ello asegura la validez del documento XML y le permite realizar búsquedas estructuradas de los datos XML. La DTD se almacena en el depósito de DTD.

Recomendación: Valide los datos XML utilizando una DTD. Para realizar la validación, es necesario que tenga una DTD en el depósito del XML Extender. Vea "Almacenamiento de una DTD en el depósito de DTD" en la página 76 para conocer cómo insertar una DTD en el depósito. Los requisitos de la DTD varían, dependiendo de si está componiendo o descomponiendo documentos XML.

- Para la composición, sólo puede validar los documentos XML generados utilizando una sola DTD. La DTD a utilizar se especifica en el archivo DAD.
- Para la descomposición, puede validar los documentos utilizando varias DTD diferentes. Es decir, puede descomponer documentos utilizando el mismo archivo DAD, pero puede invocar definiciones DTD que son diferentes. Para hacer referencia a varias DTD, siga estas directrices:
  - Como mínimo una de las DTD debe estar almacenada en la tabla DTD_REF. Todas las DTD pueden estar almacenadas en esta tabla.
  - Las DTD deben tener una estructura común, con diferencias en los subelementos.
  - Debe especificar YES para la validación en el archivo DAD.
  - El ID de sistema del documento XML debe especificar el archivo DTD utilizando una vía de acceso completa.
  - El archivo DAD contiene la especificación de cómo descomponer el documento; por tanto, el usuario sólo puede especificar elementos y atributos comunes para la descomposición. Los elementos y atributos que son exclusivos de una DTD no se pueden descomponer.

Importante: Si decide validar los datos XML, debe hacer la validación antes de insertar los datos en DB2. El XML Extender no permite validar datos que ya estén insertados en DB2.

Consideraciones:

- Debe utilizar una DTD cuando utilice XML como formato de intercambio de datos.
- El validar los datos XML puede tener un pequeño efecto sobre el rendimiento del sistema.
- Sólo puede descomponer elementos y atributos comunes cuando utilice varias DTD para la descomposición.
- Puede descomponer todos los elementos y atributos cuando utilice una sola DTD.
- Puede utilizar una sola DTD para la composición.

El archivo DAD

Para las colecciones XML, el archivo DAD correlaciona la estructura del documento XML con las tablas DB2 utilizadas para componer o descomponer el documento.

Por ejemplo, si tiene un elemento llamado <Tax> en el documento XML, puede necesitar correlacionar <Tax> con una columna llamada TAX. El usuario define la relación existente entre los datos XML y los datos relacionales especificados en la DAD.

El archivo DAD se especifica al habilitar una colección o cuando se utiliza el archivo DAD en *procedimientos almacenados* de colección XML. La DAD es un documento formateado por XML y reside en el sistema cliente. Si elige validar los documentos XML utilizando una DTD, el archivo DAD se puede asociar a esa DTD. Cuando se utiliza como parámetro de entrada de los procedimientos almacenados del XML Extender, el archivo DAD tiene el tipo de datos CLOB.Este archivo puede ser de hasta 100 KB.

Para especificar el método de acceso y almacenamiento para colecciones XML, utilice el código siguiente en el archivo DAD:

<Xcollection>
: Especifica que los datos XML se deben descomponer a partir de documentos XML para formar una colección de tablas relacionales o que se deben componer para crear documentos XML a partir de una colección de tablas relacionales.

  Una colección XML es un nombre virtual que designa un conjunto de tablas relacionales que contienen datos XML. Los programas de aplicación pueden habilitar una colección XML de tablas de usuario cualesquiera. Estas tablas de usuario pueden ser tablas existentes de datos de negocio o tablas recién creadas por el XML Extender. El acceso a los datos de la colección XML se realiza principalmente mediante los procedimientos almacenados que proporciona el XML Extender.

El archivo DAD define la estructura arborescente del documento XML, utilizando las clases de nodos siguientes:

nodo_raíz (root_node)
: Especifica el elemento raíz del documento.

nodo_de_elemento (element_node)
: Identifica un elemento, que puede ser el elemento raíz o un elemento hijo.

nodo_de_texto (text_node)
: Representa el texto CDATA de un elemento.

nodo_de_atributo (attribute_node)
Representa un atributo de un elemento.

La Figura 10 muestra parte de la correlación utilizada en un archivo DAD. Los nodos correlacionan el contenido del documento XML con columnas de una tabla relacional.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  ...
<Xcollection>
<SQL_stmt>
       ...
</SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE DAD SYSTEM "c:\dxx\sample\dtd\getstart.dtd"</doctype>
<root_node>
   <element_node name="Order">       --> Identifica el elemento <Order>
     <attribute_node name="key">     --> Identifica el atributo "key"
     <column name="order_key"/>      --> Define el nombre de la columna,
                                        "order_key", con la que se
                                        correlacionan el elemento y
                                        el atributo

     </attribute_node>
     <element_node name="Customer">  --> Identifica un elemento hijo de <Order>
                                        como <Customer>
        <text_node>                  --> Especifica el texto CDATA del
                                        elemento <Customer>
          <column name="customer">   --> Define el nombre de la columna,
                                        "customer", con la que se
                                        correlaciona el elemento hijo
        </text_node>
     </element_node>
        ...
   </element_node>

      ...
<root_node>
</Xcollection>
</DAD>
```

Figura 10. Definiciones de nodos

En este ejemplo, las dos primeras columnas de la sentencia de SQL tienen elementos y atributos correlacionados con ellas.

XML Extender también da soporte a las instrucciones de proceso para hojas de estilo, utilizando el elemento <stylesheet>. Debe estar dentro del nodo raíz del archivo DAD, con los elementos doctype y prolog definidos para el documento XML. Por ejemplo:

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

</Xcollection>
```

Puede utilizar el asistente de administración del XML Extender o un editor para crear y actualizar el archivo DAD. En la actualidad, el asistente de administración del XML Extender no da soporte al elemento <stylesheet>.

Esquemas de correlación para colecciones XML

Si está utilizando una colección XML, debe seleccionar un *esquema de correlación* que define cómo se representan los datos XML en una base de datos relacional. Debido a que las colecciones XML deben hacer corresponder la estructura jerárquica de los documentos XML con una estructura relacional, es conveniente que comprenda cómo se corresponden las dos estructuras. La Figura 11 en la página 63 muestra cómo la estructura jerárquica se puede correlacionar con columnas de una tabla relacional.

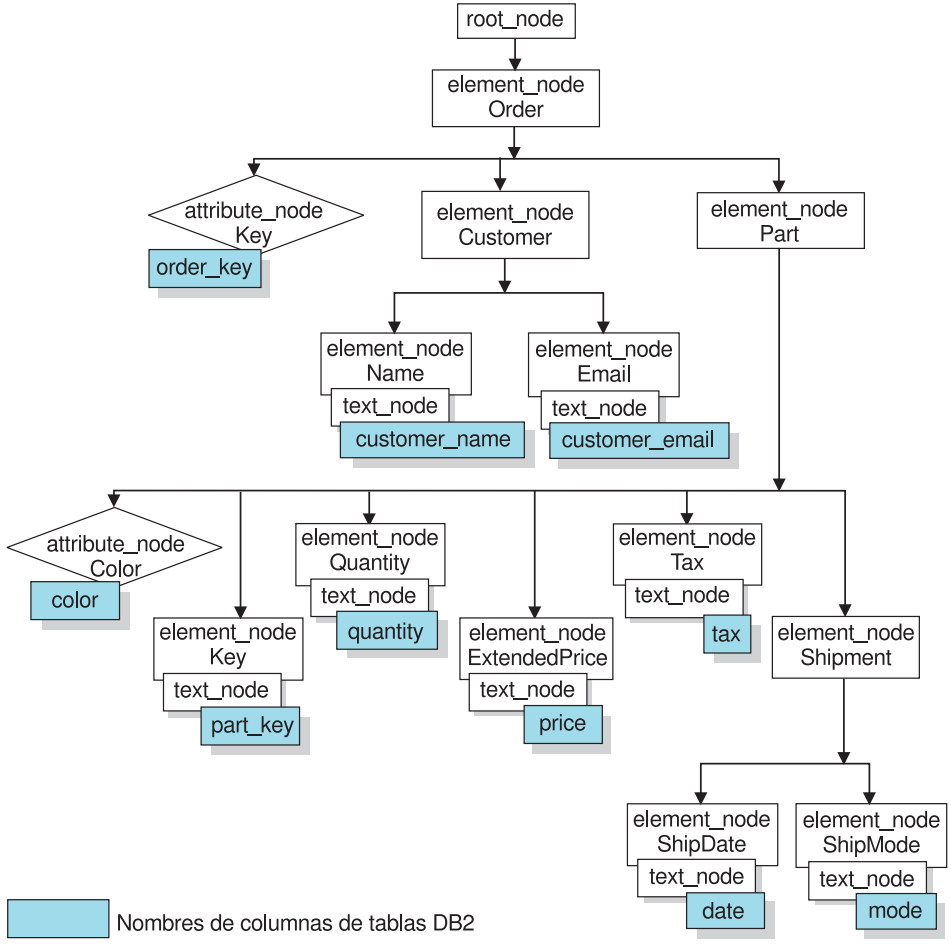


Figura 11. Documento XML estructurado correlacionado con columnas de una tabla relacional

El XML Extender utiliza el esquema de correlación al componer o descomponer documentos XML que residen en diversas tablas relacionales. El XML Extender proporciona un asistente que ayuda al usuario a crear el archivo DAD. Sin embargo, antes de crear el archivo DAD, debe considerar cómo los datos XML se correlacionan con la colección XML.

Tipos de esquemas de correlación: El esquema de correlación se especifica en el elemento <Xcollection> del archivo DAD. El XML Extender proporciona dos tipos de esquemas de correlación: la *correlación SQL* y la *correlación de base de datos relacional (correlación de nodo_RDB)*. Ambos métodos utilizan el modelo XSLT para definir la jerarquía del documento XML.

Correlación SQL

Permite una correlación simple y directa entre datos relacionales y documentos XML mediante una sola sentencia de SQL y el *modelo de datos XSLT*. La correlación SQL se utiliza para la composición de documentos; no se utiliza para la descomposición. La correlación SQL se define mediante el elemento SQL_stmt en el archivo DAD. El contenido de SQL_stmt es una sentencia de SQL válida. SQL_stmt correlaciona las columnas de la cláusula SELECT con elementos o atributos XML que se utilizan en el documento XML. Cuando se definen para componer documentos XML, los nombres de columna indicados en la cláusula SELECT de la sentencia de SQL se utilizan para definir el valor de un *nodo_de_atributo* o un contenido de *nodo_de_texto*. La cláusula FROM define las tablas donde residen los datos; la cláusula WHERE especifica la condición de *unión* y de *búsqueda*.

La correlación SQL proporciona a los usuarios de DB2 la capacidad para correlacionar los datos utilizando SQL. Si utiliza la correlación SQL, debe poder unir todas las tablas de una sola sentencia SELECT para formar una consulta. Si una sola sentencia de SQL no es suficiente, considere la posibilidad de utilizar la correlación de nodo_RDB. Para unir entre sí todas las tablas, es recomendable utilizar la relación *clave primaria-clave foránea* para estas tablas.

Correlación de nodo_RDB

Define la ubicación del contenido de un elemento XML o del valor de un atributo XML para que el XML Extender pueda determinar dónde almacenar o recuperar los datos XML.

El *nodo_RDB* contiene una o más definiciones de nodo para tablas, columnas opcionales y condiciones opcionales. Las tablas y columnas se utilizan para definir cómo deben almacenarse los datos XML en la base de datos. La condición especifica los criterios para seleccionar datos XML o la forma de unir las tablas de la colección XML.

Para definir un esquema de correlación, cree una DAD con un elemento <Xcollection>. La Figura 12 en la página 65 muestra parte de un archivo DAD de ejemplo que contiene una correlación SQL para colecciones XML, la cual compone un conjunto de documentos XML a partir de datos de tres tablas relacionales.

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

...

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

Figura 12. Esquema de correlación SQL

El XML Extender proporciona varios procedimientos almacenados que gestionan los datos de una colección XML. Estos procedimientos almacenados admiten la utilización de ambos tipos de correlación, pero exigen que el archivo DAD siga las directrices indicadas en "Requisitos del esquema de correlación".

Requisitos del esquema de correlación: Las secciones siguientes describen los requisitos de cada tipo de esquema de correlación para colecciones XML.

Requisitos cuando se utiliza la correlación SQL

En este esquema de correlación, debe especificar el elemento SQL_stmt en el elemento <Xcollection> de la DAD. SQL_stmt debe

contener una sola sentencia de SQL que pueda unir varias tablas relacionales con el *predicado* de la consulta. Además, son necesarias las cláusulas siguientes:

- **cláusula SELECT**
  - Asegúrese de que el nombre de la columna sea exclusivo. Si dos tablas tienen el mismo nombre de columna, utilice la palabra clave AS para crear un seudónimo para una de las columnas.
  - Agrupe las columnas de una misma tabla y utilice el nivel jerárquico lógico de las tablas relacionales. Esto significa agrupar las tablas de acuerdo con el nivel de importancia cuando se correlacionan con la estructura jerárquica del documento XML. En la cláusula SELECT, las columnas de las tablas de nivel superior deben ir preceder a las columnas de las tablas de nivel inferior. El ejemplo siguiente muestra la relación jerárquica existente entre las tablas:

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

    En este ejemplo, `order_key` y `customer`, de la tabla ORDER_TAB, tienen el nivel relacional más alto, pues ocupan una posición más alta en el árbol jerárquico del documento XML. Las columnas `ship_id`, `date` y `mode`, de la tabla SHIP_TAB, se encuentran en el nivel relacional más bajo.
  - Utilice una clave adecuada de columna individual para comenzar cada nivel. Si no existe una clave así en una tabla, la consulta debe generar la clave para esa tabla utilizando una expresión de tabla y la función interna generate_unique(). En el ejemplo anterior, `o.order_key` es la clave primaria de ORDER_TAB, y `part_key` es la clave primaria de PART_TAB. Estas claves aparecen al principio de su propio grupo de columnas que se deben seleccionar. Debido a que la tabla SHIP_TAB no tiene una clave primaria, es necesario generar una, que en este caso es `ship_id`. Esta clave aparece listada como primera columna del grupo de la tabla SHIP_TAB. Utilice la cláusula FROM para generar la columna de la clave primaria, tal como muestra el ejemplo siguiente.
- **cláusula FROM**
  - Utilice una expresión de tabla y la función interna generate_unique() para generar una clave simple para tablas que no tienen una clave simple primaria. Por ejemplo:

    ```
    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
    ```

En este ejemplo, la función generate_unique() se convierte al tipo de datos CHAR de TIMESTAMP y se le asigna un seudónimo cuyo nombre es `ship_id`.

- Utilice un seudónimo cuando necesite diferenciar una columna. Por ejemplo, podría utilizar `o` para ORDER_TAB, `p` para PART_TAB y `s` para SHIP_TAB.

- **cláusula WHERE**
  - Especifique una relación clave primaria-clave foránea como condición de unión para asociar entre sí tablas de la colección. Por ejemplo:

    ```
    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
    ```

  - Especifique cualquier otra condición de búsqueda adicional en el predicado. Puede utilizar cualquier predicado de validación.
- **cláusula ORDER BY**
  - Defina la cláusula ORDER BY al final de SQL_stmt.
  - Asegúrese de que los nombres de columna coincidan con los nombres de columna en la cláusula SELECT.
  - Especifique los nombres o identificadores de columna que sirven para identificar de forma inequívoca las entidades de la estructura relacional de la base de datos. Puede crear un identificador utilizando una expresión de tabla y la función interna generate_unique o una función definida por el usuario (UDF).
  - Mantenga el orden descendente de la jerarquía de las entidades. La columna especificada en la cláusula ORDER BY debe ser la primera columna listada para cada entidad. El conservar el orden asegura que no haya duplicados incorrectos en los documentos XML que se crearán.
  - No califique las columnas de ORDER BY según ningún esquema o nombre de tabla.

Aunque SQL_stmt tiene los requisitos indicados anteriormente, es una herramienta potente porque le permite especificar cualquier predicado en la cláusula WHERE, siempre que la expresión contenida en el predicado utilice las columnas de las tablas.

Requisitos cuando se utiliza la correlación de nodo_RDB

Cuando utilice este método de correlación, no utilice el elemento SQL_stmt en el elemento <Xcollection> del archivo DAD. En su lugar, utilice el elemento nodo_RDB para *nodo_de_elemento* y para cada nodo_de_atributo y nodo_de_texto.

- **Nodo_RDB para el nodo_de_elemento superior**

El *nodo_de_elemento superior* del archivo DAD representa el elemento raíz del documento XML. Para especificar un nodo_RDB para el nodo_de_elemento superior, siga estos pasos:

– Especifique todas las tablas que estén asociadas con los documentos XML. Por ejemplo, la correlación siguiente especifica tres tablas en el nodo_RDB del nodo_de_elemento <Order>, que es el nodo_de_elemento superior:

```
<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>
```

El elemento de condición puede estar vacío o puede que no esté si sólo hay una tabla en la colección.

– Si está descomponiendo, o está habilitando la colección XML especificada por el archivo DAD, debe especificar una clave primaria para cada tabla. La clave primaria consta de una o varias columnas; en este segundo caso se denomina clave compuesta. Para especificar la clave primaria, añada una clave de atributo al elemento de tabla del nodo_RDB. Cuando se proporciona una clave compuesta, el atributo de clave se especifica mediante los nombres de columnas de clave separados por un espacio. Por ejemplo:

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

La información especificada para la descomposición de un documento XML no se tiene en cuenta al componer el documento.

– Utilice el atributo orderBy para recomponer documentos XML que contienen elementos o atributos de aparición múltiple y devolverlos a su estructura original. Este atributo le permite especificar el nombre de una columna que se utilizará como clave para preservar el orden del documento. El atributo orderBy forma parte del elemento de tabla en el archivo DAD y es un atributo opcional.

Debe especificar explícitamente el nombre de tabla y el nombre de columna.

- **Nodo_RDB para cada cada nodo_de_atributo y nodo_de_texto.**

En este esquema de correlación, los datos residen en el nodo_de_atributo y nodo_de_texto de cada nodo_de_elemento. Por tanto, el XML Extender necesita conocer en qué lugar de la base de datos debe encontrar los datos. Es necesario que especifique un nodo_RDB para cada nodo_de_atributo y nodo_de_texto, indicando al procedimiento almacenado desde que tabla y columna debe obtener los datos y bajo qué condición de consulta debe obtenerlos. Debe especificar los valores para la tabla y la columna; el valor de la condición es opcional.

– Especifique el nombre de la tabla donde residen los datos de columna. El nombre de la tabla debe incluirse en el nodo_RDB del nodo_de_elemento superior. En este ejemplo, para el nodo_de_texto del elemento <Price>, la tabla está especificada como PART_TAB.

```
<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>
```

– Especifique el nombre de la columna que contiene los datos correspondientes al texto del elemento. En el ejemplo anterior, la columna está especificada como PRICE.

– Especifique una condición si desea generar documentos XML utilizando la condición de consulta. En el ejemplo anterior, la condición está especificada como `price > 2500.00`. Los documentos XML resultantes sólo contendrán los datos que cumplen la condición. La condición debe ser una cláusula WHERE válida.

– Si está descomponiendo un documento, o está habilitando la colección XML especificada por el archivo DAD, debe especificar el tipo de columna para cada attribute_node y text_node. Esto asegura el tipo de datos correcto para cada columna cuando se crean nuevas tablas al habilitar una colección XML. Para especificar los tipos de columna, añada el atributo ″type″ al elemento de columna. Por ejemplo:

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

  La información especificada para la descomposición de un documento XML no se tiene en cuenta al componer el documento.

Si utiliza la correlación de nodo_RDB, no es necesario que proporcione sentencias de SQL. Sin embargo, establecer condiciones de consulta complejas en el elemento de nodo_RDB puede ser más difícil. Por ejemplo, el utilizar una expresión u operación de *unión* es algo menos efectivo que el método que correlaciona SQL con XML.

Requisitos de tamaño de tabla de descomposición

La descomposición utiliza la correlación de nodo_RDB para especificar cómo un documento XML se descompone en tablas DB2 extrayendo los valores de elementos y de atributos en filas de tabla. Los valores para cada documento XML se almacenan en una o más tablas DB2. Cada tabla puede tener un máximo de 1.024 filas descompuestas de cada documento.

Por ejemplo, si un documento XML se descompone en cinco tablas, cada una de las cinco tablas puede tener un máximo de 1.024 filas para este documento concreto. Si la tabla tiene filas para varios documentos, puede tener un máximo de 1.024 filas para cada documento. Si la tabla tiene 20 documentos, puede tener 20.480 filas, 1.024 para cada documento.

La utilización de elementos de aparición múltiple (elementos con vías de ubicación que pueden aparecer más de una vez en la estructura XML) afecta el número de filas. Por ejemplo, un documento que contiene un elemento <Part> que aparece 20 veces, puede descomponerse como 20 filas en una tabla. Cuando utilice elementos de aparición múltiple, considere esta limitación del tamaño de tabla.

Capítulo 4. Administración de datos XML

Las tareas de administración del XML Extender consisten en habilitar la base de datos y las columnas de tabla del usuario para XML y correlacionar los datos XML con estructuras relacionales de DB2. El XML Extender proporciona varias herramientas de administración para el usuario, que le permiten desarrollar una aplicación para realizar tareas de administración o simplemente utilizar un asistente (programa de ayuda). Puede utilizar las herramientas siguientes para realizar tareas de administración para el XML Extender:

- El asistente de administración del XML Extender
- El mandato **dxxadm**
- Los procedimientos almacenados de administración del XML Extender

Este capítulo describe las tareas de administración asociadas al asistente de administración y el mandato **dxxadm**. Los procedimientos almacenados de administración se tratan en "Procedimientos almacenados de administración" en la página 215.

Para realizar las tareas contenidas en este capítulo debe conocer los conceptos y tareas de planificación que se describen en "Planificación de la administración" en la página 48.

Las secciones siguientes describen las tareas de administración del XML Extender:

1. "Arranque del asistente de administración"
2. "Habilitación de una base de datos para XML" en la página 75
3. "Almacenamiento de una DTD en el depósito de DTD" en la página 76
4. "Definición de columnas o colecciones XML" en la página 78
5. "Utilización con columnas XML" en la página 78
6. "Trabajar con colecciones XML" en la página 90

Arranque del asistente de administración

Esta sección contiene información sobre la puesta a punto e invocación del asistente de administración del XML Extender.

Puesta a punto del asistente de administración

Compruebe que ha instalado y configurado el asistente de administración, de acuerdo con las instrucciones del archivo readme correspondientes a su

sistema operativo. Esto incluye asegurarse de que ha ejecutado la sentencia de enlace (BIND) y de que las sentencias CLASSPATH contienen el software necesario.

- Las sentencias de enlace se encuentran en los archivos readme del asistente y en el archivo de iniciación de ejemplo:

  /dxx_install/samples/cmd/getstart_prep.cmd

- La sentencia CLASSPATH debe ser parecida a la siguiente (los saltos de línea son únicamente para la presentación):

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

  Importante: El asistente requiere un nombre de vía de acceso sin espacio. Si la instalación por omisión disponible es IBM DB2 Universal Database V7.1, SQLLIB\java se encuentra debajo del directorio Archivos de programa, *copie* el código Java en una vía de acceso más simple. No mueva el código Java y cambie CLASSPATH; el Centro de control requiere que se especifique CLASSPATH durante la instalación.

El asistente de administración del XML Extender utiliza un archivo de clases. El nombre de archivo completo del archivo principal de clases de administración del XML Extender es:

com.ibm.dxx.admin.Admin.

Modifique este archivo para que el sistema invoque al asistente.

- Para invocar utilizando JDK, escriba:

  java -classpath *classpath* com.ibm.dxx.admin.Admin

- Para invocar utilizando JRE, escriba:

  jre -classpath *classpath* com.ibm.dxx.admin.Admin

  donde *classpath* especifica:

  – La variable de entorno %CLASSPATH% que especifica en qué lugar se encuentran los archivos de clases del asistente de administración. Al utilizar esta opción, el sistema CLASSPATH debe señalar hacia el directorio *dxx_install*/dxxadmin, que contiene los siguientes archivos: dxxadmin.jar, xml4j.jar y db2java.zip. Por ejemplo:

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

  – Una alteración temporal de la variable de entorno %CLASSPATH% con punteros hacia los archivos del directorio *dxx_install*/dxxadmin, desde el cual se ejecuta el asistente de administración del XML Extender. Por ejemplo:

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

Opcionalmente, puede especificar los siguientes parámetros durante la ejecución:

url Vía de acceso de la dirección URL completamente calificada que conduce a la fuente de datos de IBM DB2 UDB con la que desea conectar. Por ejemplo: jdbc:db2://dxx.stl.ibm.com:8080/guidb. Lleva la etiqueta "Dirección" en el asistente.

id usuario
Id de usuario que se utiliza para acceder a la fuente de datos anterior. Por ejemplo: db2guest.

contraseña
Contraseña para el id de usuario anterior. Por ejemplo: guest.

controlador
Nombre de controlador JDBC de la dirección URL anterior. Valor por omisión: COM.ibm.db2.jdbc.net.DB2Driver. Lleva la etiqueta "Controlador JDBC" en el asistente.

Consulte el apartado "Invocación del asistente de administración" para obtener más información acerca de estos valores.

Invocación del asistente de administración

Siga estos pasos para invocar el asistente de administración del XML Extender.

1. Invoque el asistente.

   Para Windows NT:

   En el escritorio, haga una doble pulsación sobre el icono del asistente de administración del XML Extender.

   Para AIX, Sun Solaris y Linux:

   Ejecute el archivo `dxxadmin`.

   Se abrirá la ventana Conexión con el asistente de administración.

Cuando se invoca el asistente de administración del XML Extender, se visualiza la ventana Conexión. Conéctese a la base de datos que desee utilizar cuando trabaje con datos XML. El XML Extender le conecta con la instancia actual.

2. En el campo **Dirección** escriba la dirección JDBC URL totalmente calificada que conduce a la fuente de datos de IBM DB2 UDB con la que desea conectar. La dirección tiene la sintaxis siguiente:

   Para configuraciones autónomas (valor recomendado):

   `jdbc:db2:`*`nombre_base_datos`*

   Donde:

nombre_base_datos
Es la base de datos con la que desea conectar y que contiene los documentos XML.

Por ejemplo:

```
jdbc:db2:sales_bd
```

Para configuraciones de red:

```
jdbc:db2://nombre_servidor:número_puerto/nombre_base_datos
```

Donde:

nombre_servidor
Es el nombre del servidor donde reside el XML Extender.

número_puerto
Es el número de puerto utilizado para la conexión con el servidor. Para determinar el número de puerto, entre el mandato siguiente desde la línea de mandatos de DB2 en la máquina servidor:

```
db2jstrt número_puerto
```

Los usuarios de Windows NT pueden consultar el número de puerto en el siguiente archivo \winnt\system32\driver\etc\services.

nombre_base_datos
Es la base de datos con la que desea conectar y que contiene los documentos XML.

Por ejemplo:

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

3. En los campos **ID de usuario** y **Contraseña**, entre o verifique el ID de usuario y la contraseña de DB2 para la base de datos con la que se está conectando.
4. En el campo **Controlador JDBC**, verifique el nombre de controlador JDBC para la dirección especificada, utilizando los valores siguientes:

   Para configuraciones autónomas (valores por omisión y recomendado):

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

   Para configuraciones de red:

   ```
   COM.ibm.db2.jdbc.net.DB2DRIVER
   ```
5. Pulse **Finalizar** para conectarse al asistente y avanzar hasta la ventana Área de ejecución.

La ventana Área de ejecución proporciona acceso a cinco asistentes de administración. Con estos asistentes se puede:

- Habilitar una base de datos
- Añadir una DTD al depósito de DTD
- Trabajar con archivos DAD para:
  - columnas XML
  - colecciones XML
- Trabajar con columnas XML
- Trabajar con colecciones XML

Habilitación de una base de datos para XML

Para almacenar o recuperar documentos XML desde DB2 con el XML Extender, debe habilitar la base de datos para XML. El XML Extender habilita la base de datos a la que está conectado el usuario, utilizando la instancia actual.

Cuando habilita una base de datos para XML, el XML Extender:

- Crea todos los tipos definidos por el usuario (los UDT) y todas las funciones definidas por el usuario (las UDF).
- Crea tablas de control y las llena con los metadatos necesarios para el XML Extender
- Crea el esquema db2xml y asigna los privilegios necesarios

  El nombre completo de una función XML es *nombre_esquema.nombre_función*, donde *nombre_esquema* es un identificador que permite crear agrupaciones lógicas de *objetos* de SQL. Puede utilizar el nombre completo en cualquier lugar donde haga referencia a una UDF o a un UDT. Puede también omitir el nombre de esquema cuando haga referencia a una UDF o a un UDT; este caso, DB2 utiliza la vía de acceso de la función para determinar el tipo de función o de datos que el usuario desea.

Utilización del asistente de administración

Siga los pasos siguientes para habilitar una base de datos para datos XML:

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Habilitar base de datos** desde la ventana Área de ejecución para habilitar la base de datos actual.

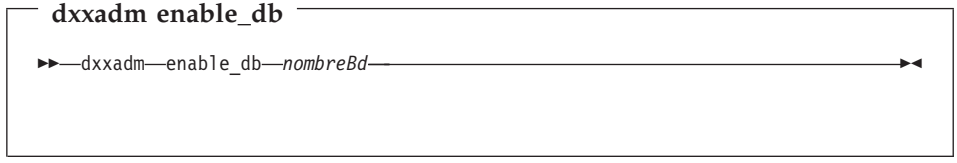
   Si ya está habilitada una base de datos, sólo puede seleccionar **Inhabilitar una base de datos**.

   Cuando la base de datos está habilitada, se le devuelve a la ventana Área de ejecución.

Desde el shell de mandatos de DB2

Entre **dxxadm** en la línea de mandatos, especificando la base de datos que se debe habilitar.

Sintaxis:



Parámetros:

nombreBd
Es el nombre de la base de datos que se debe habilitar.

Ejemplo: Habilitar una base de datos existente llamada SALES_DB.

```
dxxadm enable_db SALES_DB
```

Almacenamiento de una DTD en el depósito de DTD

Puede utilizar una DTD para validar datos XML contenidos en una columna XML o colección XML. La DTD valida la columna XML y sirve para definir archivos de DAD que se utilizan para la búsqueda estructural de XML y para la composición y descomposición de colecciones.

Todas las DTD se almacenan en el depósito DTD, que es una tabla DB2 llamada DTD_REF. Su nombre de esquema es db2xml. Cada DTD de la tabla DTD_REF tiene un ID exclusivo. El XML Extender crea la tabla DTD_REF cuando el usuario habilita una base de datos para XML.

Vea “Planificación para utilizar columnas XML” en la página 51 y “Planificación para utilizar colecciones XML” en la página 58 para conocer más sobre la utilización de las DTD.

Puede insertar la DTD desde el shell de mandatos de DB2 o utilizando el asistente de administración.

Utilización del asistente de administración

Siga los pasos siguientes para insertar una DTD:

1. Ponga a punto y arranque el asistente de administración. Vea “Arranque del asistente de administración” en la página 71 para obtener detalles.
2. Pulse **Importar una DTD** desde la ventana Área de ejecución para importar un archivo DTD existente al depósito de DTD de la base de datos actual. Se visualizará la ventana Importar una DTD.
3. Escriba el nombre de archivo DTD en el campo **Nombre de archivo DTD** o pulse **...** para localizar un archivo DTD existente.
4. Escriba el ID de DTD en el campo **ID de DTD**.

El ID de DTD es un identificador para la DTD y puede ser la vía de acceso que especifique la ubicación de la DTD en el sistema local. El ID de DTD debe coincidir con el valor especificado en el archivo DAD para el elemento <DTDID>.

5. Opcionalmente, escriba el nombre del autor de la DTD en el campo **Autor**.

   El XML Extender visualizará automáticamente el nombre del autor si está especificado en la DTD.

6. Pulse **Finalizar** para insertar la DTD en la tabla de depósito DTD, DB2XML.DTD_REF, y volver la ventana Área de ejecución.

Desde el shell de mandatos de DB2

Emita una sentencia INSERT de SQL para la tabla DTD_REF, utilizando el esquema de la Tabla 7:

Tabla 7. Esquema para la tabla DTD_REF de DTD

Nombre de columna	Tipo de datos	Descripción
DTDID	VARCHAR(128)	Clave primaria (exclusiva y no nula). La clave primaria sirve para identificar la DTD y debe coincidir con el ID de SISTEMA especificado en la línea DOCTYPE de cada documento XML, si se utiliza la validación. Cuando se especifica la clave primaria en el archivo DAD, este archivo debe seguir el esquema definido por la DTD.
CONTENT	XMLCLOB	Contenido de la DTD.
USAGE_COUNT	INTEGER	Número de columnas XML y de colecciones XML de la base de datos que utilizan esta DTD para definir una DAD.
AUTHOR	VARCHAR(128)	Autor de la DTD; esta información de entrada es opcional.
CREATOR	VARCHAR(128)	ID de usuario que realiza la primera inserción.
UPDATOR	VARCHAR(128)	ID de usuario que realiza la última actualización.

Por ejemplo:

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

Importante para colecciones de XML: El ID de DTD es una vía de acceso que especifica la ubicación de la DTD en el sistema local. El ID de DTD debe coincidir con el valor especificado en el archivo DAD para el elemento <DTDID>.

Definición de columnas o colecciones XML

Las secciones siguientes describen cómo configurar y definir la base de datos para columnas o colecciones XML, y cómo preparar los necesarios esquemas de correlación de datos.

Las secciones que siguen son:

- "Utilización con columnas XML"
- "Trabajar con colecciones XML" en la página 90

Utilización con columnas XML

Para configurar columnas XML, debe definir el archivo DAD para acceder a los datos XML y habilitar columnas para datos XML en una tabla XML. Un concepto importante en la creación de la DAD es la comprensión de la sintaxis de la vía de acceso a la ubicación puesto que se utiliza para correlacionar el elemento y los valores de atributo que desea indexar a las tablas DB2. Vea "Vía de ubicación" en la página 55 para obtener más información sobre la vía de acceso a la ubicación y su sintaxis.

Creación o edición del archivo DAD

Cuando especifica un archivo DAD, define los atributos y elementos esenciales de los datos que deben buscarse. El XML Extender utiliza esta información para crear tablas auxiliares, que permiten al usuario indexar los datos para acceder a ellos con rapidez. Vea "El archivo DAD" en la página 58 para conocer cuestiones de planificación referentes a la creación del archivo DAD.

Antes de comenzar

- Debe conocer la estructura jerárquica de los datos XML, para así poder definir elementos clave y atributos que permitan la indexación y una búsqueda rápida de los datos.
- Prepare e inserte la DTD del documento XML en la tabla DTD_REF. Este paso es necesario para la validación.

Utilización del asistente de administración

Siga los pasos siguientes para crear un archivo DAD:

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.

2. Pulse **Utilización de archivos DAD** desde la ventana Área de ejecución para editar o crear un archivo XML de DAD. Se abrirá la ventana Especificar un archivo DAD.
3. Elija si desea editar un archivo DAD existente o crear uno nuevo.
   - **Para editar una DAD existente:**
     a. Pulse **...** para localizar un archivo DAD existente en el menú desplegable o escriba el nombre de archivo DAD en el campo **Nombre de archivo**.
     b. Verifique que el asistente reconoce el archivo DAD especificado.
        – Si el asistente reconoce el archivo DAD especificado, puede seleccionar **Siguiente** y el campo **Tipo** muestra Columna XML.
        – Si el asistente no reconoce el archivo DAD especificado, no se puede seleccionar **Siguiente**. Escriba de nuevo el nombre del archivo DAD en el campo **Nombre de archivo** o bien pulse **Abrir** para localizar de nuevo un archivo DAD existente. Repita estos pasos hasta que pueda seleccionar **Siguiente**.
     c. Pulse **Siguiente**.
   - **Para crear una DAD nueva:**
     a. Deje el campo **Nombre de archivo** en blanco.
     b. Desde el menú **Tipo**, pulse **Columna XML**.
     c. Pulse **Siguiente**.
4. En la ventana Seleccionar validación, elija si desea validar o no los documentos XML.
   - Para validar:
     a. Pulse **Validar documentos XML con la DTD**.
     b. En el menú **ID de DTD**, seleccione la DTD que se utilizará para la validación.

     Si no ha importado ninguna DTD al depósito de DTD de la base de datos, no podrá validar documentos XML.
   - Pulse **NO validar documentos XML con la DTD** para continuar sin validar los documentos XML.
5. Pulse **Siguiente**.
6. En la ventana Tablas auxiliares, elija si desea añadir una nueva tabla auxiliar, editar una tabla auxiliar existente o suprimirla.
   - **Para añadir una nueva tabla auxiliar o columna de tabla auxiliar:**

     Para añadir una nueva tabla auxiliar, defina las columnas de la tabla. Realice los pasos siguientes para cada columna de una tabla auxiliar.
     a. Llene los campos del recuadro **Detalles**, en la ventana Tablas auxiliares.

1) **Nombre de tabla**: Escriba el nombre de la tabla donde reside la columna. Por ejemplo:

   ```
   ORDER_SIDE_TAB
   ```

2) **Nombre de columna**: Escriba el nombre de la columna. Por ejemplo:

   ```
   CUSTOMER_NAME
   ```

3) **Tipo**: Seleccione el tipo de la columna en el menú. Por ejemplo:

   ```
   XMLVARCHAR
   ```

4) **Longitud (sólo para tipo VARCHAR)**: Escriba el número máximo de caracteres VARCHAR. Por ejemplo:

   ```
   30
   ```

5) **Vía de acceso**: Escriba la vía de acceso a la ubicación del elemento o atributo. Por ejemplo:

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

   Vea "Vía de ubicación" en la página 55 para obtener la sintaxis de la vía de acceso a la ubicación.

6) **Aparición múltiple**: Seleccione **No** o **Sí** en el menú.

   Indica si la vía de acceso a la ubicación de este elemento o atributo se puede utilizar más de una vez en un documento.

   Importante Si especifica aparición múltiple para una columna, sólo puede especificar una columna en la tabla auxiliar que contenga la columna.

b. Pulse **Añadir** para añadir una columna.

c. Continúe añadiendo, editando o eliminando columnas para la tabla auxiliar o pulse **Siguiente**.

- **Para editar una columna de una tabla auxiliar existente:**

  Puede actualizar una tabla auxiliar cambiando las definiciones de las columnas existentes.

  a. Pulse sobre la tabla auxiliar y el nombre de columna que desea editar.

  b. Edite los campos del recuadro **Detalles**.

  c. Pulse **Cambiar** para guardar los cambios.

  d. Continúe añadiendo, editando o eliminando columnas para cada tabla auxiliar o pulse **Siguiente**.

- **Para eliminar una columna de una tabla auxiliar existente:**

  a. Pulse sobre la tabla auxiliar y la columna que desee eliminar.

  b. Pulse **Eliminar**.

  c. Continúe añadiendo, editando o eliminando columnas de tablas auxiliares o pulse **Siguiente**.

- **Para eliminar una tabla auxiliar existente:**

  Para eliminar toda una tabla auxiliar, suprima cada columna de la tabla.

  a. Pulse en cada columna de la tabla auxiliar para la tabla que desee eliminar.
  b. Pulse **Eliminar**.
  c. Continúe añadiendo, editando o eliminando columnas de tablas auxiliares o pulse **Siguiente**.

7. En el campo **Nombre de archivo** de la ventana Especificar DAD, escriba un nombre de archivo de salida para el archivo DAD modificado.
8. Pulse **Finalizar** para guardar el archivo DAD y volver a la ventana Área de ejecución.

Desde el shell de mandatos de DB2

El archivo DAD es un archivo XML que se puede crear en un editor de texto cualquiera.

Siga los pasos siguientes para crear un archivo DAD:

1. Abra un editor de texto.
2. Cree la cabecera del archivo DAD, utilizando la sintaxis siguiente:

   ```
   <?xml versión="1.0"?>
   <!DOCTYPE DAD SYSTEM "path\dtd\dad.dtd"> --> vía de acceso y nombre
          de archivo de la DTD para el archivo DAD
   ```

3. Inserte los códigos <DAD></DAD>.
4. Dentro del código <DAD>, opcionalmente puede especificar el identificador DTDID, que sirve para asociar el archivo DAD con la DTD del documento XML para la validación:

   ```
   <dtdid>path\dtd_name.dtd</dtdid> --> vía de acceso y nombre
          de archivo de la DTD para la aplicación
   ```

   El ID de DTD es necesario para la validación y debe coincidir con el valor de ID de DTD que se utiliza cuando se inserta la DTD en la tabla de referencia de DTD (db2xml.DTD_REF).
5. Especifique si desea realizar la validación (es decir, utilizar una DTD para asegurarse de que el documento XML es válido). Por ejemplo:

   ```
   <validation>SÍ</validation>   --> especifique YES o NO
   ```

   Si especifica SÍ, es necesario que haya especificado un DTDID en el paso anterior y que haya insertado una DTD en la tabla DTD_REF.
6. Utilice el elemento <Xcolumn> para definir el método de acceso y de almacenamiento como columna XML.

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

7. Define cada tabla auxiliar y los elementos y atributos importantes que se deben indexar para realizar búsquedas estructurales. Siga los pasos siguientes para cada tabla. Estas instrucciones utilizan ejemplos procedentes de un archivo DAD de muestra, que aparece en "Archivo DAD: columna XML" en la página 277:
   a. Inserte los códigos <TABLE></TABLE> y el nombre del atributo.

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

   b. A continuación del código <TABLE>, inserte un código <COLUMN> y sus atributos para cada columna de la tabla.
      - ***name***: el nombre de la columna
      - ***type***: el tipo de columna
      - ***path***: la vía de acceso a la ubicación del elemento o atributo. Vea "Vía de ubicación" en la página 55 para obtener la sintaxis de la vía de acceso a la ubicación.
      - ***multi_occurrence***: indicación de si el elemento o atributo se puede utilizar más de una vez en un documento

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

8. Compruebe que hay un código </TABLE> a continuación de la última columna de la definición.
9. Compruebe que hay un código </Xcolumn> a continuación del último código </TABLE>.
10. Compruebe que hay un código </DAD> a continuación del código </Xcolumn>.

Creación o alteración de una tabla XML

Para guardar documentos XML inalterados en una tabla, debe crear o modificar una tabla que contiene una columna con un tipo definido por el usuario (UDT) XML. Esta tabla se denomina *tabla XML*, y es una tabla que contiene documentos XML. La tabla puede ser una tabla alterada o una tabla nueva. Cuando una tabla contiene una columna de tipo XML, puede habilitar la columna para XML.

Puede modificar una tabla existente con una columna de tipo XML utilizando el asistente de administración o utilizando el shell de mandatos de DB2.

Utilización del asistente de administración

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con columnas XML** desde la ventana Área de ejecución. Se abrirá la ventana Seleccionar tarea.
3. Pulse **Añadir columna XML**. Se abrirá la ventana Añadir columna XML.
4. Seleccione el nombre de la tabla desde el menú desplegable **Nombre de tabla** o escriba el nombre de la tabla que desee modificar. Por ejemplo:
   ```
   SALES_DB
   ```
5. Escriba el nombre de la columna que debe añadirse a la tabla en el campo **Nombre de columna**. Por ejemplo:
   ```
   ORDER
   ```
6. Seccione el UDT para la columna desde el menú desplegable **Tipo de columna**. Por ejemplo:
   ```
   XMLVARCHAR
   ```
7. Pulse **Finalizar** para añadir la columna de tipo XML.

Desde el shell de mandatos de DB2

Cree o modifique una tabla con una columna de tipo XML en la cláusula de columna de la sentencia CREATE TABLE o ALTER TABLE.

Ejemplo: En la aplicación ventas, puede almacenar un pedido, utilizando un formato de línea XML, en una columna llamada ORDER de una tabla de aplicación llamada SALES_TAB. Esta tabla también tiene las columnas INVOICE_NUM y SALES_PERSON. Debido a que es un pedido pequeño, para almacenarlo puede utilizar el tipo XMLVARCHAR. La clave primaria es INVOICE_NUM. La siguiente sentencia CREATE TABLE crea la tabla con una columna de tipo XML:

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

Habilitación de columnas XML

Para guardar un documento XML en una base de datos DB2, debe habilitar una columna para XML. La habilitación de una columna permite indexarla para poder buscarla con rapidez. Puede habilitar una columna mediante el asistente de administración del XML Extender o utilizando el shell de mandatos de DB2. La columna debe ser de tipo XML.

Cuando el XML Extender habilita una columna XML, realiza estas acciones:

- Lee el archivo DAD para opcionalmente:
  - Validar el archivo DAD por comparación con la DTD del archivo.
  - Obtener el DTDID a partir de la tabla DTD_REF, si está especificado.

  - Crear tablas auxiliares para indexar en la columna XML.
  - Preparar la columna para que contenga datos XML.
- Opcionalmente crea una *vista predefinida* de la tabla XML y las tablas auxiliares, si está definido.
- Especifica un valor de ID RAÍZ, si no se ha especificado ninguno.

Después de habilitar la columna XML, puede
- Crear índices en las tablas auxiliares
- Insertar documentos XML en la columna XML
- Consultar, actualizar o buscar en los documentos XML de la columna XML.

Antes de comenzar
Puede crear una tabla XML creando o alterando una tabla DB2 que contenga una columna que sea un UDT de XML.

Utilización del asistente de administración
Siga los pasos siguientes para habilitar columnas XML:

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con columnas XML** desde la ventana Área de ejecución para ver las tareas relacionadas con columnas del XML Extender. Se abrirá la ventana Seleccionar tarea.
3. Pulse **Habilitar columna** y, a continuación, **Siguiente** para habilitar una columna de una tabla existente de la base de datos.
4. En el campo **Nombre de tabla**, seleccione la tabla donde reside la columna XML.Por ejemplo

   ```
   SALES_TAB
   ```
5. En el campo **Nombre de columna**, seleccione la columna que desea habilitar. Por ejemplo:

   ```
   ORDER
   ```

   La columna debe existir y ser de tipo XML.
6. Escriba la vía de acceso y el nombre de archivo DAD en el campo **Nombre de archivo DAD** o pulse **...** para localizar un archivo DAD existente. Por ejemplo:

   ```
   c:\dxx\samples\dad\getstart.dad
   ```
7. Opcionalmente, escriba en el campo **Espacio de tablas** el nombre de un espacio de tablas existente.

   El espacio de tablas contiene las tablas auxiliares que el XML Extender creó. Si especifica un espacio de tablas, las tablas auxiliares se crean en el espacio de tablas especificado. Si no especifica ningún espacio de tablas, las tablas auxiliares se crean en el espacio de tablas por omisión.

8. Opcionalmente, escriba en el campo **Vista predefinida** el nombre de la vista predefinida.

   Si se especifica, la vista por omisión se crea automáticamente cuando se habilita la columna y se une la tabla XML y todas las tablas auxiliares relacionadas.

9. Opcionalmente, escriba en el campo **ID raíz** el nombre de columna correspondiente a la clave primaria de la aplicación. Es recomendable hacer esto.

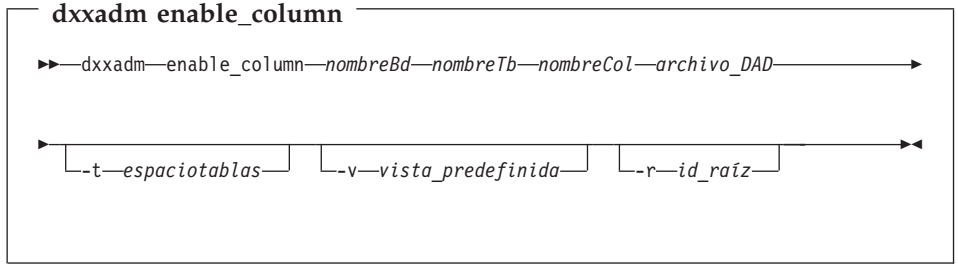
   El XML Extender utiliza el valor de ID RAÍZ como un identificador exclusivo para asociar todas las tablas auxiliares con la tabla de aplicación. Si no se especifica el ID raíz, el XML Extender añade la columna DXXROOT_ID a la tabla de aplicación y genera un identificador.

10. Pulse **Finalizar** para habilitar la columna XML, crear las tablas auxiliares y volver a la ventana Área de ejecución.
    - Si la columna se habilita satisfactoriamente, se visualiza un mensaje `Columna habilitada satisfactoriamente`.
    - Si la columna no se habilita satisfactoriamente, se visualiza una ventana de errores. Corrija los valores del campo de entrada hasta que la columna se habilite correctamente.

Desde el shell de mandatos de DB2

Para habilitar una columna XML, entre el mandato siguiente:

Sintaxis:



Parámetros:

nombreBd
: Es el nombre de la base de datos.

nombreTb
: Es el nombre de la base de datos donde reside la columna que se debe habilitar.

nombreCol
: Es el nombre de la columna XML que se debe habilitar.

archivo_DAD
Es el nombre del archivo donde reside la definición de acceso a documento (DAD).

espaciotablas
Es un espacio de tablas creado previamente que contiene las tablas auxiliares que creó el XML Extender. Si no se especifica este parámetro, se utiliza el espacio de tablas predefinido.

vista_predefinida
Opcional. Es el nombre de la vista predefinida que el XML Extender creó para unir una tabla de aplicación y todas las tablas auxiliares relacionadas.

id_raíz Opcional. El nombre de columna de la clave primaria en la tabla de aplicación y un identificador exclusivo que asocia todas las tablas auxiliares con la tabla de aplicación. El XML Extender utiliza el valor de *id_raíz* como identificador exclusivo para unir todas las tablas auxiliares con la tabla de aplicación. Es recomendable especificar el ID RAÍZ. Si no se especifica el ID RAÍZ, el XML Extender añade la columna DXXROOT_ID a la tabla de aplicación y genera un identificador.

Restricción: Si la tabla de aplicación tiene una columna llamada DXXROOT_ID, pero esta columna no contiene el valor de *id_raíz*, debe especificar este parámetro; de lo contrario se produce un error.

Ejemplo: El ejemplo siguiente habilita una columna utilizando el shell de mandatos de DB2. El archivo DAD y el documento XML se pueden encontrar en el "Apéndice B. Ejemplos" en la página 275.

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

Este ejemplo habilita la columna ORDER de la tabla SALES_DB.SALES_TAB. El archivo DAD es `getstart.dad`, la vista por omisión es sales_order_view y el ID RAÍZ es INVOICE_NUM.

De acuerdo con este ejemplo, la tabla SALES_TAB tiene el esquema siguiente:

Nombre de columna	**INVOICE_NUM**	**SALES_PERSON**	**ORDER**
Tipo de datos	CHAR(6)	VARCHAR(20)	XMLVARCHAR

De acuerdo con la especificación DAD, se crean las tablas auxiliares siguientes:

ORDER_SIDE_TAB:

Nombre de columna	ORDER_KEY	CUSTOMER	INVOICE_NUM
Tipo de datos	INTEGER	VARCHAR(50)	CHAR(6)
Vía de acceso	/Order/@key	/Order/Customer/Name	N/D

PART_SIDE_TAB:

Nombre de columna	PART_KEY	PRICE	INVOICE_NUM
Tipo de datos	INTEGER	DOUBLE	CHAR(6)
Vía de acceso	/Order/Part/@key	/Order/Part/ExtendedPrice	N/D

SHIP_SIDE_TAB:

Nombre de columna	FECHA	INVOICE_NUM
Tipo de datos	FECHA	CHAR(6)
Vía de acceso	/Order/Part/Shipment/ShipDate	N/D

Todas las tablas auxiliares tienen la columna INVOICE_NUM del mismo tipo, pues la clave primaria INVOICE_NUM especifica el ID RAÍZ en la tabla de aplicación. Una vez habilitada la columna, el valor de INVOICE_NUM se inserta en las tablas auxiliares. Si especifica el parámetro *vista_predefinida* al habilitar la columna XML, ORDER, se crea la vista predefinida sales_order_view. La vista asocia las tablas anteriores utilizando la sentencia siguiente:

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

Si el espacio de tablas se especifica en el mandato **enable_column**, las tablas auxiliares se crean en el espacio de tablas especificado. Si el espacio de tablas no se especifica, las tablas auxiliares se crean en el espacio de tablas predefinido.

Indexación de tablas auxiliares

Después de habilitar una columna XML y crear las tablas auxiliares, puede indexar las tablas auxiliares. Las tablas auxiliares contienen los datos XML en las columnas que ha especificado cuando ha creado el archivo DAD. La indexación de estas tablas ayuda a mejorar el rendimiento de las consultas de documentos XML.

Antes de comenzar

- Cree un archivo DAD que especifique tablas auxiliares para la estructura de documento XML.
- Habilite la columna XML utilizando el archivo DAD; con lo cual se crean las tablas auxiliares.

Mandato DB2 CREATE INDEX

Utilice el mandato DB2 CREATE INDEX.

Ejemplo:

En el ejemplo siguiente se crean índices en cuatro tablas auxiliares:

```
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)
```

Inhabilitación de columnas XML

Inhabilite una columna si necesita actualizar un archivo DAD para la columna XML o si desea suprimir la columna XML o la tabla que contiene la columna. Una vez inhabilitada la columna, puede volver a habilitar la columna con el archivo DAD actualizado, suprimir la columna o llevar a cabo otras tareas. Puede inhabilitar una columna utilizando el asistente de administración del XML Extender o utilizando el shell de mandatos de DB2.

Cuando el XML Extender habilita una columna XML, realiza estas acciones:

- Suprima la entrada para la columna de la tabla XML_USAGE.
- Elimine las tablas auxiliares asociadas con esta columna.

Importante: Si elimina una tabla con una columna XML, sin inhabilitar primero la columna, el XML Extender no puede eliminar ninguna tabla auxiliar asociada con la columna XML, lo que podría generar resultados inesperados.

Antes de comenzar

Asegúrese de que la columna XML que debe inhabilitarse exista en la base de datos DB2 actual.

Utilización del asistente de administración

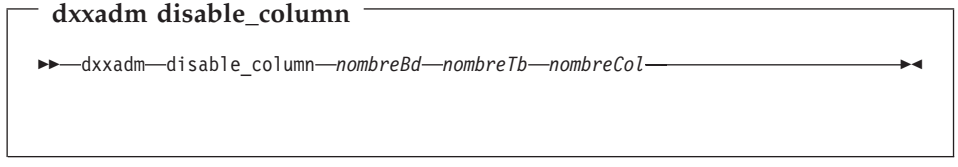
Realice los pasos siguientes para inhabilitar columnas XML:

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con columnas XML** desde la ventana Área de ejecución para ver las tareas relacionadas con columnas del XML Extender. Se abrirá la ventana Seleccionar tarea.
3. Pulse **Inhabilitar columna** y, a continuación, **Siguiente** para inhabilitar una columna existente de la base de datos.
4. En el campo **Nombre de tabla**, seleccione la tabla donde reside la columna XML.
5. Seleccione la columna que va a inhabilitar en el campo **Nombre de columna**.
6. Pulse **Finalizar**.
   - Si la columna se inhabilita satisfactoriamente, se visualiza un mensaje `Columna inhabilitada satisfactoriamente`.
   - Si la columna no se inhabilita satisfactoriamente, se visualiza un recuadro de errores. Corrija los valores del campo de entrada hasta que la columna se inhabilite satisfactoriamente.

Desde el shell de mandatos de DB2

Para inhabilitar una columna XML, entre el mandato siguiente:

Sintaxis:



Parámetros:

nombreBd
: Es el nombre de la base de datos.

nombreTb
: Es el nombre de la tabla que contiene la columna que debe inhabilitarse.

nombreCol
Es el nombre de la columna XML que se está inhabilitando.

Ejemplo: En el ejemplo siguiente se inhabilita una columna utilizando el shell de mandatos de DB2. El archivo DAD y el documento XML se pueden encontrar en el "Apéndice B. Ejemplos" en la página 275.

```
dxxadm disable_column SALES_DB sales_tab order
```

En este ejemplo, se inhabilita la columna ORDER de la tabla SALES_DB.SALES_TAB.

Cuando la columna está inhabilitada, las tablas auxiliares se eliminan.

Trabajar con colecciones XML

Para configurar colecciones XML es necesario crear un esquema de correlación y, opcionalmente, habilitar la colección con un nombre virtual que asocia las tablas DB2 con un archivo DAD.

Aunque habilitar la colección XML no es obligatorio, proporciona un mejor rendimiento.

Creación o edición del archivo DAD para el esquema de correlación

Si se utilizan colecciones XML es necesario crear un archivo DAD. El archivo DAD define la relación existente entre datos XML y varias tablas relacionales. El XML Extender utiliza el archivo DAD para:

- Componer un documento XML a partir de datos relacionales
- Descomponer un documento XML en datos relacionales

Puede utilizar dos métodos para correlacionar datos entre las tablas XML y la tabla DB2: la correlación SQL y la correlación de nodo_RDB:

Correlación SQL
Utiliza un elemento de sentencia de SQL para especificar la consulta SQL que obtiene las tablas y columnas utilizadas para contener los datos XML. La correlación SQL sólo se puede utilizar para componer documentos XML.

Correlación de nodo_RDB
Utiliza un elemento exclusivo del XML Extender, el nodo de Base de datos relacional, o nodo_RDB, que especifica tablas, columnas, condiciones y secuencias de ordenación para los datos XML. La correlación de nodo_RDB permite realizar correlaciones más complejas que las efectuadas mediante una sentencia de SQL. La correlación de nodo_RDB se puede utilizar tanto para componer como para descomponer documentos XML.

Ambos métodos de correlación utilizan el *modelo de datos XPath*, que se describe en "El archivo DAD" en la página 60.

Antes de comenzar

- Debe correlacionar la relación existente entre sus tablas DB2 y el documento XML. Para ello debe correlacionar la jerarquía del documento XML y especificar cómo los datos del documento se correlacionan con una tabla DB2.
- Si tiene intención de validar los documentos XML, inserte la DTD para el documento XML que está componiendo o descomponiendo en la tabla de referencia de DTD, db2xml.DTD_REF.

Composición de documentos XML con correlación SQL

Utilice correlación SQL cuando componga documentos XML y desee utilizar SQL.

Utilización del asistente de administración: Efectúe los pasos siguientes para crear un archivo DAD utilizando una correlación SQL para las colecciones XML.

Para crear un archivo DAD para componerlo utilizando correlación SQL:

Utilice correlación SQL cuando componga documentos XML y desee utilizar una sentencia de SQL para definir la tabla y las columnas de las cuales procederán los datos del documento XML.

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con archivos DAD** desde la ventana Área de ejecución. Se visualizará la ventana Especificar DAD.
3. Elija si desea editar un archivo DAD existente o crear uno nuevo.

   Para crear un nuevo archivo DAD:

   a. Deje el campo **Nombre de archivo** en blanco.
   b. Desde el menú **Tipo**, seleccione **Correlación SQL para colecciones XML**.
   c. Pulse **Siguiente** para abrir la ventana Seleccionar validación.

   Para editar un archivo DAD existente:

   a. Escriba el nombre de archivo DAD en el campo **Nombre de archivo** o pulse **...** para localizar un archivo DAD existente.
   b. Verifique que el asistente reconoce el archivo DAD especificado.
      - Si el asistente reconoce el archivo DAD especificado, puede seleccionar **Siguiente** y el campo **Tipo** muestra Correlación SQL para colecciones XML.

- Si el asistente no reconoce el archivo DAD especificado, no se puede seleccionar **Siguiente**. Escriba de nuevo el nombre de archivo DAD o pulse **...** para localizar de nuevo un archivo DAD existente. Corrija los valores del campo de entrada hasta que sea seleccionable **Siguiente**.

c. Pulse **Siguiente** para abrir la ventana Seleccionar validación.

4. En la ventana Seleccionar validación, elija si desea validar los documentos XML con una DTD.
   - Para validar:
     a. Pulse **Validar documentos XML con la DTD**.
     b. En el menú **ID de DTD**, seleccione la DTD que se utilizará para la validación.

     Si no ha importado ninguna DTD al depósito de DTD de la base de datos, no podrá validar documentos XML.
   - Pulse **NO validar documentos XML con la DTD** para continuar sin validar los documentos XML.
5. Pulse **Siguiente** para abrir la ventana Especificar texto.
6. Escriba el nombre del prólogo en el campo **Prólogo** para especificar el prólogo del documento XML que debe componerse.

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

   Si está editando una DAD existente, el prólogo se visualiza automáticamente en el campo **Prólogo**.
7. Escriba el tipo de documento del documento XML en el campo **Tipo de documento** de la ventana Especificar texto, apuntando a la DTD para el documento XML. Por ejemplo:

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

   Si está editando una DAD existente, el tipo de documento se visualiza automáticamente en el campo **Tipo de documento**.
8. Pulse **Siguiente** para abrir la ventana Especificar sentencia de SQL.
9. Escriba una sentencia SELECT válida de SQL en el campo **Sentencia de SQL**. Por ejemplo:

   ```
   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
   ```

Si está editando una DAD existente, la sentencia de SQL se visualiza automáticamente en el campo **Sentencia de SQL**.

10. Pulse **Probar SQL** para verificar la validez de la sentencia de SQL.
    - Si la sentencia de SQL es válida, el campo **Resultados de prueba** muestra los resultados de la prueba.
    - Si la sentencia de SQL no es válida, el campo **Resultados de prueba** muestra un mensaje de error. El mensaje de error le indica que corrija la sentencia SELECT de SQL que ha especificado y que vuelva a intentarlo.
11. Pulse **Siguiente** para abrir la ventana Correlación SQL.
12. Seleccione un nodo de elemento o atributo desde el que desee correlacionar pulsando sobre el mismo en el área izquierda de la ventana Correlación SQL.

    Correlacione los elementos y atributos del documento XML con nodos de elemento y atributo correspondientes a los datos DB2. Estos nodos proporcionan una vía de acceso desde los datos XML a los datos DB2.

    - **Para añadir el nodo raíz:**
      a. Seleccione el icono **Raíz**.
      b. Pulse **Nuevo elemento** para definir un nuevo nodo.
      c. En el recuadro **Detalles**, especifique **Tipo de nodo** como **Elemento**.
      d. Entre el nombre del nodo de nivel superior del campo **Nombre de nodo**.
      e. Pulse **Añadir** para crear el nuevo nodo.

         Ha creado el nodo raíz o elemento, que es el padre de los demás nodos de elemento o atributo de la correlación. Ahora puede añadir a este nodo los elementos y atributos asociados.
    - **Para añadir un nodo de elemento o atributo asociado:**
      a. Pulse sobre un nodo padre del campo de la izquierda para añadir un elemento o atributo asociado.

         Si no ha seleccionado ningún nodo padre, no se podrá seleccionar **Nuevo elemento**.
      b. Pulse **Nuevo elemento**.
      c. En el recuadro **Detalles**, seleccione el tipo de nodo en el menú **Tipo de nodo**.

         El menú **Tipo de nodo** sólo muestra los tipos de nodo que son válidos para el lugar del mapa:

         Elemento
         Representa un elemento XML definido en la DTD asociada con el documento XML. Se utiliza para asociar el elemento XML con una columna de una tabla DB2. Un nodo de elemento puede tener nodos de atributo, nodos de elemento

hijo o nodos de texto. Un nodo de nivel inferior tiene un nodo de texto y un nombre de columna asociados con él en la vista en árbol.

Atributo
Representa un atributo XML definido en la DTD asociada con el documento XML. Se utiliza para asociar el atributo XML con una columna de una tabla DB2. Un nodo de atributo puede tener un nodo de texto y tiene un nombre de columna asociado con él en la vista en árbol.

Texto Especifica el contenido del texto para un nodo de elemento o de atributo que tiene contenido que debe correlacionarse con una tabla relacional. Un nodo de texto tiene un nombre de columna asociado con él en la vista en árbol.

Tabla Especifica el nombre de tabla para un valor de elemento o de atributo que debe correlacionarse con una tabla relacional.

Columna
Especifica el nombre de columna para un valor de elemento o de atributo que debe correlacionarse con una tabla relacional.

Condición
Especifica una condición para la columna.

d. Escriba el nombre del nodo en el campo **Nombre de nodo** del recuadro **Detalles**. Por ejemplo:

```
Pedido
```

e. Si ha especificado **Atributo**, **Elemento** o **Texto** para un elemento de nivel inferior como Tipo de nodo, seleccione una columna en el campo **Columna** del recuadro **Detalles**. Por ejemplo:

```
Nombre_Cliente
```

Restricción: No puede utilizar el asistente de administración para crear nuevas columnas. Si especifica **Columna** como tipo de nodo, sólo puede seleccionar una columna que ya exista en la base de datos DB2.

f. Pulse **Añadir** para añadir el nuevo nodo.

Puede modificar un nodo más adelante pulsando sobre él en el campo de la izquierda y realizando las modificaciones necesarias en él en el recuadro **Detalles**. Pulse **Cambiar** para actualizar el elemento.

También puede añadir elementos o atributos asociados al nodo resaltando el nodo mediante la repetición del proceso de añadir.

g. Continúe editando la correlación SQL o pulse **Siguiente** para abrir la ventana Especificar DAD.

- **Para eliminar un nodo:**
  a. En el área izquierda de la ventana, pulse sobre un nodo.
  b. Pulse **Eliminar**.
  c. Continúe editando la correlación SQL o pulse **Siguiente** para abrir la ventana Especificar DAD.

  Tenga en cuenta que si elimina un nodo de nivel inferior, otro elemento se convertirá en un nodo de nivel inferior y puede que sea necesario definir un nombre de columna para éste.

13. En el campo **Nombre de archivo** de la ventana Especificar DAD, escriba un nombre de archivo de salida para el archivo DAD modificado.
14. Pulse **Finalizar** para volver a la ventana Área de ejecución.

Desde el shell de mandatos de DB2: Utilice notación de correlación SQL cuando componga documentos XML y desee utilizar SQL.

El archivo DAD es un archivo XML que puede crear utilizando un editor de texto cualquiera. Las instrucciones siguientes proceden del apéndice de ejemplos, "Archivos de definición de acceso a documento" en la página 276. Consulte esos ejemplos para obtener una información más completa y de contexto.

1. Abra un editor de texto.
2. Cree la cabecera de DAD:

   ```
   <?xml versión="1.0"?>
   <!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> vía de acceso y nombre de
   archivo de la DTD para la DAD
   ```

3. Inserte los códigos <DAD></DAD>.
4. A continuación del código <DAD>, especifique el ID de DTD que asocia el archivo DAD con la DTD del documento XML.

   ```
   <dtdid>vía\nombre_dtd.dtd> --> vía de acceso y nombre
          de archivo de la DTD de la aplicación
   ```

5. Especifique si desea realizar la validación (es decir, utilizar una DTD para asegurarse de que el documento XML es válido). Por ejemplo:

   ```
   <validation>NO</validation> --> especifique YES o NO
   ```

6. Utilice el elemento <Xcollection> para definir el método de acceso y de almacenamiento como colección XML. Los métodos de acceso y de almacenamiento definen que el contenido del documento XML deriva de los datos almacenados en tablas DB2.

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

7. Especifique una o más sentencias SQL para consultar datos de tablas DB2 o insertarlos en ellas. Vea "Requisitos del esquema de correlación" en la página 65 para obtener directrices. Por ejemplo, especifique una única consulta de SQL como la del ejemplo siguiente:

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

8. Añada la información de prólogo siguiente:

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

   Este texto exacto es necesario.

9. Añada los códigos <doctype></doctype>. Por ejemplo:

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

10. Defina el nodo raíz utilizando los códigos <root_node></root_node>. Dentro del nodo raíz, especifique los elementos y atributos que forman el documento XML.
11. Correlacione los elementos y atributos del documento XML con nodos de elemento y atributo correspondientes a los datos DB2. Estos nodos proporcionan una vía de acceso desde los datos XML a los datos DB2.
    a. Defina un <element_node> (nodo de elemento) para cada elemento del documento XML que se correlacione con una columna de una tabla DB2.

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

       Un element_node (nodo de elemento) puede tener los nodos siguientes.
       - nodo_de_atributo
       - nodo_de_elemento hijo
       - nodo_de_texto

    b. Defina un <attribute_node> (nodo de atributo) para cada atributo del documento XML que se correlacione con una columna de una tabla DB2. Vea las DTD de ejemplo proporcionadas al principio de esta sección para la correlación SQL, así como la DTD para el archivo DAD en el "Apéndice A. DTD para el archivo DAD" en la página 267, que proporciona la sintaxis completa para el archivo DAD.

Por ejemplo, necesita una clave de atributo para un elemento <Order> (Pedido). El valor de la clave se guarda en una columna PART_KEY.

Archivo DAD: En el archivo DAD, cree un nodo de atributo para la clave e indique la tabla donde debe guardarse el valor 1.

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

Documento XML compuesto: El valor de la clave se toma de la columna PART_KEY.

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

12. Cree un <text_node> (nodo de texto) para cada elemento o atributo cuyo contenido derive de una tabla DB2. El nodo de texto tiene un elemento <column> que especifica de qué columna se proporcionará el contenido.

    Por ejemplo, puede tener un elemento XML <Tax> con un valor que se tomará de una columna llamada TAX (IMPUESTO):

    elemento de DAD:

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

    El nombre de columna debe estar en la sentencia SQL al principio del archivo DAD.

    Documento XML compuesto:

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

    El valor `0.02` derivará de la columna TAX.

13. Compruebe que hay un código final </root_node> a continuación del último código </element_node>.
14. Compruebe que hay un código final </Xcollection> a continuación del código </root_node>.
15. Compruebe que hay un código final </DAD> a continuación del código </Xcollection>.

Composición de documentos XML con correlación de nodo_RDB

Utilice correlación de nodo_RDB para componer documentos XML utilizando una estructura similar a XML.

Este método utiliza el <RDB_node> (nodo RDB) para especificar tablas, columnas y condiciones de DB2 para un nodo de elemento o de atributo. <RDB_node> utiliza los elementos siguientes:

- <table>: define la tabla que corresponde al elemento
- <column>: define la columna que contiene el elemento correspondiente
- <condition>: opcionalmente, especifica una condición para la columna

Los elementos asociados que se utilizan en <RDB_node> dependen del contexto del nodo y siguen las normas siguientes:

Si el tipo de nodo es:	**Se utiliza el elemento hijo RDB:**		
	Tabla	**Columna**	**Condición[1]**
Elemento raíz	S	N	S
Atributo	S	S	opcional
Texto	S	S	opcional
(1) Necesario si se utilizan varias tablas			

Utilización del asistente de administración: Para crear una DAD para la composición, utilizando la correlación de nodo_RDB:

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con archivos DAD** desde la ventana Área de ejecución. Se visualizará la ventana Especificar DAD.
3. Elija si desea editar un archivo DAD existente o crear uno nuevo.

   Para editar una DAD existente:

   a. Escriba el nombre de archivo DAD en el campo **Nombre de archivo** o pulse **...** para localizar un archivo DAD existente.
   b. Verifique que el asistente reconoce el archivo DAD especificado.
      - Si el asistente reconoce el archivo DAD especificado, puede seleccionar **Siguiente** y aparece la correlación de RDB_node para colecciones XML en el campo **Tipo**.
      - Si el asistente no reconoce el archivo DAD especificado, no se puede seleccionar **Siguiente**. Escriba de nuevo el nombre del archivo DAD en el campo **Nombre de archivo** o pulse **...** para volver a localizar un archivo DAD existente. Repita estos pasos hasta que pueda seleccionar **Siguiente**.
   c. Pulse **Siguiente** para abrir la ventana Seleccionar validación.

   Para crear una DAD nueva:

   a. Deje el campo **Nombre de archivo** en blanco.
   b. En el menú **Tipo**, seleccione Correlación de nodo_RDB para colecciones XML.

c. Pulse **Siguiente** para abrir la ventana Seleccionar validación.

4. En la ventana Seleccionar validación, elija si desea validar los documentos XML con una DTD.
   - Para validar:
     a. Pulse **Validar documentos XML con la DTD**.
     b. En el menú **ID de DTD**, seleccione la DTD que se utilizará para la validación.

     Si no ha importado ninguna DTD al depósito de DTD de la base de datos, no podrá validar documentos XML.
   - Pulse **NO validar documentos XML con la DTD** para continuar sin validar los documentos XML.
5. Pulse **Siguiente** para abrir la ventana Especificar texto.
6. Escriba el nombre del prólogo en el campo **Prólogo** de la ventana Especificar texto.

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

   Si está editando una DAD existente, el prólogo se visualiza automáticamente en el campo **Prólogo**.
7. Escriba el tipo de documento del documento XML en el campo **Tipo de documento** de la ventana Especificar texto.

   Si está editando una DAD existente, el tipo de documento se visualiza automáticamente en el campo **Tipo de documento**.
8. Pulse **Siguiente** para abrir la ventana Correlación RDB.
9. Seleccione un nodo de elemento o atributo desde el que desee correlacionar pulsando sobre el mismo en el campo situado a la izquierda de la ventana Correlación RDB.

   Correlacione los elementos y atributos del documento XML con nodos de elemento y atributo correspondientes a datos DB2. Estos nodos proporcionan una vía de acceso desde los datos XML a los datos DB2.
10. **Para añadir el nodo raíz:**
    a. Seleccione el icono **Raíz**.
    b. Pulse **Nuevo elemento** para definir un nuevo nodo.
    c. En el recuadro **Detalles**, especifique **Tipo de nodo** como **Elemento**.
    d. Entre el nombre del nodo de nivel superior del campo **Nombre de nodo**.
    e. Pulse **Añadir** para crear el nuevo nodo.

       Ha creado el nodo raíz o elemento, que es el padre de los demás nodos de elemento o atributo de la correlación. El nodo raíz tiene elementos asociados de tabla y una condición de unión.
    f. Añada nodos de tabla para cada tabla que forme parte de la colección.

1) Resalte el nombre del nodo raíz y seleccione **Nuevo elemento**.
2) En el recuadro **Detalles**, especifique **Tipo de nodo**como **Tabla**.
3) Seleccione el nombre de la tabla desde **Nombre de tabla**. La tabla ya debe existir.
4) Pulse **Añadir** para añadir el nodo de tabla.
5) Repita estos pasos para cada tabla.

g. Añada una condición de unión para los nodos de tabla.
1) Resalte el nombre del nodo raíz y seleccione **Nuevo elemento**.
2) En el recuadro **Detalles**, especifique **Tipo de nodo** como **Condición**.
3) En el campo **Condición**, entre la condición de unión utilizando la sintaxis siguiente:

```
nombre_tabla.columna_tabla = nombre_tabla.columna_tabla AND
nombre_tabla.columna_tabla = nombre_tabla.columna_tabla ...
```

4) Pulse **Añadir** para añadir la condición.

11. **Para añadir un nodo de elemento o de atributo:**
a. Pulse sobre un nodo padre del campo de la izquierda para añadir un elemento o atributo asociado.
b. Pulse **Nuevo elemento**.Si no ha seleccionado ningún nodo padre, no se podrá seleccionar **Nuevo elemento**.
c. En el recuadro **Detalles**, seleccione un tipo de nodo en el menú **Tipo de nodo**.

El menú **Tipo de nodo** muestra sólo los tipos de nodo que son válidos para el lugar seleccionado del mapa. **Elemento** o **Atributo**.

d. Especifique un nombre de nodo en el campo **Nombre de nodo**.
e. Pulse **Añadir** para añadir el nuevo nodo.
f. **Para correlacionar el contenido de un nodo de elemento o de atributo con una tabla relacional:**
1) Especifique un nodo de texto.
a) Pulse sobre el nodo padre.
b) Pulse **Nuevo elemento**.
c) En el campo **Tipo de nodo**, seleccione **Texto**.
d) Seleccione **Añadir** para añadir el nodo.
2) Añada un nodo de tabla.
a) Seleccione el nodo de texto que acaba de crear y pulse **Nuevo elemento**.
b) En el campo **Tipo de nodo**, seleccione **Tabla** y especifique un nombre de tabla para el elemento.
c) Pulse **Añadir** para añadir el nodo.

3) Añada un nodo de columna.
   a) Seleccione de nuevo el nodo de texto y pulse **Nuevo elemento**.
   b) En el campo **Tipo de nodo**, seleccione **Columna** y especifique un nombre de columna para el elemento.
   c) Pulse **Añadir** para añadir el nodo.

   Restricción: No puede utilizar el asistente de administración para crear nuevas columnas. Si especifica Columna como tipo de nodo, sólo puede seleccionar una columna que ya exista en su base de datos DB2.
4) Opcionalmente, añada una condición para la columna.
   a) Seleccione de nuevo el nodo de texto y pulse **Nuevo elemento**.
   b) En el campo **Tipo de nodo**, seleccione **Condición** y la condición con la sintaxis:

      operador LIKE|<|>|= *valor*
   c) Pulse **Añadir** para añadir el nodo.

g. Continúe editando la correlación RDB o pulse **Siguiente** para abrir la ventana Especificar DAD.

12. **Para eliminar un nodo:**
    a. En el área izquierda de la ventana, pulse sobre un nodo.
    b. Pulse **Eliminar**.
    c. Continúe editando la correlación de nodo_RDB o pulse **Siguiente** para abrir la ventana Especificar DAD.
13. En el campo **Nombre de archivo** de la ventana Especificar DAD, escriba un nombre de archivo de salida para la DAD modificada.
14. Pulse **Finalizar** para eliminar el nodo y volver a la ventana Área de ejecución.

Desde el shell de mandatos de DB2: El archivo DAD es un archivo XML que puede crear utilizando un editor de texto cualquiera. Las instrucciones siguientes proceden del apéndice de ejemplos, "Archivos de definición de acceso a documento" en la página 276. Consulte esos ejemplos para obtener una información más completa y de contexto.

1. Abra un editor de texto.
2. Cree la cabecera de DAD:

```
<?xml versión="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> vía de acceso y nombre de
archivo de la DTD para la DAD
```

3. Inserte los códigos <DAD></DAD>.
4. A continuación del código <DAD>, especifique el ID de DTD que asocia el archivo DAD con la DTD del documento XML.

```
<dtdid>vía\nombre_dtd.dtd> --> vía de acceso y nombre
de archivo de la DTD para la aplicación
```

5. Especifique si desea realizar la validación (es decir, utilizar una DTD para asegurarse de que el documento XML es válido). Por ejemplo:

   ```
   <validation>NO</validation> --> especifique YES o NO
   ```

6. Utilice el elemento <Xcollection> para definir el método de acceso y de almacenamiento como colección XML. Los métodos de acceso y de almacenamiento definen que los datos XML se guardan en una colección de tablas DB2.

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

7. Añada la información de prólogo siguiente:

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

   Este texto exacto es necesario.

8. Añada los códigos <doctype></doctype>. Por ejemplo:

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

9. Defina el nodo raíz utilizando <root_node>. Dentro del nodo raíz, especifique los elementos y atributos que forman el documento XML.
10. Correlacione los elementos y atributos del documento XML con nodos de elemento y atributo correspondientes a los datos DB2. Estos nodos proporcionan una vía de acceso desde los datos XML a los datos DB2.
    a. Defina un element_node para el nodo raíz. Este element_node contiene:
       - Un RDB_node que especifica table_nodes con una condición de unión para especificar la colección
       - Elementos asociados
       - Atributos

       Para especificar los nodos de tabla y la condición:

       1) Cree un elemento RDB_node: Por ejemplo:

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

       2) Defina un <table_node> para cada tabla que contenga datos que deban incluirse en el documento XML. Por ejemplo, si tiene tres tablas, ORDER_TAB, PART_TAB y SHIP_TAB, con datos de columnas que deban estar en el documento, cree un nodo de tabla para cada una de ellas. Por ejemplo:

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

3) Opcionalmente, especifique una columna de clave para cada tabla cuando tenga planeado habilitar esta colección. El atributo de clave no suele ser necesario para la composición; no obstante, cuando habilite una colección, el archivo DAD utilizado deberá dar soporte a la composición y a la descomposición. Por ejemplo:

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

4) Defina una condición de unión para las tablas de la colección. La sintaxis es la siguiente

```
expresión = expresión AND
expresión = expresión
```

Por ejemplo:

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

b. Defina un código <element_node> para cada elemento del documento XML que se correlacione con una columna de una tabla DB2. Por ejemplo:

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

Un nodo de elemento puede tener uno de los tipos de elementos siguientes:

- <text_node>: para especificar que el elemento tiene contenido para una tabla DB2; el elemento no tiene elementos asociados
- <attribute_node>: para especificar un atributo. Los nodos de atributo se definen en el paso siguiente.

text_node contiene un <RDB_node> para correlacionar el contenido con un nombre de columna y tabla DB2.

Los RDB_nodes se utilizan para elementos de nivel inferior que tienen contenido para correlacionar con una tabla DB2. Un RDB_node tiene los siguientes elementos asociados.

- <table>: define la tabla que corresponde al elemento

- <column>: define la columna que contiene el elemento correspondiente y especifica el tipo de columna con el atributo de tipo
- <condition>: opcionalmente, especifica una condición para la columna

Por ejemplo, puede tener el elemento XML <Tax> que se correlaciona con una columna llamada TAX (impuesto):

Documento XML:

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

En este caso, desea que la columna TAX contenga el valor `0,02`.

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

En este ejemplo, <RDB_node> especifica que el valor del elemento <Tax> es un valor de texto; los datos se almacenan en la columna TAX de la tabla PART_TAB.Vea los archivos DAD de ejemplo proporcionados en "Archivos de definición de acceso a documento" en la página 276 para la correlación de nodo_RDB, así como la DTD para el archivo DAD, en el "Apéndice A. DTD para el archivo DAD" en la página 267, que proporciona la sintaxis completa del archivo DAD.

c. Opcionalmente, añada un atributo de tipo a cada elemento <column> cuando tenga planeado habilitar esta colección. El atributo de tipo no suele ser necesario para la composición; no obstante, cuando habilite una colección, el archivo DAD utilizado deberá dar soporte a la composición y a la descomposición. Por ejemplo:

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

d. Defina un <attribute_node> (nodo de atributo) para cada atributo del documento XML que se correlacione con una columna de una tabla DB2. Por ejemplo:

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

attribute_node tiene un <RDB_node> para correlacionar el valor de atributo con una columna y tabla DB2. Un <RDB_node> tiene los siguientes elementos asociados.

- <table>: define la tabla que corresponde al elemento

- <column>: define la columna que contiene el elemento correspondiente
- <condition>: opcionalmente, especifica una condición para la columna

Por ejemplo, puede que desee tener una `clave` de atributo para un elemento <Order> (Pedido). El valor de la clave se debe guardar en una columna llamada PART_KEY. En el archivo DAD, cree un <attribute_node> (nodo de atributo) para la clave e indique la tabla en la que debe guardarse el valor.

Archivo DAD

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

Documento XML compuesto:

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

11. Compruebe que hay un código final </root_node> a continuación del último código </element_node>.
12. Compruebe que hay un código final </Xcollection> a continuación del código </root_node>.
13. Compruebe que hay un código final </DAD> a continuación del código </Xcollection>.

Especificación de una hoja de estilo para el documento XML

Al componer documentos, el XML Extender también da soporte a las instrucciones de proceso para hojas de estilo, utilizando el elemento <stylesheet>. Las instrucciones de proceso han de estar dentro del elemento raíz <Xcollection>, situado con los elementos <doctype> y <prolog> definidos para la estructura de documentos XML. Por ejemplo:

```
<?xml versión="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>
```

Descomposición de documentos XML con correlación de nodo_RDB

Utilice la correlación de nodo_RDB para descomponer documentos XML. Este método utiliza el <RDB_node> (nodo RDB) para especificar tablas, columnas y condiciones de DB2 para un nodo de elemento o de atributo. <RDB_node> utiliza los elementos siguientes:

- <table>: define la tabla que corresponde al elemento
- <column>: define la columna que contiene el elemento correspondiente
- <condition>: opcionalmente, especifica una condición para la columna

Los elementos asociados que se utilizan en <RDB_node> dependen del contexto del nodo y siguen las normas siguientes:

Si el tipo de nodo es:	**Se utiliza el elemento hijo RDB:**		
	Tabla	**Columna**	**Condición[1]**
Elemento raíz	S	N	S
Atributo	S	S	opcional
Texto	S	S	opcional
(1) Necesario si se utilizan varias tablas			

Utilización del asistente de administración: Para crear una DAD para la descomposición:

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con archivos DAD** desde la ventana Área de ejecución. Se visualizará la ventana Especificar DAD.
3. Elija si desea editar un archivo DAD existente o crear uno nuevo.

   Para editar una DAD existente:

   a. Escriba el nombre de archivo DAD en el campo **Nombre de archivo** o pulse **...** para localizar un archivo DAD existente.
   b. Verifique que el asistente reconoce el archivo DAD especificado.
      - Si el asistente reconoce el archivo DAD especificado, puede seleccionar **Siguiente** y el campo **Tipo** muestra Correlación de nodo_RDB para colecciones XML.
      - Si el asistente no reconoce el archivo DAD especificado, no se puede seleccionar **Siguiente**. Escriba de nuevo el nombre del archivo DAD en el campo **Nombre de archivo** o pulse **...** para

volver a localizar un archivo DAD existente. Repita estos pasos hasta que pueda seleccionar **Siguiente**.

c. Pulse **Siguiente** para abrir la ventana Seleccionar validación.

Para crear una DAD nueva:

a. Deje el campo **Nombre de archivo** en blanco.
b. En el menú **Tipo**, seleccione Correlación de nodo_RDB para colecciones XML.
c. Pulse **Siguiente** para abrir la ventana Seleccionar validación.

4. En la ventana Seleccionar validación, elija si desea validar los documentos XML con una DTD.
   - Para validar:
     a. Pulse **Validar documentos XML con la DTD**.
     b. En el menú **ID de DTD**, seleccione la DTD que se utilizará para la validación.

     Si no ha importado ninguna DTD al depósito de DTD de la base de datos, no podrá validar documentos XML.
   - Pulse **NO validar documentos XML con la DTD** para continuar sin validar los documentos XML.
5. Pulse **Siguiente** para abrir la ventana Especificar texto.
6. Si sólo está descomponiendo un documento XML, ignore el campo **Prólogo**. Si utiliza el archivo DAD para la composición y la descomposición, escriba el nombre del prólogo en el campo **Prólogo** de la ventana Especificar texto. El prólogo no es necesario si descompone documentos XML en datos DB2.

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

   Si está editando una DAD existente, el prólogo se visualiza automáticamente en el campo **Prólogo**.
7. Si sólo está descomponiendo un documento XML, ignore el campo **Tipo de documento**. Si utiliza el archivo DAD para la composición y la descomposición, escriba el tipo de documento del documento XML en el campo **Tipo de documento**.

   Si está editando una DAD existente, el tipo de documento se visualiza automáticamente en el campo **Tipo de documento**.
8. Pulse **Siguiente** para abrir la ventana Correlación RDB.
9. Seleccione un nodo de elemento o atributo desde el que desee correlacionar pulsando sobre el mismo en el campo situado a la izquierda de la ventana Correlación RDB.

Correlacione los elementos y atributos del documento XML con nodos de elemento y atributo correspondientes a datos DB2. Estos nodos proporcionan una vía de acceso desde los datos XML a los datos DB2.

10. **Para añadir el nodo raíz:**
    a. Seleccione el icono **Raíz**.
    b. Pulse **Nuevo elemento** para definir un nuevo nodo.
    c. En el recuadro **Detalles**, especifique **Tipo de nodo** como **Elemento**.
    d. Entre el nombre del nodo de nivel superior del campo **Nombre de nodo**.
    e. Pulse **Añadir** para crear el nuevo nodo.

       Ha creado el nodo o elemento raíz, que es el padre de los demás nodos de elemento o atributo de la correlación. El nodo raíz tiene elementos asociados de tabla y una condición de unión.
    f. Añada nodos de tabla para cada tabla que forme parte de la colección.
       1) Resalte el nombre del nodo raíz y seleccione **Nuevo elemento**.
       2) En el recuadro **Detalles**, especifique **Tipo de nodo**como **Tabla**.
       3) Seleccione el nombre de la tabla desde **Nombre de tabla**. La tabla ya debe existir.
       4) Especifique una columna de clave para la tabla en el campo **Clave de tabla**.
       5) Pulse **Añadir** para añadir el nodo de tabla.
       6) Repita estos pasos para cada tabla.
    g. Añada una condición de unión para los nodos de tabla.
       1) Resalte el nombre del nodo raíz y seleccione **Nuevo elemento**.
       2) En el recuadro **Detalles**, especifique **Tipo de nodo** como **Condición**.
       3) En el campo **Condición**, entre la condición de unión utilizando la sintaxis siguiente:

          ```
          nombre_tabla.columna_tabla = nombre_tabla.columna_tabla AND
          nombre_tabla.columna_tabla = nombre_tabla.columna_tabla ...
          ```
       4) Pulse **Añadir** para añadir la condición.

    Ahora puede añadir a este nodo los elementos y atributos asociados.

11. **Para añadir un nodo de elemento o de atributo:**
    a. Pulse sobre un nodo padre del campo de la izquierda para añadir un elemento o atributo asociado.

       Si no selecciona un nodo padre, no podrá seleccionar **Nuevo**.
    b. Pulse **Nuevo elemento**.
    c. En el recuadro **Detalles**, seleccione un tipo de nodo en el menú **Tipo de nodo**.

El menú **Tipo de nodo** muestra sólo los tipos de nodo que son válidos para el lugar seleccionado del mapa. **Elemento** o **Atributo**.

d. Especifique un nombre de nodo en el campo **Nombre de nodo**.

e. Pulse **Añadir** para añadir el nuevo nodo.

f. **Para correlacionar el contenido de un nodo de elemento o de atributo con una tabla relacional:**

   1) Especifique un nodo de texto.
      a) Pulse sobre el nodo padre.
      b) Pulse **Nuevo elemento**.
      c) En el campo **Tipo de nodo**, seleccione **Texto**.
      d) Seleccione **Añadir** para añadir el nodo.
   2) Añada un nodo de tabla.
      a) Seleccione el nodo de texto que acaba de crear y pulse **Nuevo elemento**.
      b) En el campo **Tipo de nodo**, seleccione **Tabla** y especifique un nombre de tabla para el elemento.
      c) Pulse **Añadir** para añadir el nodo.
   3) Añada un nodo de columna.
      a) Seleccione de nuevo el nodo de texto y pulse **Nuevo elemento**.
      b) En el campo **Tipo de nodo**, seleccione **Columna** y especifique un nombre de columna para el elemento.
      c) Especifique un tipo de base de datos para la columna en el campo **Tipo**, para especificar el tipo de columna necesario para guardar los datos no codificados.
      d) Pulse **Añadir** para añadir el nodo.

      Restricción: No puede utilizar el asistente de administración para crear nuevas columnas. Si especifica Columna como tipo de nodo, sólo puede seleccionar una columna que ya exista en su base de datos DB2.
   4) Opcionalmente, añada una condición para la columna.
      a) Seleccione de nuevo el nodo de texto y pulse **Nuevo elemento**.
      b) En el campo **Tipo de nodo**, seleccione **Condición** y la condición con la sintaxis:

         operador LIKE|<|>|= *valor*
      c) Pulse **Añadir** para añadir el nodo.

   Puede modificar estos nodos seleccionando el nodo, cambiando los campos en el recuadro **Detalles**, y pulsando **Cambiar**.

g. Continúe editando la correlación RDB o pulse **Siguiente** para abrir la ventana Especificar DAD.

12. **Para eliminar un nodo:**
    a. En el área izquierda de la ventana, pulse sobre un nodo.
    b. Pulse **Eliminar**.
    c. Continúe editando la correlación de nodo_RDB o pulse **Siguiente** para abrir la ventana Especificar DAD.
13. En el campo **Nombre de archivo** de la ventana Especificar DAD, escriba un nombre de archivo de salida para la DAD modificada.
14. Pulse **Finalizar** para eliminar el nodo y volver a la ventana Área de ejecución.

Desde el shell de mandatos de DB2: El archivo DAD es un archivo XML que puede crear utilizando un editor de texto cualquiera. Las instrucciones siguientes proceden del apéndice de ejemplos, "Archivos de definición de acceso a documento" en la página 276. Consulte esos ejemplos para obtener una información más completa y de contexto.

1. Abra un editor de texto.
2. Cree la cabecera de DAD:

   ```
   <?xml versión="1.0"?>
   <!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> vía de acceso y nombre de
   archivo de la DTD para la DAD
   ```

3. Inserte los códigos <DAD></DAD>.
4. A continuación del código <DAD>, especifique el ID de DTD que asocia el archivo DAD con la DTD del documento XML.

   ```
   <dtdid>vía\nombre_dtd.dtd> --> vía de acceso y nombre
   de archivo de la DTD para la aplicación
   ```

5. Especifique si desea realizar la validación (es decir, utilizar una DTD para asegurarse de que el documento XML es válido). Por ejemplo:

   ```
   <validation>NO</validation> --> especifique YES o NO
   ```

6. Utilice el elemento <Xcollection> para definir el método de acceso y de almacenamiento como colección XML. Los métodos de acceso y de almacenamiento definen que los datos XML se guardan en una colección de tablas DB2.

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

7. Añada la información de prólogo siguiente:

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

   Este texto exacto es necesario.
8. Añada los códigos <doctype></doctype>. Por ejemplo:

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

9. Defina el nodo raíz (root_node) utilizando los códigos <root_node></root_node>. Dentro del nodo raíz, especifique los elementos y atributos que forman el documento XML.
10. A continuación del código <root_node>, correlacione los elementos y atributos del documento XML con nodos de elemento y atributo correspondientes a los datos DB2. Estos nodos proporcionan una vía de acceso desde los datos XML a los datos DB2.
    a. Defina un element_node raíz, de nivel superior. Este element_node contiene:
       - Nodos de tabla con una condición de unión para especificar la colección.
       - Elementos asociados
       - Atributos

       Para especificar los nodos de tabla y la condición:

       1) Cree un elemento RDB_node: Por ejemplo:

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

       2) Defina un <table_node> para cada tabla que contenga datos que deban incluirse en el documento XML. Por ejemplo, si tiene tres tablas, ORDER_TAB, PART_TAB y SHIP_TAB, con datos de columnas que deban estar en el documento, cree un nodo de tabla para cada una de ellas. Por ejemplo:

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

       3) Defina una condición de unión para las tablas de la colección. La sintaxis es la siguiente

          ```
          expresión = expresión AND
          expresión = expresión ...
          ```

          Por ejemplo:

          ```
          <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) Especifique una clave primaria para cada tabla. La clave primaria consta de una o varias columnas; en este segundo caso se denomina clave compuesta. Para especificar la clave primaria, añada una clave de atributo al elemento de tabla del nodo RDB.

El ejemplo siguiente define una clave primaria para cada tabla contenida en el nodo RDB del nodo de elemento raíz ″Order″ (pedido):

```
<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>
```

La información especificada para la descomposición de un documento XML no se tiene en cuenta al componer el documento.

El atributo de clave es necesario para la descomposición y cuando habilite una colección, porque el archivo DAD utilizado debe dar soporte a la composición y a la descomposición.

b. Defina un código <element_node> para cada elemento del documento XML que se correlacione con una columna de una tabla DB2. Por ejemplo:

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

Un nodo de elemento puede tener uno de los tipos de elementos siguientes:

- <text_node>: para especificar que el elemento tiene contenido para una tabla DB2; en este caso no tiene elementos asociados.
- <attribute_node>: para especificar un atributo; los nodos de atributo se definen en el paso siguiente
- elementos asociados

text_node contiene un RDB_node para correlacionar el contenido con un nombre de columna y tabla DB2.

Los RDB_nodes se utilizan para elementos de nivel inferior que tienen contenido para correlacionar con una tabla DB2. Un RDB_node tiene los siguientes elementos asociados.

- <table>: define la tabla que corresponde al elemento
- <column>: define la columna que contiene el elemento correspondiente
- <condition>: opcionalmente, especifica una condición para la columna

Por ejemplo, puede tener un elemento XML <Tax> para el que desee guardar el contenido no codificado en una columna llamada TAX:

Documento XML:

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

En este caso, desea que el valor `0.02` se guarde en la columna TAX.

En el archivo DAD, especifica un <RDB_node> para correlacionar el elemento XML con la tabla y columna DB2.

Archivo DAD:

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

<RDB_node> especifica que el valor del elemento <Tax> es un valor de texto, los datos se almacenan en la tabla PART_TAB, en la columna TAX.

c. Defina un <attribute_node> (nodo de atributo) para cada atributo del documento XML que se correlacione con una columna de una tabla DB2. Por ejemplo:

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

attribute_node tiene un RDB_node para correlacionar el valor de atributo con una tabla y columna DB2. Un RDB_node tiene los siguientes elementos asociados.

- <table>: define la tabla que corresponde al elemento
- <column>: define la columna que contiene el elemento correspondiente
- <condition>: opcionalmente, especifica una condición para la columna

Por ejemplo, puede tener una clave de atributo para un elemento <Order> (pedido). El valor de la clave se debe guardar en una columna llamada PART_KEY.

Documento XML:

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

En el archivo DAD, cree un attribute_node para la clave e indique la tabla donde se debe guardar el valor 1.

Archivo DAD:

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

11. Especifique el tipo de columna del nodo RDB para cada nodo de atributo y nodo de texto. Esto asegura el tipo de datos correcto para cada columna en la que se guardarán los datos no codificados. Para especificar los tipos de columna, añada el tipo de atributo al elemento de columna. El ejemplo siguiente define el tipo de columna INTEGER:

```
<attribute_node name="key">
    <RDB_node>
        <table name="order_tab"/>
        <column name="order_key" type="integer"/>
    </RDB_node>
</attribute_node>
```

12. Compruebe que hay un código final </root_node> a continuación del último código </element_node>.
13. Compruebe que hay un código final </Xcollection> a continuación del código </root_node>.
14. Compruebe que hay un código final </DAD> a continuación del código </Xcollection>.

Habilitación de colecciones XML

Cuando se habilita una colección XML, se analiza el archivo DAD para identificar las tablas y columnas relacionadas con el documento XML, y se registra información de control en la tabla XML_USAGE. Habilitar una colección XML es opcional en estos casos:

- Descomponer un documento XML y almacenar los datos en nuevas tablas DB2
- Componer un documento XML a partir de datos existentes en varias tablas DB2

Si se utiliza el mismo archivo DAD para la composición y descomposición, puede habilitar la colección para la composición y la descomposición.

Puede habilitar una colección XML mediante el asistente de administración del XML Extender, o utilizando el mandato **dxxadm** con la opción enable_collection, o bien puede utilizar el procedimiento almacenado dxxEnableCollection() del XML Extender.

Utilización del asistente de administración

Siga los pasos siguientes para habilitar una colección XML.

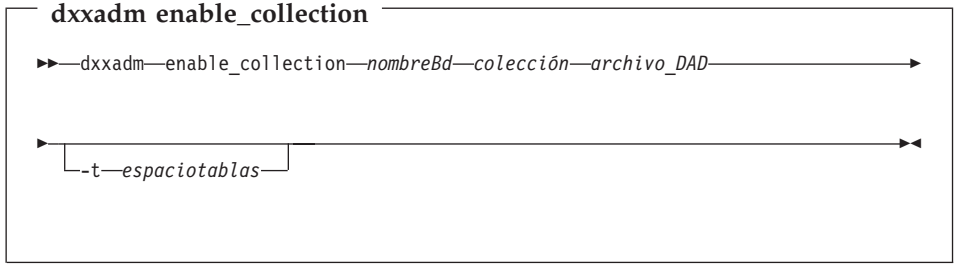
1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con colecciones XML** desde la ventana Área de ejecución. Se abrirá la ventana Seleccionar tarea.
3. Pulse **Habilitar colección** y, a continuación, **Siguiente**. Se abrirá la ventana Habilitar colección.
4. Seleccione el nombre de la colección que desee habilitar en el campo **Nombre de colección** del menú desplegable.
5. Escriba el nombre de archivo DAD en el campo **Nombre de archivo DAD** o pulse **...** para localizar un archivo DAD existente.
6. Opcionalmente, escriba en el campo **Espacio_tablas** el nombre de un espacio de tablas creado previamente.

   El espacio de tablas contendrá las nuevas tablas DB2 generadas para la descomposición.
7. Pulse **Finalizar** para habilitar la colección y volver a la ventana Área de ejecución.
   - Si la colección se habilita satisfactoriamente, se visualiza el mensaje `Colección habilitada satisfactoriamente`.
   - Si la colección no se habilita satisfactoriamente, se visualiza un mensaje de error. Repita los pasos anteriores hasta que la colección se habilite correctamente.

Desde el shell de mandatos de DB2

Para habilitar una colección XML, entre el mandato **dxxadm**:

Sintaxis:



Parámetros:

nombreBd
Es el nombre de la base de datos.

colección
: Es el nombre de la colección XML. Este valor se utiliza como parámetro de los procedimientos almacenados para colecciones XML.

archivo_DAD
: Es el nombre del archivo donde reside la definición de acceso a documento (DAD).

espaciotablas
: Es un espacio de tablas existente que contiene las nuevas tablas DB2 que se crearon para la descomposición. Si no se especifica este parámetro, se utiliza el espacio de tablas predefinido.

Ejemplo: El ejemplo siguiente habilita una colección llamada sales_ord en la base de datos SALES_DB, utilizando el shell de mandatos de DB2. El archivo DAD utiliza la correlación SQL y se puede encontrar en "Archivo DAD: colección XML - correlación SQL" en la página 278.

```
dxxadm enable_collection SALES_DB sales_ord getstart.dad
```

Una vez habilitada la colección XML, puede componer o descomponer documentos XML utilizando los procedimientos almacenados del XML Extender.

Inhabilitación de colecciones XML

Cuando se inhabilita una colección XML se elimina el registro en la tabla XML_USAGE que identifica tablas y columnas como parte de una colección. No elimina ninguna tabla de datos. Puede inhabilitar una colección cuando desee actualizar la DAD y necesite volver a habilitar una colección o eliminar una colección.

Puede inhabilitar una colección XML mediante el asistente de administración del XML Extender, o utilizando el mandato **dxxadm** con la opción disable_collection, o bien puede utilizar el procedimiento almacenado dxxDisableCollection() del XML Extender.

Utilización del asistente de administración

Siga los pasos siguientes para inhabilitar una colección XML.

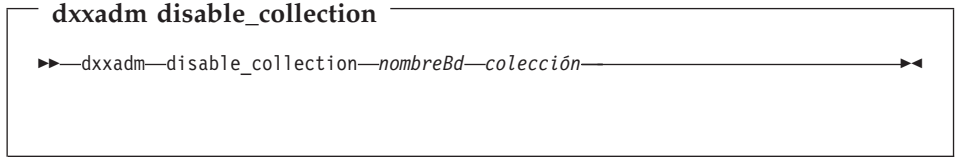
1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Trabajar con colecciones XML** desde la ventana Área de ejecución para ver las tareas relacionadas con colecciones del XML Extender. Se abrirá la ventana Seleccionar tarea.
3. Pulse **Inhabilitar una colección XML** y, a continuación, **Siguiente** para inhabilitar una colección XML. Se abrirá la ventana Inhabilitar colección.
4. En el campo **Nombre de colección**, escriba el nombre de la colección que desea inhabilitar.

5. Pulse **Finalizar** para inhabilitar la colección y volver a la ventana Área de ejecución.
   - Si la colección se inhabilita satisfactoriamente, se visualiza el mensaje `Colección inhabilitada satisfactoriamente`.
   - Si la colección no se inhabilita satisfactoriamente, se visualiza una ventana de error. Repita los pasos anteriores hasta que la colección se inhabilite correctamente.

Desde el shell de mandatos de DB2

Para inhabilitar una colección XML, entre el mandato **dxxadm**:

Sintaxis:



Parámetros:

nombreBd
Es el nombre de la base de datos.

colección
Es el nombre de la colección XML. Este valor se utiliza como parámetro de los procedimientos almacenados para colecciones XML.

Ejemplo:

```
dxxadm disable_collection SALES_DB sales_ord
```

Inhabilitación de una base de datos para XML

La base de datos se inhabilita cuando se quiere limpiar el entorno del XML Extender y eliminar procedimientos almacenados, UDT, UDF y tablas de soporte de administración del XML Extender. El XML Extender inhabilita la base de datos con la que está conectando, utilizando la instancia actual.

Cuando inhabilita una base de datos para XML, el XML Extender emprende las acciones siguientes para la base de datos:

- Suprime todos los tipos definidos por el usuario (UDT) y todas las funciones definidas por el usuario (UDF)
- Suprime las tablas de control con los metadatos para el XML Extender
- Suprime el esquema db2xml.

Antes de comenzar

Inhabilite cualquier columna o colección XML de la base de datos que deba inhabilitarse.

Utilización del asistente de administración

Realice los pasos siguientes para inhabilitar una base de datos para datos XML:

1. Ponga a punto y arranque el asistente de administración. Vea "Arranque del asistente de administración" en la página 71 para obtener detalles.
2. Pulse **Inhabilitar base de datos** en la ventana Área de ejecución para inhabilitar la base de datos actual.

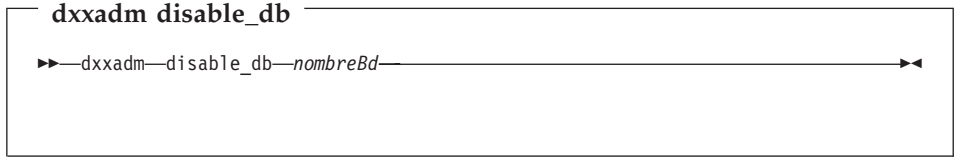
   Si no hay ninguna base de datos habilitada actualmente, sólo se puede seleccionar **Habilitar base de datos**.

   Cuando la base de datos está inhabilitada, se le devuelve a la ventana Área de ejecución.

Desde el shell de mandatos de DB2

Entre **dxxadm** en la línea de mandatos, especificando la base de datos que se debe inhabilitar.

Sintaxis:



Parámetros:

nombreBd
Es el nombre de la base de datos que se debe inhabilitar.

Ejemplo: inhabilita una base de datos existente, llamada SALES_DB.

```
dxxadm disable_db SALES_DB
```

Parte 3. Programación

Esta parte describe técnicas de programación para gestionar datos XML.

Capítulo 5. Gestión de datos de columna XML

Cuando utiliza columnas XML, almacena un documento XML completo en forma de datos de columna. Este método de acceso y almacenamiento le permite conservar el documento XML inalterado, y al mismo tiempo puede indexar y hacer búsquedas en el documento, recuperar datos de él y actualizarlo. Una columna XML contiene documentos XML en su formato nativo en DB2 como datos de columna. Una vez habilitada una base de datos para XML, puede utilizar los siguientes UDT (tipos definidos por el usuario):

XMLCLOB
: El contenido del documento XML se almacena como gran objeto de caracteres (CLOB) en DB2

XMLVARCHAR
: El contenido del documento XML se almacena como datos de tipo VARCHAR en DB2

XMLFile
: El documento XML se almacena en un archivo de un sistema de archivos local

Puede crear o modificar tablas de aplicación utilizando los UDT de XML como tipos de datos de columna. Estas tablas se denominan tablas XML. Consulte "Creación o alteración de una tabla XML" en la página 82 para obtener información sobre cómo crear o modificar una tabla para XML.

Una vez habilitada una columna para XML, puede comenzar a gestionar el contenido de la columna XML. Después de crear la columna XML, puede realizar las tareas de gestión siguientes:

- Almacenar documentos XML en DB2
- Recuperar datos o documentos XML desde DB2
- Actualizar documentos XML
- Suprimir datos o documentos XML

Para ejecutar estas tareas, puede utilizar dos métodos:

- *Funciones predefinidas de conversión de datos*, que convierten el tipo base SQL en el UDT de XML
- Funciones definidas por el usuario (UDF), que proporciona el XML Extender

El presente manual describe ambos métodos para cada tarea.

Nombres utilizados en los UDT y en las UDF

El nombre completo de una función DB2 es *nombre_esquema.nombre_función*, donde *nombre_esquema* es un identificador que permite crear agrupaciones lógicas de objetos SQL. El nombre de esquema para las UDF del XML Extender es db2xml. El nombre de esquema db2xml también es el calificador de los UDT del XML Extender. En este manual sólo se menciona el nombre de función.

La vía de función es una lista ordenada de nombres de esquema. DB2 utiliza esta lista para resolver las referencias a funciones y a los UDT. Puede especificar la vía de función mediante la sentencia SET CURRENT FUNCTION PATH de SQL. Esta sentencia define la vía de función en el registro especial CURRENT FUNCTION PATH.

Desde el punto de vista del XML Extender, es aconsejable añadir el esquema db2xml a la vía de función. Esto le permite entrar los nombres de las UDF y de los UDT sin tener que añadirles el prefijo `db2xml`. El ejemplo siguiente muestra cómo añadir el esquema db2xml a la vía de función:

```
SET CURRENT FUNCTION PATH = db2xml, CURRENT FUNCTION PATH
```

Importante: No añada `db2xml` como primer esquema de la vía de función si inicia la sesión como db2xml; db2xml se establece automáticamente como primer esquema cuando inicia la sesión como db2xml. Esto generaría una condición de error, pues la vía de función comenzaría con dos esquemas db2xml.

Almacenamiento de datos

Utilizando el XML Extender, puede insertar documentos XML inalterados en una columna XML. Si define tablas auxiliares, el XML Extender actualiza automáticamente estas tablas. Cuando almacena directamente un documento XML, el XML Extender almacena el tipo base como tipo XML.

Visión general de las tareas:

1. Asegúrese de que ha creado o actualizado el archivo DAD.
2. Determine el tipo de datos que utilizará para almacenar el documento.
3. Elija un método para almacenar los datos en la tabla DB2 (funciones de conversión de datos o funciones UDF).
4. Mediante una sentencia INSERT de SQL, especifique la tabla y columna XML donde residirá el documento XML.

El XML Extender proporciona dos métodos para almacenar documentos XML: funciones predefinidas de conversión de datos y funciones UDF de almacenamiento. La Tabla 8 en la página 123 muestra cuándo utilizar cada

método.

Tabla 8. Funciones de almacenamiento del XML Extender

Tipo base	Almacenar en DB2 como...		
	XMLVARCHAR	**XMLCLOB**	**XMLFILE**
VARCHAR	XMLVARCHAR()	N/A	XMLFileFromVarchar()
CLOB	N/A	XMLCLOB()	XMLFileFromCLOB()
FILE	XMLVarcharFromFile()	XMLCLOBFromFile()	XMLFILE

Uso de una función predefinida de conversión de datos

Para cada UDT, existe una *función predefinida de conversión de datos* para convertir el tipo base SQL al UDT. Para insertar datos, puede utilizar funciones de conversión de datos, proporcionadas por el XML Extender, en una cláusula VALUES. La Tabla 9 muestra las funciones de conversión de datos proporcionadas:

Tabla 9. Funciones predefinidas de conversión de datos del XML Extender

Conversión de datos utilizada en cláusula SELECT	Tipo devuelto	Descripción
XMLVARCHAR(VARCHAR)	XMLVARCHAR	Datos de entrada procedentes de memoria intermedia de VARCHAR
XMLCLOB(CLOB)	XMLCLOB	Datos de entrada procedentes de memoria intermedia de CLOB o de un localizador de CLOB
XMLFILE(VARCHAR)	XMLFILE	Sólo almacenar nombre de archivo

Ejemplo: La sentencia siguiente inserta un tipo convertido VARCHAR en el tipo XMLVARCHAR:

```
INSERT INTO sales_tab
VALUES('123456', 'Sriram Srinivasan', db2xml.XMLVarchar(:xml_buff))
```

Uso de una UDF de almacenamiento:

Para cada UDT de XML Extender, existe una UDF de almacenamiento para importar datos a DB2 desde una fuente distinta del tipo base de DB2. Por ejemplo, si desea importar un documento de archivos XML a DB2 en forma de XMLCLOB, puede utilizar la función XMLCLOBFromFile().

La Tabla 10 en la página 124 muestra las funciones de almacenamiento proporcionadas por el XML Extender.

Tabla 10. Las UDF de almacenamiento del XML Extender

UDF de almacenamiento	Tipo devuelto	Descripción
XMLVarcharFromFile()	XMLVARCHAR	Lee un documento XML en un archivo de servidor y devuelve un valor de tipo XMLVARCHAR.
XMLCLOBFromFile()	XMLCLOB	Lee un documento XML en un archivo de servidor y devuelve un valor de tipo XMLCLOB.
XMLFileFromVarchar()	XMLFILE	Lee un documento XML de tipo VARCHAR contenido en la memoria, lo escribe en un archivo externo y devuelve un valor de tipo XMLFILE, que es el nombre de archivo.
XMLFileFromCLOB()	XMLFILE	Lee un documento XML en forma de CLOB o de localizador de CLOB, contenido en la memoria, lo escribe en un archivo externo y devuelve un valor de tipo XMLFILE, que es el nombre de archivo.

Ejemplo: La sentencia siguiente almacena un registro en una tabla XML utilizando la función XMLCLOBFromFile() como XMLCLOB.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
    VALUES( '1234', 'Sriram Srinivasan,
      XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

El ejemplo anterior importa el objeto XML desde el archivo llamado `c:\dxx\samples\cmd\getstart.xml` a la columna ORDER de la tabla SALES_TAB.

Recuperación de datos

Mediante el XML Extender, puede recuperar un documento completo o el contenido de elementos y atributos. Cuando recupera directamente una columna XML, el XML Extender devuelve el UDT como tipo de columna. Para obtener detalles sobre la recuperación de datos, vea las secciones siguientes:

• "Recuperación de un documento completo" en la página 125

- "Recuperación del contenido de elementos y de valores de atributos" en la página 127

El XML Extender proporciona dos métodos para recuperar datos: funciones predefinidas de conversión de datos y la UDF sobrecargada Content(). La Tabla 11 muestra cuándo utilizar cada método.

Tabla 11. Funciones de recuperación del XML Extender

Tipo XML	Recuperar de DB2 como...		
	VARCHAR	**CLOB**	**FILE**
XMLVARCHAR	VARCHAR	N/A	Content()
XMLCLOB	N/A	XMLCLOB	Content()
XMLFILE	N/A	Content()	FILE

Recuperación de un documento completo

Visión general de las tareas:

1. Asegúrese de que ha almacenado el documento en una tabla XML y determine qué datos desea recuperar.
2. Elija un método para recuperar los datos de la tabla DB2 (funciones de conversión de datos o funciones UDF).
3. Si utiliza la UDF Content(), determine qué tipo de datos se recupera y qué tipo de datos se debe exportar.
4. Mediante una consulta SQL, especifique la tabla y columna XML donde reside el documento XML que se debe recuperar.

El XML Extender proporciona dos métodos para recuperar datos:

Uso de una función predefinida de conversión de datos
Utilice la función predefinida de conversión de datos, proporcionada por DB2 para los UDT, para convertir un UDT XML en un tipo base SQL, y luego trabajar sobre él. Para recuperar datos, puede utilizar funciones de conversión de datos, proporcionadas por el XML Extender, en una sentencia SELECT. La Tabla 12 muestra la funciones de conversión de datos proporcionadas:

Tabla 12. Funciones predefinidas de conversión de datos del XML Extender

Conversión de datos utilizada en cláusula SELECT	Tipo devuelto	Descripción
varchar(XMLVARCHAR)	VARCHAR	Documento XML en VARCHAR
clob(XMLCLOB)	CLOB	Documento XML en CLOB

Tabla 12. Funciones predefinidas de conversión de datos del XML Extender (continuación)

Conversión de datos utilizada en cláusula SELECT	Tipo devuelto	Descripción
varchar(XMLFile)	VARCHAR	Nombre de archivo XML en VARCHAR

Ejemplo: El ejemplo siguiente recupera los datos XMLVARCHAR y los almacena utilizando el tipo de datos VARCHAR:

```
EXEC SQL SELECT db2xml.varchar(order) from sales_tab
```

Uso de la UDF sobrecargada Content()

Utilice la UDF Content() para recuperar un documento del almacenamiento externo y colocarlo en la memoria, o para exportar el documento desde el almacenamiento interno al *archivo externo* del servidor DB2.

Por ejemplo, puede almacenar el documento XML en forma de XMLFILE y, si desea utilizarlo en la memoria, puede utilizar la UDF Content(), que puede tomar un tipo de datos XMLFILE como entrada y devolver un CLOB.

La UDF Content() realiza dos funciones de recuperación diferentes, dependiendo del tipo de datos especificado: Esta UDF:

Recupera un documento del almacenamiento externo y lo coloca en la memoria

Puede utilizar Content() para recuperar un documento XML y colocarlo en una memoria intermedia o en un localizador de CLOB cuando el documento está almacenado como archivo externo. Utilice la siguiente sintaxis de función, en la que *objxml* es la columna XML que es objeto de la consulta:

De XMLFILE a CLOB: Recupera datos de un archivo y los exporta a un localizador de CLOB.

```
Content(objxml XMLFile)
```

Recupera un documento del almacenamiento interno y lo exporta a un archivo externo

También puede utilizar Content() para recuperar un documento XML, que está almacenado en DB2 como un tipo de datos XMLCLOB, y exportarlo a un archivo del sistema de archivos del servidor de bases de datos. Content() devuelve el nombre del archivo del tipo VARCHAR. Utilice la siguiente sintaxis de función, en la que *objxml* es la columna XML que

es objeto de la consulta y *nombreArchivo* es el archivo externo. *tipo XML* puede ser el tipo de datos XMLVARCHAR o XMLCLOB.

De tipo XML **a archivo externo:** Recupera un documento XML que está almacenado como tipo de datos XML y lo exporta a un archivo externo.

```
Content(xmlobj XML type, nombreArchivo varchar(512))
```

Donde:

objxml Es el nombre de la columna XML donde reside el contenido XML que se debe recuperar; *objxml* puede ser de tipo XMLVARCHAR o XMLCLOB.

nombreArchivo
Es el nombre del archivo donde se deben almacenar los datos XML.

El ejemplo siguiente es un pequeño segmento de programa escrito en C que contiene *SQL incorporado;* este programa recupera un documento XML de un archivo y lo coloca en la memoria. En este ejemplo se supone que la columna BOOK es de tipo XMLFILE.

```
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 {
               /* aquí se indican las operaciones hechas sobre*/
               /* el doc XML del almacenamiento intermedio */
               }
             }
          EXEC SQL CLOSE c1;
          EXEC SQL CONNECT RESET;
```

Recuperación del contenido de elementos y de valores de atributos

Puede recuperar (extraer) el contenido de un *elemento* o el valor de un *atributo* en uno o más documentos XML (búsqueda en documento individual o en documento de colección). El XML Extender proporciona funciones de extracción definidas por el usuario que puede especificar en una cláusula SELECT para cada tipo de datos SQL.

Recuperar el contenido de elementos y valores de atributos es útil en el desarrollo de aplicaciones, pues le permite acceder a datos XML en forma de datos relacionales. Por ejemplo, suponga que tiene 1000 documentos XML almacenados en la columna ORDER de la tabla SALES_TAB. Puede recuperar los nombres de todos los clientes que han hecho pedido de artículos; para ello utilice la siguiente sentencia de SQL cuya cláusula SELECT contiene la UDF de extracción:

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

En este ejemplo, la UDF de extracción recupera el elemento <customer> de la columna ORDER, devolviendo el resultado en el tipo de datos VARCHAR. La vía de ubicación es `/Order/Customer/Name` (consulte el apartado "Vía de ubicación" en la página 55 para obtener la sintaxis de la vía de ubicación). Además, el número de valores devueltos se puede disminuir utilizando una cláusula WHERE, que especifica que sólo el contenido del elemento <customer> cuyo subelemento es <ExtendedPrice> tiene un valor mayor que 2500,00.

Para extraer el contenido de elementos o valores de atributos: Utilice las UDF de extracción que aparecen en la Tabla 13 en la página 129; utilice la sintaxis siguiente como función de tabla o *función escalar*:

```
extracttipoDatos_recuperado(objxml, vía)
```

Donde:

tipoDatos_recuperado
: Es el tipo de datos que devuelve la función de extracción; puede ser uno de los tipos siguientes:
  - INTEGER
  - SMALLINT
  - DOUBLE
  - REAL
  - CHAR
  - VARCHAR
  - CLOB
  - DATE
  - TIME
  - TIMESTAMP
  - FILE

objxml Es el nombre de la columna XML donde reside el elemento o atributo

que se debe extraer. Esta columna debe estar definida como perteneciente a uno de los siguientes UDT (tipos definidos por el usuario) de XML:

- XMLVARCHAR
- XMLCLOB como LOCATOR
- XMLFILE

vía Es la vía de ubicación del elemento o atributo del documento XML (como por ejemplo `/Order/Customer/Name`). Vea "Vía de ubicación" en la página 55 para obtener la sintaxis de la vía de acceso a la ubicación.

Importante: Tenga en cuenta que las UDF de extracción dan soporte a vías de ubicación que tienen predicados con atributos, pero no elementos. Por ejemplo, se da soporte al siguiente predicado:

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

El siguiente predicado no está soportado:

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

La Tabla 13 muestra las funciones de extracción, en el formato escalar y de tabla:

Tabla 13. Funciones de extracción del XML Extender

Función escalar	Función de tabla	Nombre de columna devuelto (función de tabla)	Tipo devuelto
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

Ejemplo de función escalar:

En el ejemplo siguiente, se devuelve un valor cuando el valor de atributo de clave = ″1″. El valor se extrae como un entero convertido automáticamente a un tipo DECIMAL.

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

Ejemplo de función de tabla:

En el siguiente ejemplo, cada valor de clave correspondiente al pedido de ventas se extrae como un INTEGER

```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

Actualización de datos XML

Mediante el XML Extender, puede actualizar el documento XML completo sustituyendo los datos de las columnas XML, o puede actualizar los valores de elementos o atributos especificados.

Visión general de las tareas:

1. Asegúrese de que ha almacenado el documento en una tabla XML y determine qué datos desea recuperar.
2. Elija un método para actualizar los datos de la tabla DB2 (funciones de conversión de datos o funciones UDF).
3. Mediante una consulta SQL, especifique la tabla y columna XML que se debe actualizar.

Importante: Cuando se actualiza una columna que está habilitada para XML, el XML Extender actualiza automáticamente las tablas auxiliares para reflejar los cambios. Pero el usuario no debe actualizar directamente estas tablas, modificando el correspondiente valor de elemento o de atributo XML, sin actualizar el documento XML original contenido en la columna XML. Dichas actualizaciones puede provocar problemas de falta de coherencia en los datos.

Para actualizar un documento XML:

Utilice uno de los métodos siguientes:

Uso de una función predefinida de conversión de datos

Para cada UDT (tipo definido por el usuario), existe una función predefinida de conversión de datos para convertir el tipo base SQL al UDT. Para actualizar el documento XML, puede utilizar las funciones de conversión de datos proporcionadas por el XML Extender. La Tabla 9 en la página 123 muestra las funciones de conversión de datos proporcionadas; se supone que la columna ORDER está creada mediante un UDT diferente proporcionado por el XML Extender.

Ejemplo: El ejemplo anterior actualiza el tipo XMLVARCHAR, a partir del tipo convertido VARCHAR, suponiendo que `xml_buf` es una variable de lenguaje principal que está definida con el tipo VARCHAR.

```
UPDATE sales_tab VALUES('123456', 'Sriram Srinivasan',
  db2xml.XMLVarchar(:xml_buff))
```

Uso de una UDF de almacenamiento

Para cada UDT del XML Extender, existe una UDF de almacenamiento para importar datos a DB2 desde una fuente distinta del tipo base de DB2. Puede utilizar una UDF de almacenamiento para actualizar el documento XML completo mediante su sustitución.

Ejemplo: El ejemplo siguiente actualiza un documento XML utilizando la función XMLVarcharFromFile():

```
UPDATE sales_tab
       set order = XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml')
   WHERE sales_person = 'Sriram Srinivasan'
```

El ejemplo siguiente actualiza el objeto XML desde el archivo llamado `c:\dxx\samples\cmd\getstart.xml` a la columna ORDER de la tabla SALES_TAB.

Vea la Tabla 10 en la página 124 para obtener una lista de las funciones de almacenamiento proporcionadas por el XML Extender.

Para actualizar elementos o atributos determinados de un documento XML:

Utilice la UDF Update() para actualizar un valor cada vez, en vez de actualizar el documento completo. Mediante la UDF, especifique una vía de ubicación y el valor del elemento o atributo que se debe cambiar, que está representado por la vía de ubicación. (Vea “Vía de ubicación” en la página 55 para obtener la sintaxis de la vía de ubicación.) No es necesario que edite el documento XML: el XML Extender realiza el cambio automáticamente.

La UDF Update actualiza el archivo XML completo y reconstruye el archivo basándose en información del analizador de XML. Para saber cómo UDF Update procesa el documento y para obtener documentos de ejemplo antes y después de que se actualicen, consulte el apartado “Cómo la función Update procesa el documento XML” en la página 207.

Sintaxis:

`Update(`*`xmlobj`*`, `*`vía`*`, `*`valor`*`)`

Donde:

objxml Es el nombre de la columna XML para la que debe actualizar el valor del elemento o atributo.

vía Es la vía de ubicación del elemento o atributo que se debe actualizar. Vea "Vía de ubicación" en la página 55 para obtener la sintaxis de la vía de acceso a la ubicación. Para conocer las consideraciones para aparición múltiple, consulte el apartado "Aparición múltiple" en la página 210.

valor Es el valor que se debe actualizar.

Ejemplo: La sentencia siguiente actualiza el valor del elemento <Customer> para que sea igual a la serie de caracteres ″IBM″; para ello se utiliza la UDF Update():

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

Aparición múltiple:

Cuando se proporciona una vía de ubicación en UDF Update(), el contenido de cada elemento o atributo con una vía coincidente se actualiza con el valor suministrado. Esto significa que si un documento tiene vías de ubicación de aparición múltiple, la función Update sustituye los valores existentes por el valor suministrado en el parámetro *valor*.

Búsqueda de documentos XML

Buscar datos XML es similar a recuperar datos XML: ambas técnicas recuperan datos para su posterior manejo, pero la búsqueda utiliza una cláusula WHERE para definir predicados que sirven como criterios de recuperación.

El XML Extender proporciona varios métodos para buscar documentos XML en una columna XML, de acuerdo con las necesidades de la aplicación del usuario. Puede hacer una búsqueda estructural de documento y obtener resultados basados en el contenido de elementos y valores de atributos. Puede buscar una vista de la columna XML y sus tablas auxiliares, buscar directamente las tablas auxiliares para obtener un mejor rendimiento, o utilizar las UDF de extracción junto con cláusulas WHERE. Además, puede utilizar el Text Extender de DB2 y buscar datos de columna dentro del contenido estructural correspondiente a una cadena de texto.

A fin de realizar búsquedas rápidas, con el XML Extender puede utilizar índices de columnas de tabla auxiliar, que contienen elementos o atributos XML extraídos de documentos XML. Puede especificar el tipo de datos de un elemento o atributo para buscar de acuerdo con el tipo de datos general de SQL o para realizar búsquedas dentro de un ámbito. Por ejemplo, en el supuesto referente a la orden de compra, podría buscar todos los pedidos cuyo precio global (″extended price″) fuera mayor que 2500,00.

Puede también utilizar el DB2 UDB Text Extender para realizar una búsqueda estructural de texto o una búsqueda completa de texto. Por ejemplo, suponga que la columna RESUME contiene currículums de personas en formato XML. Desea obtener el nombre de todos los candidatos a un puesto de trabajo que tengan conocimientos sobre Java. Podría utilizar el Text Extender de DB2 para buscar en los documentos XML todos los currículums cuyo elemento <conocimientos> contenga la cadena de caracteres JAVA.

Las secciones siguientes describen métodos de búsqueda:

- "Búsqueda en el documento XML por estructura"
- "Utilización del Text Extender para la búsqueda estructural de texto" en la página 135

Búsqueda en el documento XML por estructura

Mediante las funciones de búsqueda del XML Extender, puede buscar datos XML de una columna basándose en la estructura del documento, es decir en elementos y atributos. Para buscar los datos de la columna, puede utilizar una sentencia SELECT de varias maneras y obtener un *conjunto resultante* basado en las coincidencias con elementos y atributos del documento. Puede buscar datos de columnas utilizando los métodos siguientes:

- Búsqueda mediante consulta directa sobre tablas auxiliares
- Búsqueda desde una *vista de unión*
- Búsqueda mediante las UDF de extracción
- Búsqueda de elementos o atributos de aparición múltiple

Las secciones que siguen describen estos métodos y utilizan ejemplos de acuerdo con los supuestos siguientes. La tabla de aplicación SALES_TAB tiene una columna XML llamada ORDER. Esta columna tiene tres tablas auxiliares: ORDER_SIDE_TAB, PART_SIDE_TAB y SHIP_SIDE_TAB. Cuando se habilitó la columna ORDER, se especificó una vista predefinida, sales_order_view, que une las tablas auxiliares utilizando la siguiente sentencia 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
```

Búsqueda mediante consulta directa sobre tablas auxiliares

La búsqueda mediante consulta y subconsulta directa proporciona un rendimiento óptimo en la búsqueda estructural cuando las tablas auxiliares están indexadas. Puede utilizar una consulta o subconsulta para buscar en tablas auxiliares correctamente.

Ejemplo: La sentencia siguiente utiliza una consulta y una subconsulta para buscar directamente en una tabla auxiliar:

```
SELECT sales_person from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

En este ejemplo, invoice_num es la clave primaria de la tabla SALES_TAB.

Búsqueda desde una vista de unión

Puede hacer que el XML Extender cree una vista predefinida que asocie la tabla de aplicación y las tablas auxiliares utilizando un ID exclusivo. Puede utilizar esta vista predefinida, o cualquier vista que asocie la tabla de aplicación y las tablas auxiliares, para buscar datos de columna y consultar tablas auxiliares. Este método proporciona una vista virtual única de la tabla de aplicación y de sus tablas auxiliares. Sin embargo, cuantas más tablas auxiliares se creen, más costosa es la consulta.

Sugerencia: Puede utilizar el *id_raíz*, o DXXROOT_ID (creado por el XML Extender), para unir las tablas cuando cree la vista.

Ejemplo: La sentencia siguiente busca en una vista

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

La sentencia de SQL devuelve los valores de sales_person a partir de la vista de unión sales_order_view, cuyos pedidos tiene un precio mayor que 2500,00.

Búsqueda mediante las UDF de extracción

Puede también utilizar las UDF de extracción del XML Extender para buscar elementos y atributos, cuando no haya creado índices ni tablas auxiliares para la tabla de aplicación. La utilización de las UDF de extracción para examinar los datos XML es muy costosa y sólo debe hacerse con cláusulas WHERE que restrinjan el número de documentos XML que se incluyen en la búsqueda.

Ejemplo: La sentencia siguiente realiza una búsqueda mediante una UDF de extracción del XML Extender:

```
SELECT sales_person from sales_tab
     WHERE extractVarchar(order, '/Order/Customer/Name')
    like '%IBM%'
AND invoice_num > 100
```

En este ejemplo, la UDF de extracción extrae los elementos </Order/Customer/Name> que tienen el valor ″IBM″.

Búsqueda de elementos o atributos de aparición múltiple

Cuando realice una búsqueda de elementos o atributos de aparición múltiple, utilice la cláusula DISTINCT para evitar valores duplicados.

Ejemplo: La sentencia siguiente realiza una búsqueda mediante la cláusula DISTINCT:

```
SELECT sales_person from sales_tab
       WHERE invoice_num in
          (SELECT DISTINCT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

En este ejemplo, el archivo DAD especifica que `/Order/Part/Price` aparece varias veces y crea la tabla auxiliar PART_SIDE_TAB para él. La tabla PART_SIDE_TAB puede contener más de una fila con el mismo valor para invoice_num. La utilización de DISTINCT permite que los valores devueltos sean exclusivos.

Utilización del Text Extender para la búsqueda estructural de texto

Cuando se realizan búsquedas de la estructura de documentos XML, el XML Extender busca valores de elementos y atributos que se convierten en tipos de datos generales, pero no busca texto. Puede utilizar el DB2 UDB Text Extender para realizar una búsqueda estructural o completa de texto en una columna que está habilitada para XML. El Text Extender da soporte a la búsqueda de documentos XML en DB2 UDB versión 6.1 o posterior. Text Extender está disponible en los sistemas operativos Windows, AIX y Sun Solaris.

Búsqueda estructural de texto

Busca cadenas de texto que están basadas en la estructura arborescente del documento XML. Por ejemplo, si tiene la estructura de documento `/Order/Customer/Name` y desea buscar la cadena de caracteres “IBM” dentro del subelemento <Customer>, puede utilizar una búsqueda estructural de texto. El documento puede también tener la cadena de caracteres ″IBM″ en un subelemento <Comment> o como nombre de parte de un producto. La búsqueda estructural de texto sólo busca la cadena de caracteres en los elementos especificados. En este ejemplo, sólo se obtienen los documentos que tienen ″IBM″ en el subelemento </Order/Customer/Name>; no se obtienen los documentos que tienen ″IBM″ en otros elementos, pero no en el subelemento </Order/Customer/Name>.

Búsqueda completa de texto

Busca cadenas de texto en todos los lugares de la estructura del documento, sin distinguir elementos ni atributos. En el ejemplo

anterior, esta búsqueda obtendría todos los documentos que tienen la cadena de texto "IBM", con independencia del lugar donde aparece la cadena.

Para realizar búsquedas mediante el Text Extender, debe instalar dicho producto y habilitar la base de datos y las tablas tal como se describe más abajo. Para aprender a realizar búsquedas con el Text Extender, vea el capítulo del manual *DB2 Universal Database Text Extender Administration and Programming* referente a búsquedas hechas con las UDF del Text Extender.

Habilitación de una columna XML para el Text Extender

Si tiene una base de datos habilitada para XML, siga los pasos siguientes para que el Text Extender pueda examinar el contenido de una columna habilitada para XML. Para los fines de nuestro ejemplo, la base de datos se llama SALES_DB, la tabla se llama ORDER y los nombres de las columnas XML son XVARCHAR y XCLOB:

1. Vea el archivo `install.txt`, contenido en el CD de Extenders, para conocer cómo instalar el Text Extender.
2. Entre el mandato **txstart** desde una de las ubicaciones siguientes:
   - En sistemas operativos UNIX, entre el mandato en el indicador de mandatos del propietario de instancia.
   - En Windows NT, entre el mandato en la ventana de mandatos donde está especificado DB2INSTANCE.
3. Abra la ventana de línea de mandatos del Text Extender. En este paso se supone que el usuario tiene una base de datos llamada SALES_DB y una tabla llamada ORDER, que tiene dos columnas XML cuyos nombres son XVARCHAR y XCLOB. Puede ser necesario que ejecute los programas de ejemplo contenidos en `dxx\samples\c`.
4. Conecte con la base de datos. En el indicador de mandatos de **db2tx**, escriba:

   ```
   'connect to SALES_DB'
   ```

5. Habilite la base de datos para el Text Extender.

   En el indicador de mandatos de **db2tx**, escriba:

   ```
   'enable database'
   ```

6. Habilite las columnas de la tabla XML para el Text Extender; para ello defina los tipos de datos del documento XML, el idioma, las páginas de códigos y demás información sobre la columna.
   - Para la columna XVARCHAR, de tipo VARCHAR, escriba:

     ```
     'enable text column order xvarchar function db2xml.varchartovarchar handle
     varcharhandle ccsid 850 language us_english format xml indextype precise
         indexproperty sections_enabled
     documentmodel (Order) updateindex update'
     ```

   - Para la columna XCLOB, de tipo CLOB, escriba:

```
'enable text column order xclob function db2xml.clob handle clobhandle
 ccsid 850 language us_english indextype precise updateindex update'
```

7. Compruebe el estado del índice.
   - Para la columna XVARCHAR, escriba: `get index status order handle varcharhandle`
   - Para la columna XCLOB, escriba: `get index status order handle clobhandle`
8. Defina el modelo del documento XML en el archivo de modelos de documento INI cuyo nombre es `desmodel.ini`. Este archivo está situado en `/db2tx/txins000` para UNIX, y en `\instance\\db2tx\txins000` para Windows NT, y secciones de él se encuentran en un archivo de inicialización. Por ejemplo, para el archivo `textmodel.ini`:

```
;list of document models
[MODELS]
modelname=Order

; an 'Order' document model definition
; left side = section name identifier
; right side = section name tag

[Order]
Order = /Order
Order/Customer/Name = /Order/Customer/Name
Order/Customer/Email = /Order/Customer/Email
Order/Part/@color = /Order/Part/@color
Order/Part/Shipment/ShipMode = /Order/Part/Shipment/ShipMode
```

Búsqueda de texto mediante el Text Extender

El recurso de búsqueda del Text Extender funciona bien con la búsqueda estructural de documentos del XML Extender. Es aconsejable crear una consulta para buscar elementos o atributos del documento y utilizar el Text Extender para buscar el contenido de elementos o los valores de atributos.

Ejemplo: Las sentencias siguientes buscan texto de un documento XML mediante el Text Extender. En la ventana de mandatos de DB2, escriba:

```
'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'
```

La UDF Contains() del Text Extender realiza una búsqueda.

Este ejemplo no contiene todos los pasos necesarios para buscar datos de columna con el Text Extender. Para obtener información sobre la búsqueda mediante el Text Extender, vea el capítulo del manual *DB2 Universal Database Text Extender Administration and Programming* referente a búsquedas hechas con las UDF del Text Extender.

Supresión de documentos XML

Utilice la sentencia DELETE de SQL para suprimir una fila de un documento XML en una columna XML. Puede especificar cláusulas WHERE para delimitar qué documentos se deben suprimir.

Ejemplo: Las sentencias siguientes suprimen todos los documentos que tienen un valor mayor que 2500,00 para <ExtendedPrice>.

```
DELETE from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

Limitaciones al invocar funciones desde JDBC

Cuando se utilizan marcadores de parámetros en funciones, una restricción de JDBC requiere que el marcador de parámetros de la función se convierta al tipo de datos de la columna en la que se insertarán los datos devueltos. La lógica de selección de la función no sabe cómo será el tipo de datos del argumento y no puede resolver la referencia.

Como resultado, JDBC no puede resolver el siguiente código:

```
db2xml.XML función_predefinida_conversión(longitud)
```

Puede utilizar la especificación CAST para ofrecer un tipo para el marcador de parámetros, como por ejemplo, VARCHAR y la lógica de selección de la función podrá proseguir:

```
db2xml.XML función_predefinida_conversión(CAST(? AS tipo_conv(longitud))
```

Ejemplo 1: En el siguiente ejemplo, el marcador de parámetros se convierte como VARCHAR. El parámetro que se pasa es un documento XML, que se convierte como VARCHAR(1000) y se inserta en la columna ORDER.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarchar(cast (? as varchar(1000))))";
```

Ejemplo 2: En el siguiente ejemplo, el marcador de parámetros se convierte como VARCHAR. El parámetro que se pasa es un nombre de archivo y el contenido del mismo se convierte a VARCHAR y se inserta en la columna ORDER.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarcharfromFILE(cast (? as varchar(1000))))";
```

Capítulo 6. Gestión de datos de una colección XML

Una colección XML es un conjunto de tablas relacionales que contienen datos que se correlacionan con documentos XML. Este método de acceso y almacenamiento le permite componer un documento XML a parir de datos existentes, descomponer un documento XML y utilizar XML como método de intercambio de datos.

Las tablas relacionales pueden ser tablas nuevas que el XML Extender genera al descomponer documentos XML, o tablas existentes cuyos datos son utilizados por el XML Extender para generar documentos XML para las aplicaciones del usuario. Los datos de las columnas de estas tablas no contienen distintivos XML, sino el contenido y los valores correspondientes a elementos y atributos, respectivamente. Los procedimientos almacenados son el método acceso y almacenamiento para almacenar, recuperar, actualizar, buscar y suprimir datos de colecciones XML.

Los límites de parámetro utilizados por los procedimientos almacenados de la colección XML se describen en el apartado "Apéndice D. Límites del XML Extender" en la página 295.

Puede aumentar los tamaños de CLOB para los resultados de los procedimientos almacenados que se describen en el apartado "Aumento del límite del CLOB" en la página 214.

Las secciones siguientes proporcionan información sobre la gestión de una colección XML:

- "Composición de documentos XML a partir de datos DB2"
- "Descomposición de documentos XML en datos DB2" en la página 148

Composición de documentos XML a partir de datos DB2

La composición es el proceso de crear un conjunto de documentos XML a partir de datos relacionales de una colección XML. Puede componer documentos XML utilizando procedimientos almacenados. Para utilizar estos procedimientos almacenados, debe crear un archivo DAD, el cual especifica la correlación existente entre el documento XML y las tablas DB2. Los procedimientos almacenados utilizan el archivo DAD para componer el documento XML. Vea "Planificación para utilizar colecciones XML" en la página 58 para aprender a crear un archivo DAD.

Antes de comenzar

- Correlacione la estructura del documento XML con las tablas relacionales que contienen el contenido de los elementos y los valores de atributos.
- Seleccione un método de correlación: correlación SQL o correlación de nodo_RDB.
- Prepare el archivo DAD. Vea "Planificación para utilizar colecciones XML" en la página 58 para obtener detalles completos.
- Opcionalmente, habilite la colección XML.

Proceso de composición del documento XML

El XML Extender proporciona dos procedimientos almacenados, dxxGenXML() y dxxRetrieveXML(), para componer documentos XML.

dxxGenXML()

Este procedimiento almacenado se utiliza para aplicaciones que realizan actualizaciones ocasionales o para aplicaciones que desean eludir la actividad asociada a la administración de datos XML. El procedimiento almacenado dxxGenXML() no necesita una colección habilitada; en su lugar utiliza un archivo DAD.

El procedimiento almacenado dxxGenXML() crea documentos XML utilizando datos que están almacenados en tablas de colecciones XML, que están especificadas por el elemento <Xcollection> en el archivo DAD. Este procedimiento almacenado inserta cada documento XML como una fila en una *tabla resultante*. El usuario también puede abrir un cursor en la tabla resultante y obtener el conjunto resultante. La tabla resultante debe ser creada por la aplicación y contiene siempre una sola columna de tipo VARCHAR, CLOB, XMLVARCHAR o XMLCLOB.

Además, si se especifica el elemento de validación del archivo DAD como YES, el XML Extender añade la columna DXX_VALID de tipo INTEGER a la tabla resultante e inserta un valor de 1 para un documento XML válido y 0 para un documento no válido.

El procedimiento almacenado dxxGenXML() también permite al usuario especificar el número máximo de filas que se han de generar en la tabla resultante. Esto reduce el tiempo de proceso. El procedimiento almacenado devuelve el número real de filas de la tabla, junto con los códigos y mensajes de retorno producidos.

El correspondiente procedimiento almacenado para la descomposición es dxxShredXML(); también utiliza el archivo DAD como parámetro de entrada y no necesita que la colección XML esté habilitada.

Para componer una colección XML: dxxGenXML()

Intercale una llamada de procedimiento almacenado en su aplicación, mediante esta declaración:

```
dxxGenXML(CLOB(100K)     DAD,             /* entrada */
          char(resultTabName) resultTabName,   /* entrada */
          integer        overrideType,    /* entrada */
          varchar(1024)  override,        /* entrada */
          integer        maxRows,         /* entrada */
          integer        numRows,         /* salida */
          long           returnCode,      /* salida */
          varchar(1024)  returnMsg)       /* salida */
```

Consulte “dxxGenXML()” en la página 224 para conocer la sintaxis completa y ver ejemplos.

Ejemplo: El ejemplo siguiente compone un documento XML:

```
#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; /* archivo DAD */
     char    result_tab[32];       /* nombre de la tabla resultante */
     char    override[2];          /* alteración temporal, para NULL */
     short   overrideType;         /* definido en dxx.h */
     short   max_row;              /* número máximo de filas */
     short   num_row;              /* número real de filas */
     long    returnCode;           /* código de error de retorno */
     char    returnMsg[1024];      /* texto del mensaje de error */
     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;

     /* crear tabla */
     EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

     /* leer datos de un archivo y ponerlos en un CLOB */
     strcpy(dadfile.name,"c:\dxx\samples\dad\
                 getstart_xcollection.dad");
     dadfile.name_length = strlen("c:\dxx\samples\dad\
                 getstart_xcollection.dad");
     dadfile.file_options = SQL_FILE_READ;
     EXEC SQL VALUES (:dadfile) INTO :dad;
     strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
     max_row = 500;
     num_row = 0;
     returnCode = 0;
     msg_txt[0] = '\0';
     collection_ind = 0;
     dad_ind = 0;
     rtab_ind = 0;
     ov_ind = -1;
     ovtype_ind = 0;
     maxrow_ind = 0;
     numrow_ind = -1;
     returnCode_ind = -1;
     returnMsg_ind = -1;

     /* Llamar al procedimiento almacenado */
     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);
```

Después de llamar al procedimiento almacenado, la tabla resultante contiene 250 filas, pues la consulta SQL especificada en el archivo DAD generó 250 documentos XML.

dxxRetrieveXML()

Este procedimiento almacenado se utiliza para aplicaciones que realizan actualizaciones frecuentes. Debido a que se repiten las mismas tareas, es importante conseguir un rendimiento mejorado. Habilitar una colección XML y utilizar su nombre en el procedimiento almacenado mejora el rendimiento.

El procedimiento almacenado dxxRetrieveXML() trabaja de la misma manera que el procedimiento almacenado dxxGenXML(), salvo que utiliza como entrada el nombre de una colección XML habilitada en lugar de un archivo DAD. Cuando se habilita una colección XML, se almacena un archivo DAD en la tabla XML_USAGE. Por tanto, el XML Extender recupera el archivo DAD y, a partir de este punto, dxxRetrieveXML() se comporta igual que dxxGenXML().

dxxRetrieveXML() permite que un mismo archivo DAD sea utilizado tanto para la composición como para la descomposición. Este procedimiento almacenado también se puede utilizar para recuperar documentos XML descompuestos.

El correspondiente procedimiento almacenado para la descomposición es dxxInsertXML(); también utiliza como entrada el nombre de una colección XML habilitada.

Para componer una colección XML: dxxRetrieveXML()

Intercale una llamada de procedimiento almacenado en su aplicación, mediante esta declaración:

```
dxxRetrieveXML(char(collectionName) collectionName,  /* entrada */
               char(resultTabName) resultTabName,   /* entrada */
               integer        overrideType,    /* entrada */
               varchar(1024)  override,        /* entrada */
               integer        maxRows,         /* entrada */
               integer        numRows,         /* salida */
               long          returnCode,       /* salida */
               varchar(1024) returnMsg)        /* salida */
```

Consulte “dxxRetrieveXML()” en la página 228 para conocer la sintaxis completa y ver ejemplos.

Ejemplo: El ejemplo siguiente es una llamada a dxxRetrieveXML(). En el ejemplo, se crea la tabla resultante XML_ORDER_TAB, la cual tiene una sola columna de tipo XMLVARCHAR.

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

        EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
        char    collection[32];  /* almacenamiento intermedio DAD */
        char    result_tab[32];  /* nombre de la tabla resultante */
        char    override[2];     /* alteración temporal, para NULL */
        short   overrideType;    /* definido en dxx.h */
        short   max_row;         /* número máximo de filas */
        short   num_row;         /* número real de filas */
        long    returnCode;      /* código de error de retorno */
        char    returnMsg[1024]; /* texto del mensaje de error */
        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;

        /* crear tabla */
        EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

        /* inicializar variable de lenguaje principal e indicadores */
        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;

        /* Llamar al procedimiento almacenado */
        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);
```

Anulación dinámica de valores en el archivo DAD

En las consultas dinámicas, puede utilizar dos parámetros opcionales para anular condiciones definidas en el archivo DAD: *alteración_temporal* y *tipo_de_alteración_temporal*. Basándose en el valor de *tipo_alter_temp*, la aplicación puede anular la sentencia SQL_stmt, para la correlación SQL, o las condiciones del nodo_RDB, para la correlación de nodo_RDB, en el archivo DAD.

Estos parámetros tienen los valores y reglas siguientes:

tipo_alter_temp
Es un parámetro de entrada obligatorio que indica el tipo del parámetro *alteración_temporal*. *tipo_alter_temp* tiene los valores siguientes:

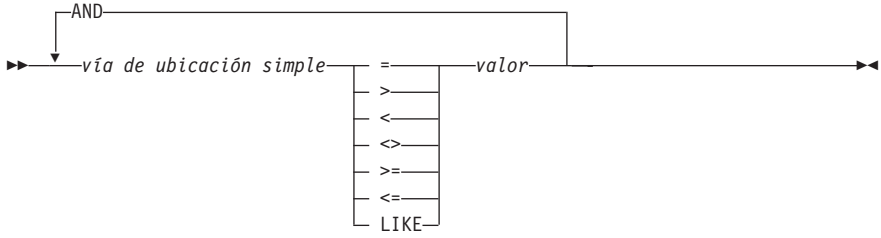
NO_OVERRIDE
Especifica que no se invalide una condición definida en el archivo DAD.

SQL_OVERRIDE
Especifica que se invalide, mediante una sentencia de SQL, una condición definida en el archivo DAD.

XML_OVERRIDE
Especifica que se invalide, mediante una condición basada en XPath, una condición definida en el archivo DAD.

alter_temp
Es un parámetro opcional de entrada que especifica la condición de alteración temporal para el archivo DAD. La sintaxis del valor de entrada se corresponde con el valor especificado en *overrideType*.

- Si especifica NO_OVERRIDE, el valor de entrada es una serie nula.
- Si especifica SQL_OVERRIDE, el valor de entrada es una sentencia de SQL válida.***Obligatorio:*** Si utiliza SQL_OVERRIDE y una sentencia de SQL, deberá utilizar el esquema de correlación de SQL del archivo DAD. La sentencia de SQL de entrada invalida la sentencia de SQL especificada por el elemento <SQL_stmt> en el archivo DAD.
- Si utiliza XML_OVERRIDE, el valor de entrada es una serie que contiene una o más expresiones. ***Obligatorio:*** Si utiliza XML_OVERRIDE y una expresión, deberá utilizar el esquema de correlación de RDB_node del archivo DAD. La expresión XML de entrada invalida la condición de nodo_RDB especificada en el archivo DAD. La expresión utiliza la sintaxis siguiente:



Donde:

vía de ubicación simple
Es una vía de ubicación simple cuya sintaxis está definida por XPATH; vea la Tabla 5 en la página 57 para conocer la sintaxis.

operadores
Pueden tener un espacio para separar el operador de las demás partes de la expresión.

valor
Es un valor numérico o una serie individual entre comillas.

Pueden haber espacios opcionales alrededor de las operaciones; los espacios son obligatorios alrededor del operador LIKE.

Cuando se especifica el valor XML_OVERRIDE, la expresión especificada invalida la condición del nodo_RDB, en el nodo_de_texto o nodo_de_atributo correspondiente a la vía de ubicación simple.

XML_OVERRIDE no se ajusta completamente a XPath. La vía de ubicación simple sólo se utiliza para identificar el elemento o atributo que está correlacionado con una columna.

Ejemplos:

Los ejemplos siguientes muestran una alteración dinámica de valores mediante el uso de SQL_OVERRIDE y XML_OVERRIDE. En el presente manual, la mayoría de los ejemplos de procedimientos almacenados utilizan NO_OVERRIDE.

Ejemplo: Procedimiento almacenado que hace uso de SQL_OVERRIDE.

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

         EXEC SQL INCLUDE SQLCA;
         EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];   /* almacenamiento intermedio con DAD */
         char    result_tab[32];   /* nombre de la tabla resultante */
```

```
char    override[256];    /* alter. temporal, SQL_stmt */
short   overrideType;     /* definido en dxx.h */
short   max_row;          /* número máximo de filas */
short   num_row;          /* número real de filas */
long    returnCode;       /* código de error de retorno */
char    returnMsg[1024];  /* texto del mensaje de error */
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;

/* crear tabla */
EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

/* inicializar variable de lenguaje principal e indicadores */
strcpy(collection,"sales_ord");
strcpy(result_tab,"xml_order_tab");
sprintf(override,"%s %s %s %s %s %s %s",
       "SELECT o.order_key, customer, p.part_key, quantity, price,",
       "tax, ship_id, date, mode ",
       "FROM order_tab o, part_tab p,",
       "table(select substr(char(timestamp(generate_unique())),16",
       "as ship_id,date,mode from ship_tab)as s",
       "WHERE p.price > 50.00 and s.date >'1998-12-01' AND",
       "p.order_key = o.order_key and s.part_key = p.part_key");
overrideType = SQL_OVERRIDE;
max_row = 500;
num_row = 0;
returnCode = 0;
msg_txt[0] = '\0';
collection_ind = 0;
rtab_ind = 0;
ov_ind = 0;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Llamar al procedimiento almacenado */
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);
```

En este ejemplo, el elemento <xcollection> del archivo DAD debe tener un elemento <SQL_stmt>. El parámetro *alteración_temporal* invalida el valor de

<SQL_stmt>, pues cambia el precio para que sea mayor que 50.00 y cambia a la fecha para que sea posterior a 1998-12-01.

Ejemplo: Procedimiento almacenado que hace uso de XML_OVERRIDE.

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

       EXEC SQL INCLUDE SQLCA;
       EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];    /* almacenamiento intermedio con DAD */
         char    result_tab[32];    /* nombre de la tabla resultante */
         char    override[256];    /* alter. temporal, SQL_stmt */
         short   overrideType;      /* definido en dxx.h */
         short   max_row;           /* número máximo de filas */
         short   num_row;           /* número real de filas */
         long    returnCode;        /* código de error de retorno */
         char    returnMsg[1024];   /* texto del mensaje de error */
         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;

         /* crear tabla */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

         /* inicializar variable de lenguaje principal e indicadores */
         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;

         /* Llamar al procedimiento almacenado */
         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);
```

En este ejemplo, el elemento <xcollection> del archivo DAD tiene un nodo_RDB para el nodo_de_elemento raíz. El valor de *alteración_temporal* está basado en el contenido de XML. El XML Extender convierte la vía de ubicación simple en la columna DB2 correlacionada.

Descomposición de documentos XML en datos DB2

Descomponer un documento XML es desglosar los datos que contiene y almacenarlos en tablas relacionales. El XML Extender proporciona procedimientos almacenados para descomponer datos XML, procedentes de documentos XML fuente, en tablas relacionales. Para utilizar estos procedimientos almacenados debe crear un archivo DAD, el cual especifica la correlación existente entre el documento XML y la estructura de tablas DB2. Los procedimientos almacenados utilizan el archivo DAD para descomponer el documento XML. Vea "Planificación para utilizar colecciones XML" en la página 58 para aprender a crear un archivo DAD.

Habilitación de una colección XML para descomposición

En la mayoría de los casos, es necesario que habilite una colección XML para poder utilizar los procedimientos almacenados. Es necesario que habilite una colección XML en estos casos:

- Al descomponer documentos XML y almacenar los datos en nuevas tablas, se debe habilitar una colección XML, pues el XML Extender crea todas las tablas de la colección XML cuando ésta se habilita.
- Cuando sea importante mantener la secuencia de elementos y atributos que aparecen varias veces (aparición múltiple). El XML Extender sólo conserva el orden secuencial de elementos o atributos de aparición múltiple para las tablas que se crean al habilitar una colección. Cuando se descomponen documentos XML en tablas relacionales existentes, no se puede garantizar que se conserve el orden secuencial.

Si desea pasar el archivo DAD espontáneamente cuando las tablas ya existen en la base de datos, no es necesario que habilite una colección XML.

Límites de tamaño de tabla de descomposición

La descomposición utiliza la correlación de nodo_RDB para especificar cómo un documento XML se descompone en tablas DB2 extrayendo los valores de elementos y de atributos en filas de tabla. Los valores para cada documento XML se almacenan en una o más tablas DB2. Cada tabla puede tener un máximo de 1.024 filas descompuestas de cada documento.

Por ejemplo, si un documento XML se descompone en cinco tablas, cada una de las cinco tablas puede tener un máximo de 1.024 filas para este documento concreto. Si la tabla tiene filas para varios documentos, puede tener un máximo de 1.024 filas para cada documento. Si la tabla tiene 20 documentos, puede tener 20.480 filas, 1.024 para cada documento.

La utilización de elementos de aparición múltiple (elementos con vías de ubicación que pueden aparecer más de una vez en la estructura XML) afecta el número de filas. Por ejemplo, un documento que contiene un elemento <Part> que aparece 20 veces, puede descomponerse como 20 filas en una tabla. Cuando utilice elementos de aparición múltiple, considere esta limitación del tamaño de tabla.

Antes de comenzar

- Correlacione la estructura del documento XML con las tablas relacionales que contienen el contenido de los elementos y los valores de atributos.
- Prepare el archivo DAD, utilizando la correlación de nodo_RDB. Vea "Planificación para utilizar colecciones XML" en la página 58 para obtener detalles.
- Opcionalmente, habilite la colección XML.

Descomposición del documento XML

El XML Extender proporciona dos procedimientos almacenados, dxxShredXML() y dxxInsertXML(), para descomponer documentos XML.

dxxShredXML()

Este procedimiento almacenado se utiliza para aplicaciones que realizan actualizaciones ocasionales o para aplicaciones que desean eludir la actividad asociada a la administración de datos XML. El procedimiento almacenado dxxShredXML() no necesita una colección habilitada; en su lugar utiliza un archivo DAD.

Este procedimiento almacenado utiliza dos parámetros de entrada: un archivo DAD y el documento XML que se debe descomponer; devuelve dos parámetros de salida: un código de retorno y un mensaje de retorno.

El procedimiento almacenado dxxShredXML() inserta un documento XML en una colección XML de acuerdo con la especificación <Xcollection> del archivo DAD de entrada. Se supone que existen las tablas utilizadas en la especificación <Xcollection> del archivo DAD y que las columnas satisfacen los tipos de datos especificados en la correlación DAD. Si esto no es así, se devuelve un mensaje de error. Luego, el procedimiento almacenado dxxShredXML() descompone el documento XML e inserta los datos XML, sin códigos, en las tablas especificadas en el archivo DAD.

El correspondiente procedimiento almacenado para la composición es dxxGenXML(); también utiliza el archivo DAD como parámetro de entrada y no necesita que la colección XML esté habilitada.

Para descomponer una colección XML: dxxShredXML()

Intercale una llamada de procedimiento almacenado en su aplicación, mediante esta declaración:

```
dxxShredXML(CLOB(100K)    DAD,            /* entrada */
            CLOB(1M)       xmlobj,         /* entrada */
            long           returnCode,     /* salida */
            varchar(1024) returnMsg)       /* salida */
```

Consulte "dxxShredXML()" en la página 233 para conocer la sintaxis completa y ver ejemplos.

Ejemplo: Lo siguiente es un ejemplo de una llamada a 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; /* archivo DAD */
         SQL TYPE is CLOB xmlDoc;        /* documento XML de entrada */
         SQL TYPE is CLOB_FILE xmlFile;  /* archivo XML de entrada */
         long        returnCode; /* código de error */
         char        returnMsg[1024]; /* texto del mensaje de error */
         short       dad_ind;
         short       mlDoc_ind;
         short       returnCode_ind;
         short       returnMsg_ind;
        EXEC SQL END DECLARE SECTION;

         /* inicializar variable de lenguaje principal e indicadores */
        strcpy(dadFile.name,"c:\dxx\samples\dad\
                            getstart_xcollection.dad");
        dadFile.name_length=strlen("c:\dxx\samples\dad\
                            getstart_xcollection.dad");
        dadFile.file_option=SQL_FILE_READ;
        strcpy(xmlFile.name,"c:\dxx\samples\cmd\
                            getstart_xcollection.xml");
        xmlFile.name_length=strlen("c:\dxx\samples\cmd\
                            getstart_xcollection.xml");
        xmlFile.file_option=SQL_FILE_READ;
        SQL EXEC VALUES (:dadFile) INTO :dad;
        SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
        returnCode = 0;
        returnMsg[0] = '\0';
        dad_ind = 0;
        xmlDoc_ind = 0;
        returnCode_ind = -1;
        returnMsg_ind = -1;

         /* Llamar al procedimiento almacenado */
        EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
             :xmlDoc:xmlDoc_ind,
             :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

Este procedimiento almacenado se utiliza para aplicaciones que realizan actualizaciones frecuentes. Debido a que se repiten las

mismas tareas, es importante conseguir un rendimiento mejorado. Habilitar una colección XML y utilizar su nombre en el procedimiento almacenado mejora el rendimiento. El procedimiento almacenado dxxInsertXML() trabaja de la misma manera que dxxShredXML(), salvo que utiliza una colección XML habilitada como primer parámetro de entrada.

El procedimiento almacenado dxxInsertXML() inserta un documento XML en una colección XML habilitada, que está asociada a un archivo DAD. El archivo DAD contiene especificaciones para las tablas de la colección y para la correlación. Las tablas de la colección se comprueban o crean de acuerdo con las especificaciones de <Xcollection>. Luego, el procedimiento almacenado dxxInsertXML() descompone el documento XML de acuerdo con la correlación e inserta datos XML, sin códigos, en las tablas de la colección XML especificada.

El correspondiente procedimiento almacenado para la composición es dxxRetrieveXML(); también utiliza como entrada el nombre de una colección XML habilitada.

Para descomponer una colección XML: dxxInsertXML()

Intercale una llamada de procedimiento almacenado en su aplicación, mediante esta declaración:

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

Consulte “dxxInsertXML()” en la página 235 para conocer la sintaxis completa y ver ejemplos.

Ejemplo: Lo siguiente es un ejemplo de una llamada a dxxInsertXML():

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

EXEC SQL INCLUDE SQLCA;
       EXEC SQL BEGIN DECLARE SECTION;
       char          collection[64];  /* nombre de una colección XML */
       SQL TYPE is CLOB_FILE xmlFile; /* archivo XML de entrada */
       SQL TYPE is CLOB      xmlDoc;  /* documento XML de entrada */
       long          returnCode;      /* código de error */
       char          returnMsg[1024]; /* texto del mensaje de error */
       short         collection_ind;
       short         xmlDoc_ind;
       short         returnCode_ind;
       short         returnMsg_ind;
       EXEC SQL END DECLARE SECTION;

       /* inicializar variable de lenguaje principal e indicadores */
       strcpy(collection,"sales_ord")
       strcpy(xmlobj.name,"c:\dxx\samples\cmd\
              getstart_xcollection.xml");
       xmlobj.name_length=strlen("c:\dxx\samples\cmd\
```

```
      getstart_xcollection.xml");
xmlobj.file_option=SQL_FILE_READ;
SQL EXEC VALUES (:xmlFile) INTO (:xmlDoc);
returnCode = 0;
returnMsg[0] = '\0';
collection_ind = 0;
xmlobj_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Llamar al procedimiento almacenado */
EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
        :xmlDoc:xmlDoc_ind,
        :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Acceso a una colección XML

Puede actualizar, suprimir, buscar y recuperar colecciones XML. Pero recuerde que la finalidad de utilizar una colección XML es almacenar o recuperar datos puros, sin códigos, en tablas de base de datos. Los datos contenidos en las tablas de base de datos existentes no tienen ninguna relación con los documentos XML de entrada; las operaciones de actualización, supresión y búsqueda utilizan el acceso normal mediante SQL a estas tablas. Si los documentos XML de entrada se descomponen en datos, los documentos XML originales dejan de existir.

El XML Extender permite realizar operaciones sobre los datos de una vista de colección XML. Mediante las sentencias UPDATE y DELETE de SQL, puede modificar los datos utilizados para componer documentos XML y, por tanto, actualizar la colección XML.

Consideraciones:

- Para actualizar un documento, no suprima una fila que contenga la clave primaria de la tabla, que es la fila de la clave foránea de las demás tablas de la colección. Cuando se suprime la fila de la clave primaria y de la clave foránea, se suprime el documento.
- Para sustituir o suprimir elementos o valores de atributos, puede suprimir e insertar filas en tablas de nivel inferior sin suprimir el documento.
- Para suprimir un documento, suprima la fila que forma el nodo_de_elemento superior especificado en el archivo DAD.

Actualización de datos en una colección XML

XML Extender le permite actualizar datos no marcados que están almacenados en tablas de una colección XML. Cuando actualiza los valores de las tablas de una colección XML, está actualizando el texto de un elemento XML o el valor de un atributo XML. Además, las actualizaciones pueden suprimir una instancia de datos en elementos o atributos de aparición múltiple.

Desde el punto de vista del SQL, el cambiar el valor de un elemento o atributo es una operación de actualización, y el suprimir una instancia de un elemento o atributo es una operación de supresión. Desde el punto de vista de XML, mientras exista el texto de elemento o valor de atributo correspondiente al nodo_de_elemento raíz, el documento XML sigue existiendo y es, por tanto, una operación de actualización.

Requisitos: Para actualizar datos en una colección XML, siga las reglas siguientes.

- Especifique la relación clave primaria-clave foránea para las tablas de la colección cuando las tablas existentes tengan esta relación. Si no tienen esta relación, asegúrese de que haya columnas que se puedan unir.
- Incluya la condición de unión que está especificada en el archivo DAD:
  - Para la correlación SQL, en el elemento <SQL_stmt>
  - Para la correlación de nodo_RDB, en el nodo_RDB del nodo de elemento raíz.

Actualización de valores de elementos y de atributos

En una colección XML, el texto de elementos y los valores de atributos están todos correlacionados con columnas de tablas de una base de datos. Con independencia de si los datos de las columnas ya existían previamente o proceden de la descomposición de documentos XML de entrada, puede sustituir los datos utilizando el método normal de actualización del SQL.

Para actualizar un valor de elemento o atributo, especifique una cláusula WHERE en la sentencia UPDATE de SQL donde reside la condición de unión que está especificada en el archivo DAD.

Por ejemplo:

```
UPDATE SHIP_TAB
    set MODE = 'BOAT'
  WHERE MODE='AIR' AND PART_KEY in
    (SELECT PART_KEY from PART_TAB WHERE ORDER_KEY=68)
```

El valor del elemento <ShipMode> se actualiza para cambiar de AIR a BOAT en la tabla SHIP_TAB, donde la clave es 68.

Supresión de instancias de elemento y de atributo

Puede actualizar documentos XML compuestos eliminando elementos o atributos de aparición múltiple; para ello suprima la fila donde reside el valor del elemento o atributo, utilizando la cláusula WHERE. Esta supresión se considera una actualización del documento XML mientras no suprima la fila que contiene los valores del elemento_de_nodo superior.

Por ejemplo, la siguiente sentencia DELETE suprime un elemento <shipment> especificando un valor exclusivo de uno de sus subelementos.

```
DELETE from SHIP_TAB
   WHERE DATE='1999-04-12'
```

Mediante la especificación de valor de fecha (DATE), se suprime la fila que coincide con ese valor. El documento compuesto original contenía dos elementos <shipment>, ahora contiene uno solamente.

Supresión de un documento XML de una colección XML

Puede suprimir un documento XML que se ha formado a partir de una colección. Esto significa que si tiene una colección XML a partir de la cual se crean varios documentos XML, puede suprimir uno de estos documentos compuestos.

Para suprimir el documento, suprima la fila de la tabla que sirve para formar el nodo_de_elemento superior que está especificado en el archivo DAD. Esta tabla contiene la clave primaria correspondiente a la tabla de colección de nivel superior, y la tabla foránea de las tablas de nivel inferior.

Por ejemplo, la siguiente sentencia DELETE especifica el valor de la columna de la clave primaria.

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

ORDER_KEY es la clave primaria de la tabla ORDER_TAB y es el elemento_de_nodo superior cuando se crea el documento XML. Cuando se suprime esta fila, se suprime un documento XML que se genera durante la composición. Por tanto, desde el punto de vista de XML, se suprime un documento XML individual de la colección XML.

Recuperación de documentos XML de una colección XML

Recuperar documentos XML de una colección es similar a componer documentos a partir de una colección.

Para recuperar documentos XML, utilice el procedimiento almacenado dxxRetrieveXML(). Consulte “dxxRetrieveXML()” en la página 228 para conocer la sintaxis y ver ejemplos.

Consideración referente al archivo DAD: Cuando descompone un documento XML y forma una colección XML, se puede perder el orden de los elementos y valores de atributos de aparición múltiple, a menos que especifique el orden en el archivo DAD. Para conservar este orden, debe utilizar el esquema de correlación de nodo_RDB. Este esquema de correlación le permite especificar el atributo ″orderBy″ para la tabla en cuyo nodo_RDB reside el elemento raíz.

Búsqueda en una colección XML

Esta sección describe la búsqueda en una colección con el fin de conseguir los siguientes objetivos:

- **Generar documentos XML utilizando criterios de búsqueda:**

  Esta tarea equivale en realidad a una composición utilizando una condición. Puede especificar los criterios de búsqueda siguientes:

  – Especifique la condición en el nodo_de_texto y nodo_de_atributo del archivo DAD.

  – Especifique el parámetro *overwrite* cuando utilice los procedimientos almacenados dxxGenXML() y dxxRetrieveXML().

    Por ejemplo, si habilitó la colección XML "sales_ord", utilizando el archivo DAD `order.dad`, pero ahora desea modificar el precio (price_value) utilizando datos obtenidos de la Web, puede modificar el valor del elemento <SQL_stmt> del archivo DAD, de esta manera:

```
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
       ...

EXEC SQL END DECLARE SECTION;

         float    price_value;

         /* crear tabla */
         EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

         /* inicializar variable de lenguaje principal e indicadores */
         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;

         /* obtener price_value de algún lugar, como de los datos */
         price_value = 1000,00         /* por ejemplo*/

         /* especificar la sobreescritura */
         sprintf(overwrite,
                "SELECT o.order_key, customer, p.part_key, quantity,
                       price, tax, ship_id, date, mode
         FROM order_tab o, part_tab p,
              table (select substr(char(timestamp(generate_unique())),16)
```

```
     as ship_id, date, mode from ship_tab)as s
 WHERE p.price > %d and s.date >'1996-06-01' AND
     p.order_key = o.order_key and s.part_key = p.part_key",
     price_value);

 /* Llamar al procedimiento almacenado */
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);
```

La condición de precio > 2500,00 en `order.dad` se ve alterada por la condición de precio > ?, donde ? se basa en la variable de entrada *price_value*.

- **Búsqueda de datos XML descompuestos:**

  Puede utilizar las operaciones normales de consulta SQL para hacer búsquedas en tablas de colección. Puede unir tablas de colección, o utilizar subconsultas, y luego realizar búsqueda estructurales en columnas de texto. Puede aplicar los resultados de la búsqueda estructural para recuperar o generar el documento XML especificado.

Parte 4. Referencia

Esta parte proporciona información de sintaxis sobre el mandato de administración del XML Extender, los tipos de datos definidos por el usuario (los UDT), las funciones definidas por el usuario (las UDF) y los procedimientos almacenados. También se proporciona texto de mensajes para realizar tareas de determinación de problemas.

Capítulo 7. El mandato de administración del XML Extender: dxxadm

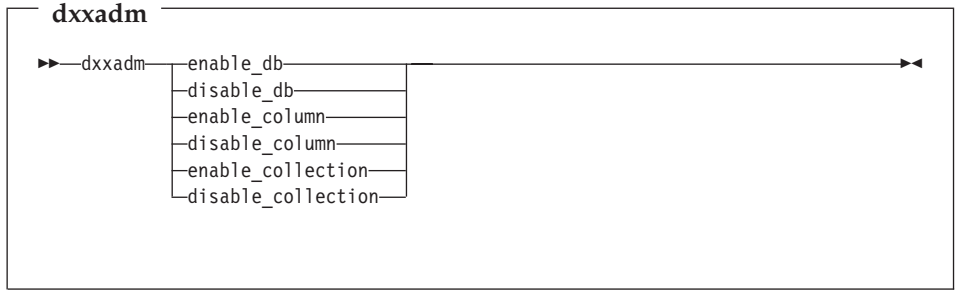
El XML Extender proporciona un mandato de administración, **dxxadm**, para llevar a cabo las siguientes tareas de admnistración.Las siguientes tareas de administración del XML Extender se llevan a cabo invocando el mandato **dxxadm** y utilizando varias opciones de mandato:

- Habilitar o inhabilitar una base de datos para el XML Extender
- Habilitar o inhabilitar una columna XML
- Habilitar o inhabilitar una colección XML

Puede también utilizar el asistente de administración o los procedimientos almacenados del XML Extender para realizar cada una de las tareas de administración.

Sintaxis de nivel superior

El diagrama de sintaxis siguiente proporciona la sintaxis de nivel superior del mandato **dxxadm**. Cada opción del diagrama aparece descrita en las secciones que siguen a continuación.



Opciones del mandato de administración

Las secciones siguientes describen cada una de las opciones del mandato **dxxadm** que están disponibles para los programadores de sistemas:

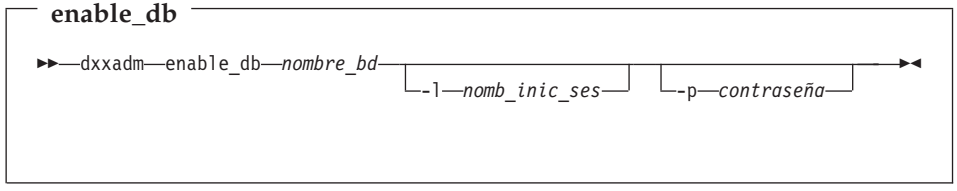
enable_db

Propósito

Conecta y habilita una base de datos para que pueda ser utilizada con el XML Extender. Cuando se habilita la base de datos, el XML Extender crea los objetos siguientes:

- Los tipos definidos por el usuario (los UDT) del XML Extender.
- Las funciones definidas por el usuario (las UDF) del XML Extender
- La tabla de referencia DTD del XML Extender, DTD_REF, donde se almacenan las DTD e información sobre cada DTD. Para obtener una descripción completa de la tabla DTD_REF, vea "Tabla de referencia DTD" en la página 237.
- La tabla de utilización del XML Extender, XML_USAGE, que almacena información común para cada columna habilitada para XML y para cada colección. Para obtener una descripción completa de la tabla XML_USAGE, vea "Tabla de utilización de XML" en la página 238.

Formato



Parámetros

Tabla 14. Parámetros de enable_db

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos donde residen los datos XML.
-l *nomb_inic_ses*	Es el ID usuario (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplos

El ejemplo siguiente habilita la base de datos SALES_DB.

```
dxxadm enable_db SALES_DB
```

disable_db

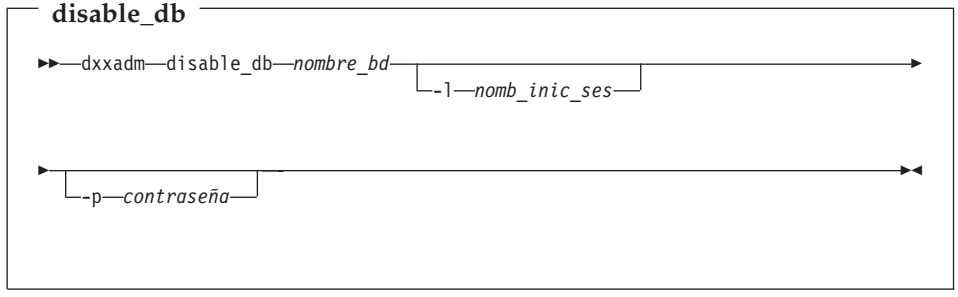
Propósito

Conecta y desconecta la base de datos habilitada para XML. Cuando la base de datos está inhabilitada, ya no puede ser utilizada por el XML Extender. Cuando el XML Extender inhabilita la base de datos, elimina los objetos siguientes:

- Los tipos definidos por el usuario (los UDT) del XML Extender.
- Las funciones definidas por el usuario (las UDF) del XML Extender
- La tabla de referencia DTD del XML Extender, DTD_REF, donde se almacenan las DTD e información sobre cada DTD. Para obtener una descripción completa de la tabla DTD_REF, vea "Tabla de referencia DTD" en la página 237.
- La tabla de utilización del XML Extender, XML_USAGE, que almacena información común para cada columna habilitada para XML y para cada colección. Para obtener una descripción completa de la tabla XML_USAGE, vea "Tabla de utilización de XML" en la página 238.

Importante: Debe inhabilitar todas las columnas XML antes de intentar inhabilitar una base de datos. El XML Extender no puede inhabilitar una base de datos que contenga columnas o colecciones habilitadas para XML.

Formato



Parámetros

Tabla 15. Parámetros de disable_db

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos donde residen los datos XML.

Tabla 15. Parámetros de disable_db (continuación)

Parámetro	Descripción
-l *nomb_inic_ses*	Es el ID usuario (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplos

El ejemplo siguiente inhabilita la base de datos SALES_DB.

```
dxxadm disable_db SALES_DB
```

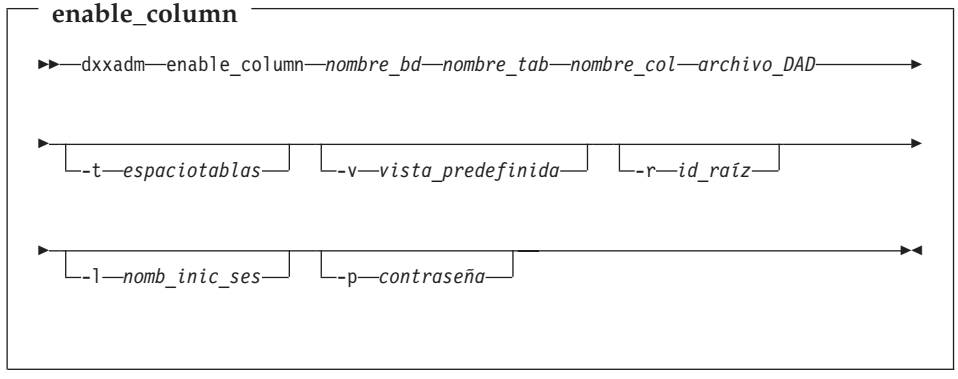
enable_column

Propósito

Conecta a una base de datos y habilita una columna XML para que pueda contener los UDT del XML Extender. Cuando habilita una columna, el XML Extender realiza estas tareas:

- Determina si la tabla XML tiene una clave primaria; si no la tiene, el XML Extender modifica la tabla XML y añade una columna llamada DXXROOT_ID.
- Crea tablas auxiliares, que se especifican en el archivo DAD, con una columna que contiene un identificador exclusivo para cada fila de la tabla XML. Esta columna es el *id_raíz* que el usuario especificó o el valor DXXROOT_ID proporcionado por el XML Extender.
- Crea una vista predefinida para la tabla XML y sus tablas auxiliares, utilizando opcionalmente un nombre especificado por el usuario.

Formato



Parámetros

Tabla 16. Parámetros de enable_column

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos donde residen los datos XML.
nombre_tab	Es el nombre de la tabla donde reside la columna XML.
nombre_col	Es el nombre de la columna XML.
archivo_DAD	Es el nombre del archivo DAD que correlaciona el documento XML con la columna XML y las tablas auxiliares.

Tabla 16. Parámetros de enable_column (continuación)

Parámetro	Descripción
-t *espaciotablas*	Es el espacio de tablas (opcional) donde residen las tablas auxiliares asociadas a la columna XML. Si no se especifica este parámetro, se utiliza el espacio de tablas predefinido.
-v *vista_predefinida*	Es el nombre de la vista predefinida (opcional) que asocia la columna XML y las tablas auxiliares.
-r *id_raíz*	Es el nombre de la clave primaria, contenida en la tabla de columnas XML, que se utilizará como id_raíz para las tablas auxiliares. El id_raíz es opcional.
-l *nomb_inic_ses*	Es el ID usuario (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplos

El ejemplo siguiente habilita una columna XML.

```
dxxadm enable_column SALES_DB SALES_TAB ORDER
       -v sales_order_view -r INVOICE_NUMBER
```

disable_column

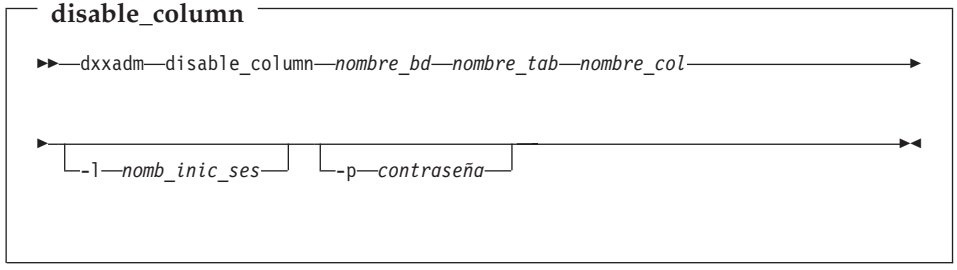
Propósito

Conecta a una base de datos e inhabilita la columna habilitada para XML. Cuando la columna está inhabilitada, ya no puede contener tipos de datos XML. Cuando se inhabilita una columna habilitada para XML, se producen las acciones siguientes:

- Se suprime la entrada de la tabla XML_USAGE correspondiente a la utilización de la columna XML.
- Disminuye el valor USAGE_COUNT en la tabla DTD_REF.
- Se eliminan todos los desencadenantes asociados a la columna.
- Se eliminan todas las tablas auxiliares asociadas a la columna.

Importante: Debe inhabilitar una columna XML antes de eliminar una tabla XML. Si se elimina una tabla XML, pero no se inhabilita su columna XML, el XML Extender conserva las tablas auxiliares que creó y también la entrada de la tabla XML_USAGE correspondiente a la columna XML.

Formato



Parámetros

Tabla 17. Parámetros de disable_column

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos donde residen los datos.
nombre_tab	Es el nombre de la tabla donde reside la columna XML.
nombre_col	Es el nombre de la columna XML.
-l *nomb_inic_ses*	Es el ID usuario (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza el ID de usuario actual.

Tabla 17. Parámetros de disable_column (continuación)

Parámetro	Descripción
-p *contraseña*	Es la contraseña (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplos

El ejemplo siguiente inhabilita una columna habilitada para XML.

```
dxxadm disable_column SALES_DB SALES_TAB ORDER
```

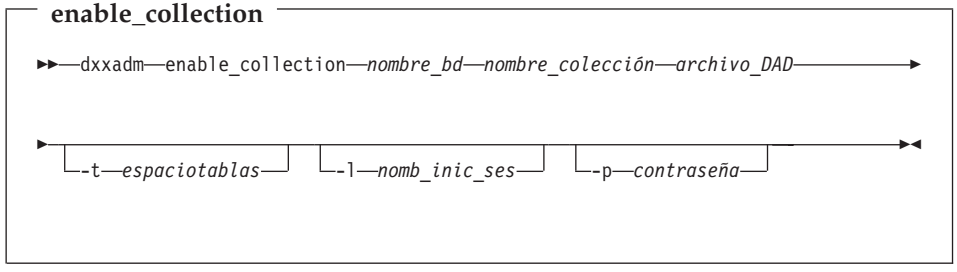
enable_collection

Propósito

Conecta a una base de datos y habilita una colección XML de acuerdo con la DAD especificada. Cuando habilita una colección, el XML Extender realiza estas tareas:

- Crea una entrada en la tabla XML_USAGE para la colección XML.
- Para la correlación de nodo_RDB, crea tablas de colección especificadas en la DAD, si las tablas no existen en la base de datos.

Formato



Parámetros

Tabla 18. Parámetros de enable_collection

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos donde residen los datos.
nombre_colección	Es el nombre de la colección XML.
archivo_DAD	Es el nombre del archivo DAD que correlaciona el documento XML con las tablas relacionales de la colección.
-t *espaciotablas*	Es el nombre del espacio de tablas (opcional) que está asociado a la colección. Si no se especifica este parámetro, se utiliza el espacio de tablas predefinido.
-l *nomb_inic_ses*	Es el ID usuario (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza el ID de usuario actual.

Tabla 18. Parámetros de enable_collection (continuación)

Parámetro	Descripción
-p *contraseña*	Es la contraseña (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplos

El ejemplo siguiente habilita una colección XML.

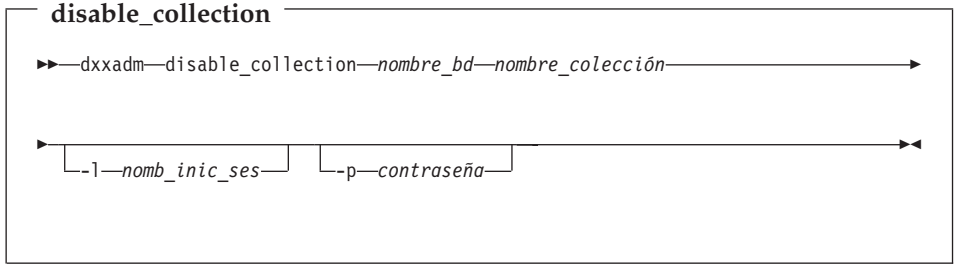
```
dxxadm enable_collection SALES_DB sales_ord
             getstart_xcollection.dad -t orderspace
```

disable_collection

Propósito

Conecta a una base de datos e inhabilita la colección habilitada para XML. Cuando se inhabilita una colección XML, el nombre de la colección ya no puede utilizarse en los procedimientos almacenados de composición (dxxRetrieveXML) y descomposición (dxxInsertXML) de documentos. Se suprime la entrada de la tabla XML_USAGE correspondiente a la colección. Observe que el inhabilitar la colección no elimina las tablas de colección que se crean durante el paso enable_collection.

Formato



Parámetros

Tabla 19. Parámetros de disable_collection

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos donde residen los datos.
nombre_colección	Es el nombre de la colección XML.
-l *nomb_inic_ses*	Es el ID usuario (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña (opcional) que se utiliza para conectarse a la base de datos, si se especifica. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplos

El ejemplo siguiente inhabilita una colección XML.

```
dxxadm disable_collection SALES_DB sales_ord
```

Capítulo 8. Tipos definidos por el usuario del XML Extender

Los tipos definidos por el usuario (los UDT) del XML Extender son tipos de datos que se utilizan para columnas XML y colecciones XML. Todos los UDT tienen el nombre de esquema db2xml. El XML Extender crea los UDT para almacenar y recuperar documentos XML. La Tabla 20 muestra una visión general de los UDT.

Tabla 20. Los UDT del XML Extender

Columna de tipo definido por el usuario	Tipo de datos fuente	Descripción
XMLVARCHAR	VARCHAR(*long_varchar*)	Almacena un documento XML completo, en forma de VARCHAR, dentro de DB2.
XMLCLOB	CLOB(*long_clob*)	Almacena un documento XML completo, en forma de gran objeto de caracteres (CLOB), dentro de DB2.
XMLFILE	VARCHAR(512)	Especifica el nombre de archivo del servidor de archivos local. Si se especifica XMLFILE para la columna XML, el XML Extender guarda el documento XML en un archivo de servidor externo. El Text Extender no se puede habilitar mediante XMLFILE. Corresponde al usuario asegurar la integridad entre el contenido del archivo y DB2, así como la tabla auxiliar creada para indexar.

long_varchar y *long_clob* son específicos del sistema operativo.

Para DB2 UDB, *long_varchar* = 3K y *long_clob* = 2G.

Estos UDT sólo se utilizan para especificar los tipos de columnas de aplicación; no son aplicables a las tablas auxiliares que crea el XML Extender.

Capítulo 9. Funciones definidas por el usuario del XML Extender

El XML Extender proporciona funciones para almacenar, recuperar, buscar y actualizar documentos XML, y para extraer elementos o atributos XML. Las funciones definidas por el usuario (las UDF) de XML se pueden utilizar para columnas XML, pero no para colecciones XML. Todas las UDF tienen el nombre de esquema db2xml, que puede omitirse como prefijo de las UDF.

Las funciones del XML Extender pueden ser de cuatro tipos: funciones de almacenamiento, recuperación, extracción y actualización.

funciones de almacenamiento
Las funciones de almacenamiento insertan documentos XML en una base de datos DB2. Para conocer la sintaxis y ver ejemplos, consulte "Funciones de almacenamiento" en la página 176.

funciones de recuperación
Las funciones de recuperación recuperan documentos XML a partir de columnas XML de una base de datos DB2. Para conocer la sintaxis y ver ejemplos, consulte "Funciones de recuperación" en la página 182.

funciones de extracción
Las funciones de extracción extraen el contenido de un elemento o el valor de un atributo en un documento XML y lo convierten al tipo de datos especificado por el nombre de la función. El XML Extender proporciona un juego de funciones de extracción para diversos tipos de datos de SQL. Para conocer la sintaxis y ver ejemplos, consulte "Funciones de extracción" en la página 189.

función de actualización
La función Update() modifica el contenido de un elemento o el valor de un atributo y devuelve una copia de un documento XML con un valor actualizado que está especificado por la vía de ubicación. La función Update() permite al programador de aplicaciones especificar el elemento o atributo que se debe actualizar. Para conocer la sintaxis y ver ejemplos, consulte "Función de actualización" en la página 206.

La Tabla 21 en la página 176 proporciona un resumen de las funciones del XML Extender.

Tabla 21. Funciones definidas por el usuario del XML Extender

Tipo	Función
Funciones de almacenamiento	XMLVarcharFromFile()
	XMLCLOBFromFile()
	XMLFileFromVarchar()
	XMLFileFromCLOB()
Funciones de recuperación	Content(): recuperar de XMLFile y almacenar en CLOB
	Content(): recuperar de XMLVarchar y almacenar en un archivo de servidor externo
	Content(): recuperar de XMLCLOB y almacenar en archivo de servidor externo
Funciones de extracción	extractInteger() y extractIntegers()
	extractSmallint() y extractSmallints()
	extractDouble() y extractDoubles()
	extractReal() y extractReals()
	extractChar() y extractChars()
	extractVarchar() y extractVarchars()
	extractCLOB() y extractCLOBs()
	extractDate() y extractDates()
	extractTime() y extractTimes()
	extractTimestamp() y extractTimestamps()
Función de actualización	Update()

Cuando se utilizan marcadores de parámetros en las UDF, una restricción de JDBC requiere que el marcador de parámetros de la UDF se convierta al tipo de datos de la columna en la que se insertarán los datos devueltos. Para aprender a convertir los marcadores de parámetros, consulte el apartado "Limitaciones al invocar funciones desde JDBC" en la página 138.

Funciones de almacenamiento

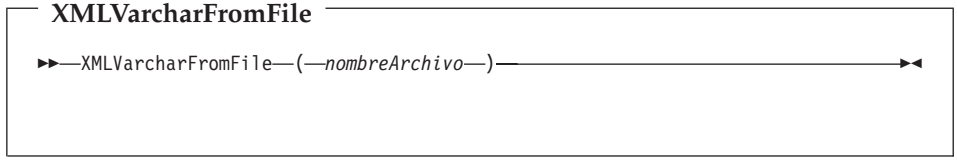
Utilice funciones de almacenamiento para insertar documentos XML en una base de datos DB2. Puede utilizar las funciones predefinidas de conversión de datos de un UDT directamente en las sentencias INSERT o SELECT, tal como se describe en el apartado "Almacenamiento de datos" en la página 122. Además, XML Extender proporciona a las UDF acceso a documentos XML de orígenes distintos del tipo de datos base UDT y los convierte al UDT deseado.

El XML Extender proporciona las funciones de almacenamiento siguientes:

XMLVarcharFromFile()

Propósito
Lee un documento XML de un archivo de servidor y devuelve el documento como un tipo XMLVARCHAR.

Sintaxis



Parámetros

Tabla 22. Parámetro de XMLVarcharFromFile

Parámetro	Tipo de datos	Descripción
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.

Tipo de devolución
XMLVARCHAR

Ejemplo
El ejemplo siguiente lee un documento XML de un archivo de servidor y lo inserta en una columna XML como un tipo XMLVARCHAR.

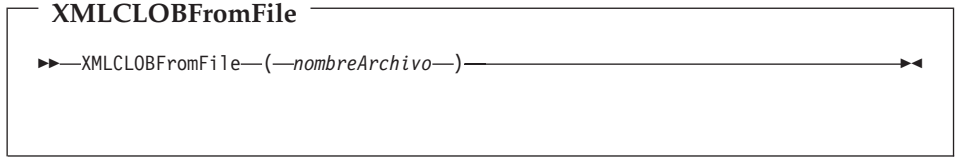
```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

Este ejemplo inserta un registro en la tabla SALES_TAB. La función XMLVarcharFromFile() importa el documento XML de un archivo a DB2 y lo guarda en forma de XMLVARCHAR.

XMLCLOBFromFile()

Propósito
Lee un documento XML de un archivo de servidor y devuelve el documento como un tipo XMLCLOB.

Sintaxis



Parámetros

Tabla 23. Parámetro de XMLCLOBFromFile

Parámetro	Tipo de datos	Descripción
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.

Tipo de devolución
XMLCLOB como LOCATOR

Ejemplo
El ejemplo siguiente lee un documento XML de un archivo de servidor y lo inserta en una columna XML como un tipo XMLCLOB.

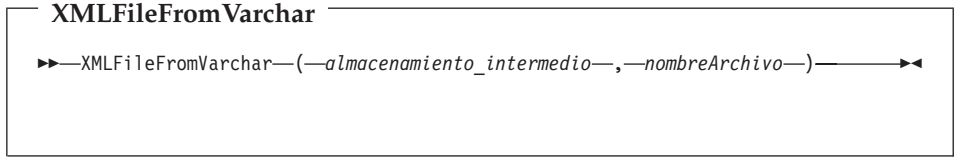
```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

La columna ORDER de la tabla SALES_TAB se ha definido con el tipo XMLCLOB. El ejemplo anterior muestra cómo se inserta la columna ORDER en la tabla SALES_TAB.

XMLFileFromVarchar()

Propósito
Lee un documento XML de la memoria como VARCHAR, lo escribe en un archivo de servidor externo y devuelve el nombre de archivo y la vía de acceso como un tipo XMLFILE.

Sintaxis



Parámetros

Tabla 24. Parámetros de XMLFileFromVarchar

Parámetro	Tipo de datos	Descripción
almacenamiento_intermedio	VARCHAR(3K)	Es la memoria intermedia.
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.

Tipo de devolución
XMLFILE

Ejemplo
En los ejemplos siguientes se lee un documento XML de la memoria como VARCHAR, se graba en un archivo de servidor externo y se inserta el nombre de archivo y la vía de acceso como un tipo XMLFILE en una columna XML.

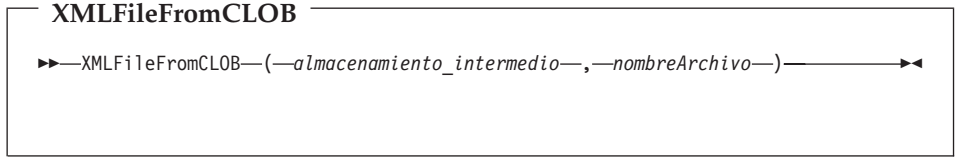
```
EXEC SQL BEGIN DECLARE SECTION;
      struct { short len; char data[3000]; } xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
  VALUES('1234', 'Sriram Srinivasan',
       XMLFileFromVarchar(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

La columna ORDER de la tabla SALES_TAB se ha definido con el tipo XMLFILE. El ejemplo anterior muestra cómo almacenar en un archivo de servidor un documento XML contenido en almacenamiento intermedio.

XMLFileFromCLOB()

Propósito
Lee un documento XML como un localizador de CLOB, lo escribe en un archivo de servidor externo y devuelve el nombre de archivo y la vía de acceso como un tipo XMLFILE.

Sintaxis



Parámetros

Tabla 25. Parámetros de XMLFileFromCLOB()

Parámetros	Tipo de datos	Descripción
almacenamiento_intermedio	CLOB como LOCATOR	Es el almacenamiento intermedio donde reside el documento XML.
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.

Tipo de devolución
XMLFILE

Ejemplo
El ejemplo siguiente lee un documento XML como un localizador de CLOB, lo escribe en un archivo de servidor externo e inserta el nombre de archivo y la vía de acceso como un tipo XMLFILE en una columna XML.

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLFileFromCLOB(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

La columna ORDER de la tabla SALES_TAB se ha definido con el tipo XMLFILE. Si tiene un documento XML en el almacenamiento intermedio, puede almacenarlo en un archivo de servidor.

Funciones de recuperación

Puede utilizar las funciones predefinidas de conversión de datos para convertir un UDT de XML en el tipo de datos base según se describe en el apartado “Recuperación de un documento completo” en la página 125. El XML Extender también proporciona una *función sobrecargada* Content(), que se utiliza para la recuperación de datos.

La función Content() proporciona los tipos siguientes de recuperación de datos:

- **Recuperación desde almacenamiento externo en el servidor a una variable del sistema principal en el cliente.**

  Puede utilizar Content() para recuperar un documento XML, contenido en un archivo de servidor externo, y colocarlo en una memoria intermedia.. Puede utilizar “Content(): recuperar de XMLFILE y almacenar en CLOB” en la página 183 con este fin.
- **Recuperación desde almacenamiento interno a un archivo de servidor externo**

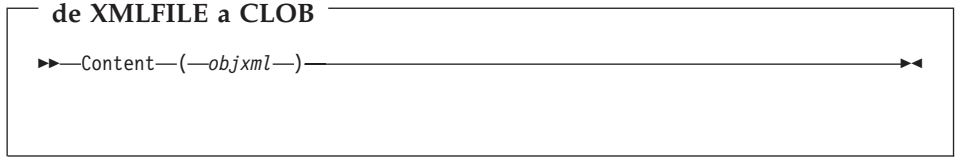
  Puede también utilizar Content() para recuperar un documento XML, contenido dentro de DB2, y almacenarlo en un archivo del sistema de archivos del servidor de DB2. Las funciones Content() siguientes se utilizan para almacenar información en archivos de servidor externos:
  - “Content(): recuperar de XMLVARCHAR y almacenar en archivo de servidor externo” en la página 185
  - “Content(): recuperar de XMLCLOB y almacenar en archivo de servidor externo” en la página 187

Los ejemplos de la sección siguiente dan por supuesto que el usuario está utilizando el shell de mandatos de DB2, en el que no es necesario escribir “DB2” al principio de cada mandato.

Content(): recuperar de XMLFILE y almacenar en CLOB

Propósito
Recupera datos de un archivo de servidor y los almacena en un localizador de CLOB.

Sintaxis



Parámetros

Tabla 26. Parámetro de XMLFILE a CLOB

Parámetro	Tipo de datos	Descripción
objxml	XMLFILE	Es el documento XML.

Tipo de devolución
CLOB (*long_clob*) como LOCATOR

long_clob es 2G para DB2 UDB.

Ejemplo
El ejemplo siguiente recupera datos de un archivo de servidor y los almacena en un localizador de CLOB.

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT TO SALES_DB

EXEC SQL DECLARE c1 CURSOR FOR

     SELECT Content(order) from sales_tab
     WHERE sales_person = 'Sriram Srinivasan'

EXEC SQL OPEN c1;

do {
   EXEC SQL FETCH c1 INTO :xml_buff;
   if (SQLCODE != 0) {
      break;
      }
   else {
     /* do with the XML doc in buffer */
```

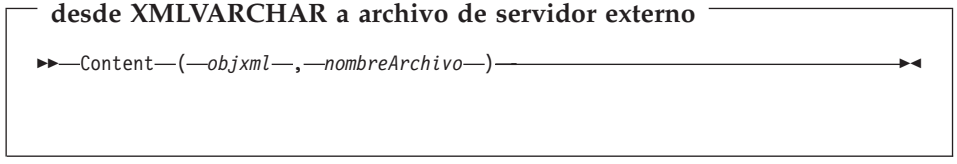
```
    }
  }

EXEC SQL CLOSE c1;

EXEC SQL CONNECT RESET;
```

La columna ORDER de la tabla SALES_TAB es de tipo XMLFILE, por tanto la UDF Content() recupera datos de un archivo de servidor y los almacena en un localizador de CLOB.

Content(): recuperar de XMLVARCHAR y almacenar en archivo de servidor externo

Propósito
Recupera un documento XML de tipo XMLVARCHAR y lo almacena en un archivo de servidor externo.

Sintaxis



Importante: Si ya existe un archivo con el nombre especificado, la función Content() escribirá encima de él.

Nota:

Parámetros

Tabla 27. Parámetros de la recuperación XMLVarchar a un archivo de servidor externo

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR	Es el documento XML.
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.

Tipo de devolución
VARCHAR(512)

Ejemplo
El ejemplo siguiente recupera un documento XML de tipo XMLVARCHAR y lo almacena en un archivo de servidor externo.

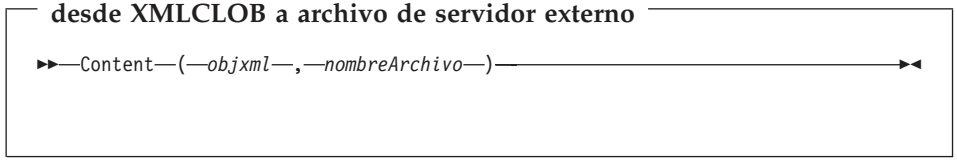
```
CREATE table app1 (id int NOT NULL, order db2xml.XMLVarchar);
INSERT into app1 values (1, '<?xml version="1.0"?>
   <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
      <Customer>
          <Name>American Motors</Name>
          <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
          <key>68</key>
```

```
            <Quantity>36</Quantity>
            <ExtendedPrice>34850.16</ExtendedPrice>
            <Tax>6.000000e-02</Tax>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>AIR </ShipMode>
            </Shipment>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>BOAT </ShipMode>
            </Shipment>
        </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart_.dad')
  from app1 where ID=1;
```

Content(): recuperar de XMLCLOB y almacenar en archivo de servidor externo

Propósito
Recupera un documento XML de tipo XMLCLOB y lo almacena en un archivo de servidor externo.

Sintaxis



Importante: Si ya existe un archivo con el nombre especificado, la función Content() escribirá encima de él.

Parámetros

Tabla 28. Parámetros de la recuperación desde XMLCLOB a un archivo de servidor externo

Parámetro	Tipo de datos	Descripción
objxml	XMLCLOB como LOCATOR	Es el documento XML.
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.

Tipo de devolución
VARCHAR(512)

Ejemplo
El ejemplo siguiente recupera un documento XML de tipo XMLCLOB y lo almacena en un archivo de servidor externo.

```
CREATE table app1 (id int NOT NULL, order db2xml.XMLCLOB not logged);

INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
      <Customer>
          <Name>American Motors</Name>
          <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
          <key>68</key>
```

```
            <Quantity>36</Quantity>
            <ExtendedPrice>34850.16</ExtendedPrice>
            <Tax>6.000000e-02</Tax>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>AIR </ShipMode>
            </Shipment>
            <Shipment>
                <ShipDate>1998-08-19</ShipDate>
                <ShipMode>BOAT </ShipMode>
            </Shipment>
        </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart.xml')
from app1 where ID=1;
```

Funciones de extracción

Las funciones de extracción extraen el contenido de un elemento o el valor de un atributo en un documento XML y devuelven los tipos de datos SQL especificados. El XML Extender proporciona un juego de funciones de extracción para diversos tipos de datos de SQL. Las funciones de extracción utilizan dos parámetros de entrada. El primer parámetro es el UDT del XML Extender, que puede ser uno de los UDT de XML. El segundo parámetro es la vía de ubicación, que especifica el elemento o atributo de XML. Cada función de extracción devuelve el valor o contenido especificado por la vía de ubicación.

Debido a que algunos valores de elemento o atributo aparecen varias veces, las funciones de extracción devuelven un valor escalar o un valor de tabla; este último se denomina función de tabla.

El XML Extender proporciona las funciones de extracción siguientes:

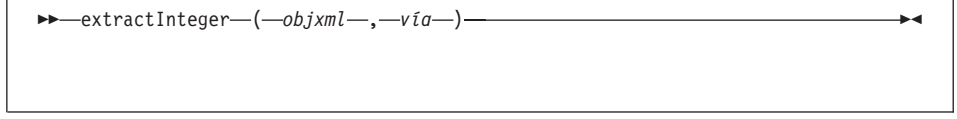
Los ejemplos de la sección siguiente dan por supuesto que el usuario está utilizando el shell de mandatos de DB2, en el que no es necesario escribir "DB2" al principio de cada mandato.

extractInteger() y extractIntegers()

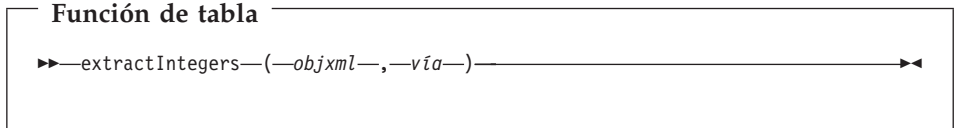
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo INTEGER.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 29. Parámetros de las funciones extractInteger y extractIntegers

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
INTEGER

Nombre de columna devuelta (función de tabla)
returnedInteger

Ejemplo
Ejemplo de función escalar:

En el ejemplo siguiente, se devuelve un valor cuando el valor de atributo de clave = ″1″. El valor se extrae como un entero convertido automáticamente a un tipo DECIMAL.

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

Ejemplo de función de tabla:

En el siguiente ejemplo, cada valor de clave correspondiente al pedido de ventas se extrae como un INTEGER

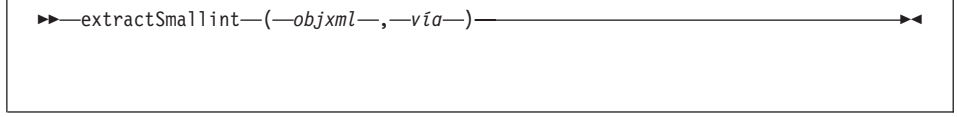
```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

extractSmallint() y extractSmallints()

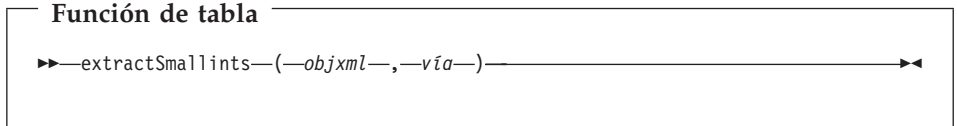
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo SMALLINT.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 30. Parámetros de las funciones extractSmallint y extractSmallints

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
SMALLINT

Nombre de columna devuelta (función de tabla)
returnedSmallint

Ejemplo
En el ejemplo siguiente, el valor de Quantity en todos los pedidos de ventas se extrae como SMALLINT

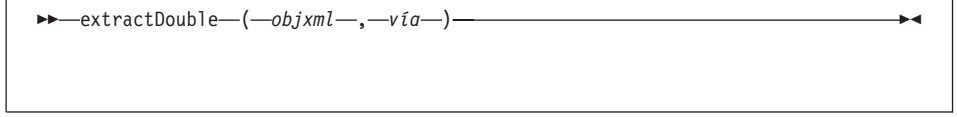
```
SELECT * from table(db2xml.extractSmallints(db2xml.File
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Quantity')) as x;
```

extractDouble() y extractDoubles()

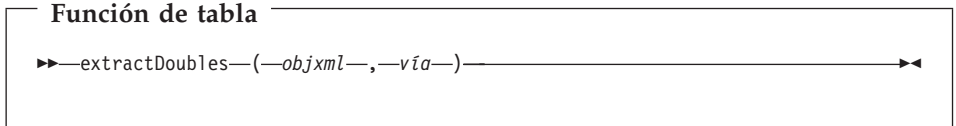
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo DOUBLE.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 31. Parámetros de las funciones extractDouble y extractDoubles

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
DOUBLE

Nombre de columna devuelta (función de tabla)
returnedDouble

Ejemplo
Ejemplo de función escalar:

El ejemplo siguiente convierte automáticamente el precio de un pedido de un tipo DOUBLE a a DECIMAL.

```
CREATE TABLE t1(price, DECIMAL(5,2));
INSERT into t1 values (db2xml.extractDouble(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'),
    '/Order/Part[@color="black "]/ExtendedPrice'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En el ejemplo siguiente, el valor de ExtendedPrice en cada parte del pedido de ventas se extrae como DOUBLE.

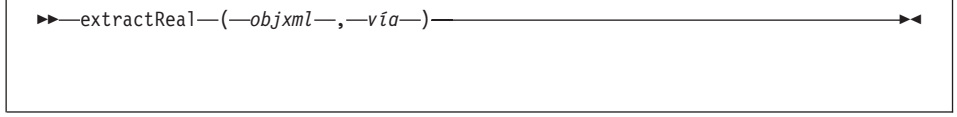
```
SELECT * from table(db2xml.extractDoubles(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/ExtendedPrice')) as x;
```

extractReal() y extractReals()

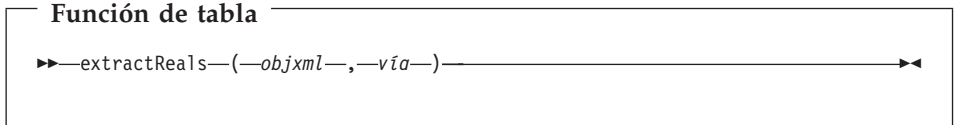
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo REAL.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 32. Parámetros de las funciones extractReal y extractReals

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
REAL

Nombre de columna devuelta (función de tabla)
returnedReal

Ejemplo
En el ejemplo siguiente, cada valor de Tax se extrae como REAL.

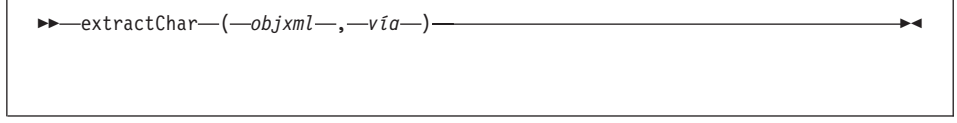
```
SELECT * from table(db2xml.extractReals(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Tax')) as x;
```

extractChar() y extractChars()

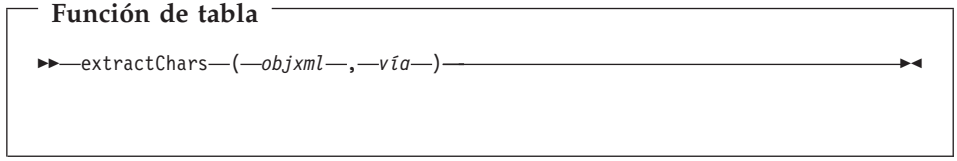
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo CHAR.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 33. Parámetros de las funciones extractChar y extractChars

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
CHAR

Nombre de columna devuelta (función de tabla)
returnedChar

Ejemplo
En el ejemplo siguiente, el valor de Color se extrae como CHAR.

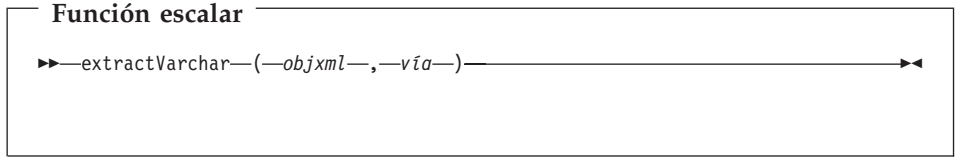
```
SELECT * from table(db2xml.extractChars(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/@Color')) as x;
```

extractVarchar() y extractVarchars()

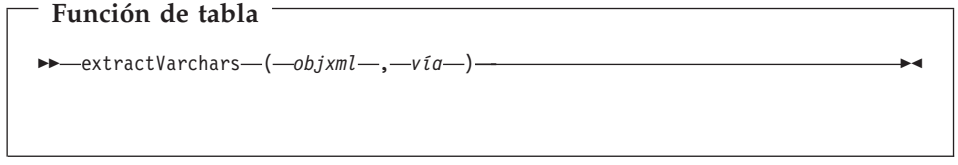
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo VARCHAR.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 34. Parámetros de las funciones extractVarchar y extractVarchars

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
VARCHAR(4K)

Nombre de columna devuelta (función de tabla)
returnedVarchar

Ejemplo
Suponga una base de datos que contiene más de 1000 documentos XML, los cuales están almacenados en la columna ORDER de la tabla SALES_TAB. Desea encontrar los clientes que han solicitado artículos cuyo precio global (ExtendedPrice) sea mayor que 2500,00. La siguiente sentencia de SQL utiliza la UDF de extracción en la cláusula SELECT:

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

La UDF extractVarchar() utiliza la columna ORDER como entrada y la vía de ubicación `/Order/Customer/Name` como identificador de selección. La UDF devuelve los nombres de los clientes. Mediante la cláusula WHERE, la función de extracción evalúa sólo los pedidos cuyo precio global sea mayor que 2500,00.

extractCLOB() y extractCLOBs()

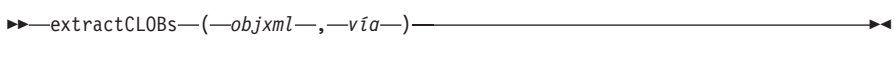
Propósito

Extrae un fragmento de documentos XML, con markup para el elemento y el atributo, el contenido de los elementos y los atributos, incluidos los subelementos. Esta función es diferente de las demás funciones de extracción; éstas sólo devuelven el contenido de los elementos y los atributos. Las funciones extractClob(s) sólo se deben utilizar para extraer fragmentos de documentos, mientras que extractVarchar(s) y extractChar(s) se deben utilizar para extraer valores simples.

Sintaxis

Función escalar

Función de tabla



Parámetros

Tabla 35. Parámetros de las funciones extractCLOB y extractCLOBs

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución

CLOB(10K)

Nombre de columna devuelta (función de tabla)

returnedCLOB

Ejemplo

En este ejemplo, todos los elementos de la parte se extraen de un pedido de compra.

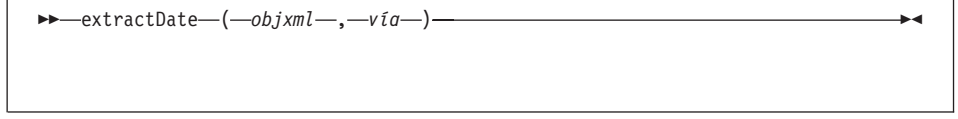
```
SELECT returnedCLOB as part
   from table(db2xml.extractCLOBs(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part')) as x;
```

extractDate() y extractDates()

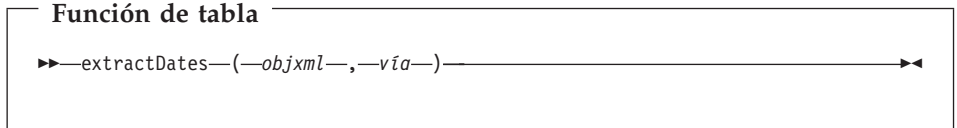
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo DATE.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 36. Parámetros de las funciones extractDate y extractDates

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
DATE

Nombre de columna devuelta (función de tabla)
returnedDate

Ejemplo
En el ejemplo siguiente, el valor de ShipDate se extrae como DATE.

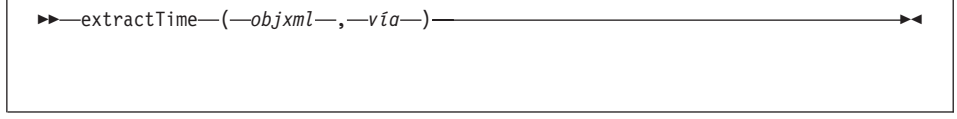
```
SELECT * from table(db2xml.extractDates(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Shipment/ShipDate')) as x;
```

extractTime() y extractTimes()

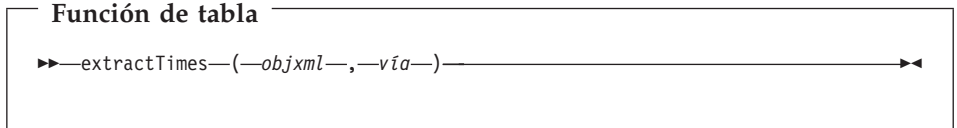
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo TIME.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 37. Parámetros de las funciones extractTime y extractTimes

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
TIME

Nombre de columna devuelta (función de tabla)
returnedTime

Ejemplo
Este ejemplo utiliza los archivos de ejemplo de libros. Busca en el archivo XML `book.xml` la hora en que se ha puesto precio a los libros y devuelve los valores como TIME.

Archivo XML:

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13"
     timestamp="1998-12-22-11.12.13.888888">
38.281
</price>
</book>
```

Ejemplo de extracción:

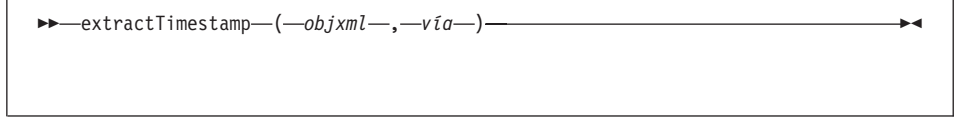
```
CREATE TABLE t1(SaleTime TIME);
INSERT INTO t1 values(db2xml.extractTime(doc, '/book/price/@time'));
SELECT * from t1;
```

extractTimestamp() y extractTimestamps()

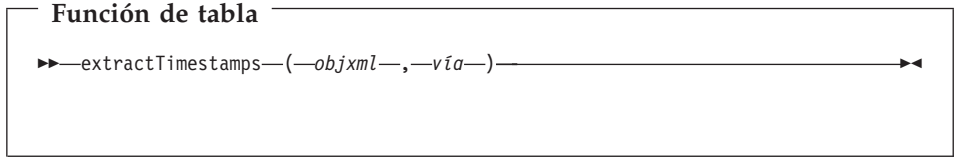
Propósito
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo TIMESTAMP.

Sintaxis

Función escalar



Función de tabla



Parámetros

Tabla 38. Parámetros de las funciones extractTimestamp y extractTimestamps

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLFILE o XMLCLOB	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.

Tipo de devolución
TIMESTAMP

Nombre de columna devuelta (función de tabla)
returnedTimestamp

Ejemplo
Este ejemplo utiliza los archivos de ejemplo de libros. Busca en el archivo XML `book.xml` la hora que especifica cuándo se ha puesto precio a cada libro y extrae el valor como TIMESTAMP.

Archivo XML:

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13"
        timestamp="1998-12-22-11.12.13.888888"> 38.281
</price>
</book>
```

Ejemplo de extracción:

```
CREATE TABLE t1(SaleTime TIMESTAMP);
INSERT INTO t1 values(db2xml.extractTimestamp
       (doc, '/book/price/@timestamp'));
SELECT * from t1;
```

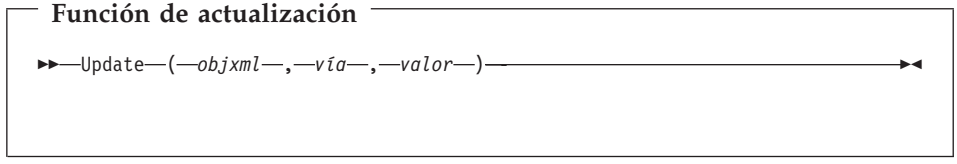
Función de actualización

La función Update() actualiza un valor de elemento o atributo especificado en uno o más documentos XML almacenados en la columna XML. También puede utilizar las funciones predefinidas de conversión de datos para convertir un tipo base de SQL a UDT de XML, según se describe en el apartado "Actualización de datos XML" en la página 130.

Propósito

Utiliza como entrada el nombre de columna de un UDT de XML, una vía de ubicación y una serie de caracteres representativa del valor a actualizar, y devuelve un UDT de XML que es igual al primer parámetro de entrada. Mediante la función Update(), puede especificar el elemento o atributo que se debe actualizar.

Sintaxis



Parámetros

Tabla 39. Parámetros de la UDF Update

Parámetro	Tipo de datos	Descripción
objxml	XMLVARCHAR, XMLCLOB como LOCATOR	Es el nombre de la columna.
vía	VARCHAR	Es la vía de ubicación del elemento o atributo.
valor	VARCHAR	Es la serie de caracteres a actualizar.

Importante: Tenga presente que la UDF Update da soporte a las vías de ubicación que tienen predicados con atributos, pero no elementos. Por ejemplo, se da soporte al siguiente predicado:

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

El siguiente predicado no está soportado:

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

Tipo devuelto

Tipo de datos	Tipo devuelto
XMLVARCHAR	XMLVARCHAR
XMLCLOB como LOCATOR	XMLCLOB

Ejemplo

En el ejemplo siguiente se actualiza el pedido de compra manejado por el vendedor Sriram Srinivasan.

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

En este ejemplo, el contenido de `/Order/Customer/Name` se actualiza para que sea `IBM`.

Uso

Cuando se utiliza la función Update para cambiar un valor en uno o más documentos XML, en realidad esta función sustituye los documentos XML dentro de la columna XML. Basándose en la salida procedente del analizador XML, se conservan algunas partes del documento original, otras se pierden o se cambian. Los siguientes apartados describen cómo se procesa el documento y ofrecen ejemplos de qué aspecto tienen los documentos antes y después de las actualizaciones.

Cómo la función Update procesa el documento XML

Cuando la función Update sustituye los documentos XML, debe reconstruir el documento basándose en la salida del analizador de XML. La Tabla 40 en la página 208 describe cómo se manejan las partes del documento con ejemplos. Para obtener ejemplos que comparan los documentos XML antes y después de una actualización, consulte el apartado “Ejemplos” en la página 210

Tabla 40. Normas de la función Update

Tipo de elemento o nodo	Ejemplo de código de documento XML	Estado después de la actualización
Declaración XML	`<?xml version='1.0' encoding='utf-8' standalone='yes' >`	Se conserva la declaración XML: • Se conserva la información de la versión. • La declaración de codificación se conserva y aparece cuando se especifica en el documento original. • La declaración autónoma se conserva y aparece cuando se especifica en el documento original. • Después de la actualización, se utilizan comillas simples para describir los valores.

Tabla 40. Normas de la función Update (continuación)

Tipo de elemento o nodo	Ejemplo de código de documento XML	Estado después de la actualización

Declaración DOCTYPE	`<!DOCTYPE books SYSTEM`
	`  "http://dtds.org/books.dtd" >`
	`<!DOCTYPE books PUBLIC`
	`  "local.books.dtd"`
	`  "http://dtds.org/books.dtd" >`
	`<!DOCTYPE books>`
	`-Any of`
	`<!DOCTYPE books`
	`  ( S ExternalID ) ?`
	`  [ internal-dtd-subset ] >`
	`-Such as`
	`<!DOCTYPE books`
	`  [ <!ENTITY mydog "Spot"> ] >?`
	`  [ internal-dtd-subset ] >`
		Se conserva la declaración del tipo de documento:
		• Se soporta el nombre de elemento raíz.
		• Se conservan los `ExternalID` públicos y del sistema y aparecen cuando se especifican en el documento original.
		• El subconjunto Internal DTD NO se conserva. Se sustituyen las entidades; se procesan los valores por omisión de los atributos y aparecen en los documentos de salida.
		• Después de la actualización, se utilizan comillas dobles para describir los valores URI públicos y del sistema.
		• El analizador XML4c actual no informa sobre una declaración XML que no contenga un `ExternalID` o un subconjunto DTD interno. Después de la actualización, faltaría en este caso la declaración DOCTYPE.

Instrucciones de proceso	`<?xml-stylesheet`
	`  title="compact"`
	`  href="datatypes1.xsl"`
	`  type="text/xsl"?>`
		Las instrucciones de proceso se conservan.

Comentarios	`<!-- comment -->`	Los comentarios se conservan cuando están dentro del elemento raíz. Se eliminan los comentarios fuera del elemento raíz.

Tabla 40. Normas de la función Update (continuación)

Tipo de elemento o nodo	Ejemplo de código de documento XML	Estado después de la actualización

Elementos	`<books>`
	content
	`</books>`
		Se conservan los elementos.

Atributos	`id='1' date='01/02/1997"`	Se conservan los atributos de los elementos. • Después de la actualización, se utilizan comillas dobles para describir los valores. • Se escapan datos dentro de los atributos. • Se sustituyen las entidades.
Nodos de texto	`This chapter is about my dog &mydoc;.` `This chapter is about my dog Spot.`	Se conservan los nodos de texto (contenido del elemento). • Se escapan datos dentro de los nodos de texto. • Se sustituyen las entidades.

Aparición múltiple

Cuando se proporciona una vía de ubicación en la UDF Update(), el contenido de cada elemento o atributo con una vía coincidente se actualiza con el valor suministrado. Esto significa que si un documento tiene vías de ubicación de aparición múltiple, la función Update sustituye los valores existentes por el valor suministrado en el parámetro *valor*.

Puede utilizar y especificar un predicado en el parámetro *vía* para ofrecer diferentes vías de ubicación con el fin de evitar actualizaciones involuntarias. Tenga presente, que la función Update de UDF da soporte a vías de ubicación que tienen predicados con atributos, pero no elementos. Consulte el apartado "Parámetros" en la página 206 para obtener más información.

Ejemplos

En los siguientes ejemplos se muestran casos de un documento XML antes y después de una actualización.

Ejemplo 1:

Antes:

```
<?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>
```

- Contiene un espacio en blanco en la declaración XML
- Especifica una instrucción de proceso
- Contiene un documento fuera del nodo raíz
- Especifica PUBLIC ExternalID
- Contiene un comentario dentro de una nota raíz

Después:

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<!DOCTYPE book PUBLIC "public.dtd" "system.dtd">
<?pitarget option1='value1' option2='value2'?><book>
  <chapter id="1" date="07/01/1997">
        <!-- first section -->
<section>This is a section in Chapter One.</section>
</chapter>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
        <footnote>A footnote in Chapter Two is here.</footnote>
</chapter>
  <price date="12/22/1998" time="11.12.13"
       timestamp="1998-12-22-11.12.13.888888">60.02</price>
</book>
```

- Se elimina el espacio en blanco dentro del markup
- Se conserva la instrucción de proceso
- No se conserva el comentario fuera del nodo raíz
- Se conserva PUBLIC ExternalID
- Se conserva el comentario dentro del nodo raíz
- El valor modificado es el valor del elemento <price>

Ejemplo 2:

Antes:

```
<?xml version='1.0'     ?>
<!DOCTYPE book>
<!-- comment -->
<book>
  ...
</book>
```

Contiene una declaración DOCTYPE sin un ExternalID o subconjunto DTD interno. No está soportado.

Después:

```
<?xml version='1.0'?>
<book>
  ...
</book>
```

El analizador XML no ha informado sobre la declaración DOCTYPE y ésta no se conserva.

Ejemplo 3:

Antes:

```
<?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>
```

- Contiene un espacio en blanco en el markup
- Especifica un subconjunto DTD interno
- Especifica la entidad en el nodo de texto

Después:

```
<?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>
```

- Se elimina el espacio en blanco del markup
- No se conserva el subonconjunto DTD interno
- Se resuelve y se sustituye la entidad en el nodo de texto

Capítulo 10. Procedimientos almacenados del XML Extender

El XML Extender proporciona procedimientos almacenados para la administración y gestión de columnas y colecciones XML. Estos procedimientos almacenados se pueden llamar desde un cliente DB2. La interfaz cliente puede estar incorporada en SQL, ODBC o JDBC. Consulte la sección sobre procedimientos almacenados del manual *DB2 UDB Administration Guide* para obtener detalles sobre cómo invocar procedimientos almacenados.

Los procedimientos almacenados utilizan el esquema db2xml, que es el nombre de esquema del XML Extender

El XML Extender proporciona tres tipos de procedimientos almacenados:

- "Procedimientos almacenados de administración" en la página 215: ayuda al usuario a realizar tareas administrativas
- "Procedimientos almacenados de composición" en la página 223: genera documentos XML utilizando datos contenidos en tablas de base de datos existentes
- "Procedimientos almacenados de descomposición" en la página 232: descompone documentos XML de entrada y almacena datos en tablas de base de datos, nuevas o existentes

Los límites de parámetro utilizados por los procedimientos almacenados de la colección XML se describen en el apartado "Apéndice D. Límites del XML Extender" en la página 295.

Especificación de archivos de inclusión

Asegúrese de que el programa utilizado para invocar procedimientos almacenados contiene los archivos de cabecera externos del XML Extender. Los archivos de cabecera están situados en el directorio `DXX_INSTALL/include`. `DXX_INSTALL` es el directorio de instalación del XML Extender. Este directorio varía según el sistema operativo. Los archivos de cabecera son:

dxx.h Los tipos definidos de constantes y de datos del XML Extender.

dxxrc.h El código de retorno del XML Extender

La sintaxis para incluir estos archivos de cabecera es:

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

Asegúrese de que la vía de acceso de los archivos de inclusión está especificada en el archivo makefile mediante la opción de compilación.

Invocación de procedimientos almacenados del XML Extender

En general, la sintaxis para invocar el XML Extender es:

```
CALL db2xml.punto_entrada_función
```

Donde:

db2xml
Especifica la biblioteca de los procedimientos almacenados del XML Extender. En los sistemas operativos UNIX, la biblioteca se almacena en el directorio `sqllib/function`. En los sistemas operativos Windows, se almacena en el directorio *DXX_INSTALL*`/bin`.

punto_entrada_función
Especifica los argumentos que se pasan al procedimiento almacenado.

En la sentencia CALL, los argumentos que se pasan al procedimiento almacenado deben ser variables de lenguaje principal, no constantes ni expresiones. Las variables de lenguaje principal pueden tener indicadores nulos. Puede ver ejemplos de invocación de procedimientos almacenados en los directorios `DXX_INSTALL/samples/c` y `DXX_INSTALL/samples/cli`, y en las secciones siguientes del presente manual: "Proceso de composición del documento XML" en la página 41 y "Capítulo 6. Gestión de datos de una colección XML" en la página 139.

El directorio `DXX_INSTALL/samples/c` contiene archivos de código SQC que invocan procedimientos almacenados para colecciones XML utilizando SQL incorporado. El directorio `DXX_INSTALL/samples/cli` contiene archivos de ejemplo que invocan procedimientos almacenados utilizando la interfaz de nivel de llamada (interfaz CLI).

Aumento del límite del CLOB

El límite por omisión del parámetro CLOB cuando se pasa a un procedimiento almacenado es de 1 MB. Puede aumentarlo realizando los pasos siguientes:

1. Elimine cada procedimiento almacenado. Por ejemplo:
   ```
   db2   "drop procedure db2xml.dxxShredXML"
   ```
2. Cree un nuevo procedimiento con el límite de CLOB aumentado. Por ejemplo:

```
db2   "create procedure db2xml.dxxShredXML(in     dadBuf      clob(100K),
                                           in     XMLObj      clob(10M),
                                           out    returnCode  integer,
                                           out    returnMsg   varchar(1024)
                                           )
       external name 'db2xml.dxxShredXML'
       language C
       parameter style DB2DARI
       not deterministic
       fenced
       null call;
```

Antes de comenzar

Enlace su base de datos con el procedimiento almacenado del XML Extender y con los archivos de enlace para CLI de DB2. Puede utilizar un archivo de mandatos de ejemplo, `getstart_prep.cmd`, para enlazar los archivos. Este archivo de mandatos está en el directorio `DXX_INSTALL/samples/cmd`.

1. Conecte con la base de datos. Por ejemplo:

   ```
   db2 "connect to SALES_DB"
   ```

2. Cambie al directorio `DXX_INSTALL/bnd` y enlace el XML Extender con la base de datos.

   ```
   db2 "bind @dxxbind.lst"
   ```

3. Cambie al directorio `sqllib/bnd` y enlace la interfaz CLI con la base de datos.

   ```
   db2 "bind @db2cli.lst"
   ```

4. Finalice la conexión.

   ```
   db2 "terminate"
   ```

Procedimientos almacenados de administración

Estos procedimientos almacenados se utilizan para tareas administrativas, tales como habilitar o inhabilitar una columna o colección XML. Se invocan mediante el asistente de administración del XML Extender y el mandato de administración **dxxadm**.

dxxEnableDB()

Propósito

Habilita la base de datos. Cuando se habilita la base de datos, el XML Extender crea los objetos siguientes:

- Los tipos definidos por el usuario (los UDT) del XML Extender.
- Las funciones definidas por el usuario (las UDF) del XML Extender
- La tabla de referencia DTD del XML Extender, DTD_REF, donde se almacenan las DTD e información sobre cada DTD. Para obtener una descripción completa de la tabla DTD_REF, vea "Tabla de referencia DTD" en la página 237.
- La tabla de utilización del XML Extender, XML_USAGE, que almacena información común para cada columna habilitada para XML y para cada colección. Para obtener una descripción completa de la tabla XML_USAGE, vea "Tabla de utilización de XML" en la página 238.

```
dxxEnableDB(char(nombreBd) nombreBd,         /* entrada */
            long           códigoRetorno,    /* salida */
            varchar(1024)  mensRetorno)      /* salida */
```

Parámetros

Tabla 41. Parámetros de dxxEnableDB()

Parámetro	Descripción	Parámetro de E/S
nombreBd	Es el nombre de la base de datos.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

dxxDisableDB()

Propósito

Inhabilita la base de datos. Cuando el XML Extender inhabilita la base de datos, elimina los objetos siguientes:

- Los tipos definidos por el usuario (los UDT) del XML Extender.
- Las funciones definidas por el usuario (las UDF) del XML Extender
- La tabla de referencia DTD del XML Extender, DTD_REF, donde se almacenan las DTD e información sobre cada DTD. Para obtener una descripción completa de la tabla DTD_REF, vea "Tabla de referencia DTD" en la página 237.
- La tabla de utilización del XML Extender, XML_USAGE, que almacena información común para cada columna habilitada para XML y para cada colección. Para obtener una descripción completa de la tabla XML_USAGE, vea "Tabla de utilización de XML" en la página 238.

Importante: Debe inhabilitar todas las columnas XML antes de intentar inhabilitar una base de datos. El XML Extender no puede inhabilitar una base de datos que contenga columnas o colecciones habilitadas para XML.

```
dxxDisableDB(char(nombreBd)     nombreBd,     /* entrada */
             long           códigoRetorno,    /* salida */
             varchar(1024)  mensRetorno)      /* salida */
```

Parámetros

Tabla 42. Parámetros de dxxDisableDB()

Parámetro	Descripción	Parámetro de E/S
nombreBd	Es el nombre de la base de datos.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

dxxEnableColumn()

Propósito

Habilita una columna XML. Cuando habilita una columna, el XML Extender realiza estas tareas:

- Determina si la tabla XML tiene una clave primaria; si no la tiene, el XML Extender modifica la tabla XML y añade una columna llamada DXXROOT_ID.
- Crea tablas auxiliares, que se especifican en el archivo DAD, con una columna que contiene un identificador exclusivo para cada fila de la tabla XML. Esta columna es el *id_raíz* que el usuario especifica o el valor DXXROOT_ID proporcionado por el XML Extender.
- Crea una vista predefinida para la tabla XML y sus tablas auxiliares, utilizando opcionalmente un nombre especificado por el usuario.

```
dxxEnableColumn(char(nombreBd) nombreBd,       /* entrada */
                char(nombreTb) nombreTb,       /* entrada */
                char(nombreCol) nombreCol,     /* entrada */
                CLOB(100K)     DAD,            /* entrada */
                char(espaciotablas) espaciotablas,  /* entrada */
                char(vistaPredefinida) vistaPredefinida, /* entrada */
                char(idRaíz) idRaíz,           /* entrada */
                long           códigoRetorno,  /* salida */
                varchar(1024)  mensRetorno)    /* salida */
```

Parámetros

Tabla 43. Parámetros de dxxEnableColumn()

Parámetro	Descripción	Parámetro de E/S
nombreBd	Es el nombre de la base de datos.	ENTRADA
nombreTb	Es el nombre de la tabla donde reside la columna XML.	ENTRADA
nombreCol	Es el nombre de la columna XML.	ENTRADA
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
espaciotablas	Es el espacio de tablas, distinto del espacio de tablas predefinido, que contiene las tablas auxiliares. Si no se especifica este parámetro, se utiliza el espacio de tablas predefinido.	ENTRADA
vistaPredefinida	Es el nombre de la vista predefinida que asocia la tabla de aplicación y las tablas auxiliares.	ENTRADA

Tabla 43. Parámetros de dxxEnableColumn() (continuación)

Parámetro	Descripción	Parámetro de E/S
ID_raíz	Es el nombre de la clave primaria, contenida en la tabla de aplicación, que se utilizará como *id_raíz* para la tabla auxiliar.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

dxxDisableColumn()

Propósito

Inhabilita la columna habilitada para XML. Cuando la columna XML está inhabilitada, ya no puede contener tipos de datos XML.

```
dxxDisableColumn(char(nombreBd) nombreBd,      /* entrada */
                 char(nombreTb) nombreTb,      /* entrada */
                 char(nombreCol) nombreCol,    /* entrada */
                 long            códigoRetorno,        /* salida */
                 varchar(1024)   mensRetorno)          /* salida */
```

Parámetros

Tabla 44. Parámetros de dxxDisableColumn()

Parámetro	Descripción	Parámetro de E/S
nombreBd	Es el nombre de la base de datos.	ENTRADA
nombreTb	Es el nombre de la tabla donde reside la columna XML.	ENTRADA
nombreCol	Es el nombre de la columna XML.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

dxxEnableCollection()

Propósito

Habilita una colección XML que está asociada a una tabla de aplicación.

```
dxxEnableCollection(char(nombreBd) nombreBd,       /* entrada */
                    char(nombreCol) nombreCol,     /* entrada */
                    CLOB(100K)     DAD,           /* entrada */
                    char(espaciotablas) espaciotablas,  /* entrada */
                    long           códigoRetorno, /* salida */
                    varchar(1024)  mensRetorno)   /* salida */
```

Parámetros

Tabla 45. Parámetros de dxxEnableCollection()

Parámetro	Descripción	Parámetro de E/S
nombreBd	Es el nombre de la base de datos.	ENTRADA
nombreCol	Es el nombre de la colección XML.	ENTRADA
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
espaciotablas	Es el espacio de tablas, distinto del espacio de tablas predefinido, que contiene las tablas auxiliares. Si no se especifica este parámetro, se utiliza el espacio de tablas predefinido.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

dxxDisableCollection()

Propósito

Inhabilita una colección habilitada para XML, eliminando los marcadores que identifican tablas y columnas como integrantes de una colección.

```
dxxDisableCollection(char(nombreBd) nombreBd,      /* entrada */
                     char(nombreCol) nombreCol,    /* entrada */
                     long           códigoRetorno,   /* salida */
                     varchar(1024)  mensRetorno)     /* salida */
```

Parámetros

Tabla 46. Parámetros de dxxDisableCollection()

Parámetro	Descripción	Parámetro de E/S
nombreBd	Es el nombre de la base de datos.	ENTRADA
nombreCol	Es el nombre de la colección XML.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Procedimientos almacenados de composición

Los procedimientos almacenados de composición, dxxGenXML() y dxxRetrieveXML(), se utilizan para generar documentos XML a partir de datos XML existentes en tablas de base de datos. El procedimiento almacenado dxxGenXML() utiliza como entrada un archivo DAD; no necesita una colección XML habilitada. El procedimiento dxxRetrieveXML() utiliza como entrada un nombre de colección XML habilitada.

dxxGenXML()

Propósito

Crea documentos XML utilizando datos almacenados en tablas de colección XML, que están especificadas por el elemento <Xcollection> en el archivo DAD, e inserta cada documento XML en la tabla resultante en calidad de fila. El usuario también puede abrir un cursor en la tabla resultante y obtener el conjunto resultante.

Para proporcionar flexibilidad, dxxGenXML() también permite al usuario especificar el número máximo de filas que se han de generar en la tabla resultante. Esto disminuye la cantidad de tiempo que la aplicación debe esperar los resultados durante un proceso de prueba. El procedimiento almacenado devuelve el número de filas de la tabla y la información de error producida, que incluye los códigos de error y los mensajes de error.

Para permitir realizar consultas dinámicas, dxxGenXML() admite el parámetro de entrada *alter_temp* (alteración temporal). Basándose en el valor de *tipo_alter_temp*, la aplicación puede anular la sentencia SQL_stmt, para la correlación SQL, o las condiciones del nodo_RDB, para la correlación de nodo_RDB, en el archivo DAD. El parámetro de entrada *tipo_alter_temp* se utiliza para especificar el tipo de *alteración temporal*. Para obtener detalles sobre el parámetro *alter_temp*, vea "Anulación dinámica de valores en el archivo DAD" en la página 144.

```
dxxGenXML(CLOB(100K)     DAD,             /* entrada */
          char(nombreTabResul) nombreTabResul, /* entrada */
          integer        tipo_alter_temp  /* entrada */
          varchar(1024)  alter_temp,      /* entrada */
          integer        máxFilas,        /* entrada */
          integer        númFilas,        /* salida */
          long           códigoRetorno,   /* salida */
          varchar(1024)  mensRetorno)     /* salida */
```

Parámetros

Tabla 47. Parámetros de dxxGenXML()

Parámetro	Descripción	Parámetro de E/S
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
nombreTabResul	Es el nombre de la tabla resultante, la cual debe existir antes de ejecutar la llamada. La tabla contiene una sola columna, de tipo XMLVARCHAR o XMLCLOB.	ENTRADA

Tabla 47. Parámetros de dxxGenXML() (continuación)

Parámetro	Descripción	Parámetro de E/S
tipo_alter_temp	Es un distintivo que indica el tipo del parámetro *alter_temp* que sigue a continuación: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal por una sentencia SQL_stmt. • **XML_OVERRIDE**: Alteración temporal por una condición basada en XPath.	ENTRADA
alter_temp	Invalida la condición especificada en el archivo DAD. El valor de entrada está basado en *tipo_alter_temp*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia SQL válida. Si se utiliza este valor para *tipo_alter_temp*, se debe utilizar la correlación SQL en el archivo DAD. La sentencia SQL de entrada invalida la sentencia SQL_stmt especificada en el archivo DAD. • **XML_OVERRIDE**: Serie de caracteres que contiene una o más expresiones entre comillas dobles, separadas por "AND". Si se utiliza este valor para *tipo_alter_temp*, se debe utilizar la correlación de nodo_RDB en el archivo DAD.	ENTRADA
máxFilas	Es el número máximo de filas de la tabla resultante.	ENTRADA

Tabla 47. Parámetros de dxxGenXML() (continuación)

Parámetro	Descripción	Parámetro de E/S
númFilas	Es el número real de filas creadas de la tabla resultante.	SALIDA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos

En el ejemplo, se crea la tabla resultante XML_ORDER_TAB, la cual tiene una sola columna de tipo XMLVARCHAR.

```
#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;    /* archivo DAD */
            char    result_tab[32];   /* nombre de la tabla resultante */
            char    override[2];      /* alteración temporal, para nulo */
            short   overrideType;     /* definido en dxx.h */
            short   max_row;          /* número máximo de filas */
            short   num_row;          /* número real de filas */
            long    returnCode;       /* código de error de retorno */
            char    returnMsg[1024];  /* texto del mensaje de error */
            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;

            /* crear tabla */
            EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

           /* leer datos de un archivo y ponerlos en un CLOB */
            strcpy(dadfile.name,"e:\dxx\dad\litem3.dad");
            dadfile.name_length = strlen("e:\dxx\dad\litem3.dad");
            dadfile.file_options = SQL_FILE_READ;
            EXEC SQL VALUES (:dadfile) INTO :dad;
            strcpy(result_tab,"xml_order_tab");
            override[0] = '\0';
```

```
overrideType = NO_OVERRIDE;
max_row = 500;
num_row = 0;
returnCode = 0;
msg_txt[0] = '\0';
collection_ind = 0;
dad_ind = 0;
rtab_ind = 0;
ov_ind = -1;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Llamar al procedimiento almacenado */
EXEC SQL CALL dxxGenXML(:dad:dad_ind;
       :result_tab:rtab_ind,
       :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
       :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxRetrieveXML()

Propósito

Permite que un mismo archivo DAD sea utilizado tanto para la composición como para la descomposición. El procedimiento almacenado dxxRetrieveXML() también sirve para recuperar documentos XML descompuestos. dxxRetrieveXML() utiliza como entrada un almacenamiento intermedio donde está el archivo DAD, el nombre de la tabla resultante creada y el número máximo de filas a devolver. Devuelve un conjunto resultante formado por la tabla resultante, el número real de filas del conjunto resultante, un código de error y un texto de mensaje.

Para permitir realizar consultas dinámicas, dxxRetrieveXML() admite el parámetro de entrada *alter_temp* (alteración temporal). Basándose en el valor de *tipo_alter_temp*, la aplicación puede anular la sentencia SQL_stmt, para la correlación SQL, o las condiciones del nodo_RDB, para la correlación de nodo_RDB, en el archivo DAD. El parámetro de entrada *tipo_alter_temp* se utiliza para especificar el tipo de *alteración temporal*. Para obtener detalles sobre el parámetro *alter_temp*, vea "Anulación dinámica de valores en el archivo DAD" en la página 144.

Los requisitos del archivo DAD para dxxRetrieveXML() son los mismos que para dxxGenXML(). La única diferencia es que el archivo DAD no es un parámetro de entrada de dxxRetrieveXML(), sino que se utiliza el nombre de una colección XML habilitada.

```
dxxRetrieveXML(char(nombreColección) nombreColección,   /* entrada */
         char(nombreTabResul) nombreTabResul,          /* entrada */
         integer        tipo_alter_temp,    /* entrada */
         varchar(1024)  alter_temp,         /* entrada */
         integer        máxFilas,           /* entrada */
         integer        númFilas,           /* salida */
         long           códigoRetorno,      /* salida */
         varchar(1024)  mensRetorno)        /* salida */
```

Parámetros

Tabla 48. Parámetros de dxxRetrieveXML()

Parámetro	Descripción	Parámetro de E/S
nombreColección	Es el nombre de una colección XML habilitada.	ENTRADA
nombreTabResul	Es el nombre de la tabla resultante, la cual debe existir antes de ejecutar la llamada. La tabla contiene una sola columna, de tipo XMLVARCHAR o XMLCLOB.	ENTRADA

Tabla 48. Parámetros de dxxRetrieveXML() (continuación)

Parámetro	Descripción	Parámetro de E/S
tipo_alter_temp	Es un distintivo que indica el tipo del parámetro *alter_temp* que sigue a continuación: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal por una sentencia SQL_stmt. • **XML_OVERRIDE**: Alteración temporal por una condición basada en XPath.	ENTRADA
alter_temp	Invalida la condición especificada en el archivo DAD. El valor de entrada está basado en *tipo_alter_temp*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia SQL válida. Si se utiliza este valor para *tipo_alter_temp*, se debe utilizar la correlación SQL en el archivo DAD. La sentencia SQL de entrada invalida la sentencia SQL_stmt especificada en el archivo DAD. • **XML_OVERRIDE**: Serie de caracteres que contiene una o más expresiones entre comillas dobles, separadas por "AND". Si se utiliza este valor para *tipo_alter_temp*, se debe utilizar la correlación de nodo_RDB en el archivo DAD.	ENTRADA
máxFilas	Es el número máximo de filas de la tabla resultante.	ENTRADA

Tabla 48. Parámetros de dxxRetrieveXML() (continuación)

Parámetro	Descripción	Parámetro de E/S
númFilas	Es el número real de filas creadas de la tabla resultante.	SALIDA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos

Lo siguiente es un ejemplo de una llamada a dxxRetrieveXML(). En este ejemplo, se crea la tabla resultante XML_ORDER_TAB, la cual tiene una sola columna de tipo XMLVARCHAR.

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

            EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char    collection[32];  /* almacenamiento intermedio con DAD */
              char    result_tab[32];  /* nombre de la tabla resultante */
              char    override[2];     /* alteración temporal, para nulo */
              short   overrideType;    /* definido en dxx.h */
              short   max_row;         /* número máximo de filas */
              short   num_row;         /* número real de filas */
              long    returnCode;      /* código de error de retorno */
              char    returnMsg[1024]; /* texto del mensaje de error */
              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;

              /* crear tabla */
              EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

              /* inicializar variable de lenguaje principal e indicadores */
              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;

/* Llamar al procedimiento almacenado */
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);
```

Procedimientos almacenados de descomposición

Los procedimientos almacenados de descomposición, dxxInsertXML() y dxxShredXML(), se utilizan para desglosar documentos XML de entrada y almacenar datos en tablas de base de datos, nuevas o existentes. El procedimiento dxxInsertXML() utiliza como entrada un nombre de colección XML habilitada. El procedimiento almacenado dxxShredXML() utiliza como entrada un archivo DAD; no necesita una colección XML habilitada.

dxxShredXML()

Propósito

El procedimiento almacenado dxxShredXML() forma pareja con el procedimiento almacenado dxxGenXML(). Para que dxxShredXML() sea efectivo, todas las tablas especificadas en el archivo DAD deben existir, y todas las columnas y sus tipos de datos que están especificados en ese archivo DAD deben ser coherentes con las tablas existentes. El procedimiento almacenado dxxShredXML() no necesita que exista la relación clave primaria-clave foránea entre las tablas que se unen; esta relación es creada por el XML Extender durante el proceso de habilitación de la colección. Sin embargo, las columnas de la condición de unión que están especificadas en el nodo_RDB del nodo_de_elemento raíz deben existir en las tablas.

```
dxxShredXML(CLOB(100K)    DAD,             /* entrada */
            CLOB(1M)       objxml,         /* entrada */
            long           códigoRetorno,  /* salida */
            varchar(1024)  mensRetorno)    /* salida */
```

Parámetros

Tabla 49. Parámetros de dxxShredXML()

Parámetro	Descripción	Parámetro de E/S
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
objxml	Es un objeto de documento XML cuyo tipo es XMLCLOB.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos

Lo siguiente es un ejemplo de una llamada a 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; /* archivo DAD */
              SQL TYPE is CLOB xmlDoc;        /* documento XML de entrada */
              SQL TYPE is CLOB_FILE xmlFile;  /* archivo XML de entrada */
              long     returnCode;            /* código de error */
              char     returnMsg[1024];    /* texto del mensaje de error */
```

```
short    dad_ind;
short    xmlDoc_ind;
short    returnCode_ind;
short    returnMsg_ind;
EXEC SQL END DECLARE SECTION;

/* inicializar variable de lenguaje principal e indicadores */
strcpy(dadFile.name,"c:\dxx\samples\dad\
       getstart_xcollection.dad");
dadFile.name_length=strlen("c:\dxx\samples\dad\
       getstart_xcollection.dad");
dadFile.file_option=SQL_FILE_READ;
strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart.xml");
xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart.xml");
xmlFile.file_option=SQL_FILE_READ;
SQL EXEC VALUES (:dadFile) INTO :dad;
SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
returnCode = 0;
returnMsg[0] = '\0';
dad_ind = 0;
xmlDoc_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Llamar al procedimiento almacenado */
EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
         :xmlDoc:xmlDoc_ind,
         :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

Propósito

Utiliza dos parámetros de entrada: el nombre de una colección XML habilitada y el documento que se debe descomponer; devuelve dos parámetros de salida: un código de retorno y un mensaje de retorno.

```
dxxInsertXML(char(nombreColección) nombreColección,   /* entrada */
             CLOB(1M)       objxml,          /* entrada */
             long           códigoRetorno,   /* salida */
             varchar(1024)  mensRetorno)     /* salida */
```

Parámetros

Tabla 50. Parámetros de dxxInsertXML()

Parámetro	Descripción	Parámetro de E/S
nombreColección	Es el nombre de una colección XML habilitada.	ENTRADA
objxml	Es un objeto de documento XML cuyo tipo es CLOB.	ENTRADA
códigoRetorno	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
mensRetorno	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos

En el ejemplo siguiente, la llamada a dxxInsertXML() descompone el documento XML de entrada, `e:\xml\order1.xml`, e inserta datos en las tablas de colección SALES_ORDER de acuerdo con la correlación definida en el archivo DAD mediante el cual se habilitó el documento.

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

EXEC SQL INCLUDE SQLCA;
            EXEC SQL BEGIN DECLARE SECTION;
              char           collection[64];  /* nombre de una colección XML */
              SQL TYPE is CLOB_FILE xmlDoc;   /* documento XML de entrada */
              long           returnCode;      /* código de error */
              char           returnMsg[1024]; /* texto del mensaje de error */
              short          collection_ind;
              short          xmlDoc_ind;
              short          returnCode_ind;
              short          returnMsg_ind;
              EXEC SQL END DECLARE SECTION;

              /* inicializar variable de lenguaje principal e indicadores */
              strcpy(collection,"sales_ord")
```

```
strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart.xml");
xmlobj.name_length=strlen
       ("c:\dxx\samples\cmd\getstart.xml");
xmlobj.file_option=SQL_FILE_READ;
returnCode = 0;
returnMsg[0] = '\0';
collection_ind = 0;
xmlobj_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Llamar al procedimiento almacenado */
EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
     :xmlobj:xmlobj_ind,
     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Capítulo 11. Tablas de soporte de administración

Cuando se habilita una base de datos, se crea una tabla de referencia DTD, DTD_REF, y una tabla XML_USAGE. La tabla DTD_REF contiene información sobre todas las DTD. La tabla XML_USAGE contiene información común sobre cada columna habilitada para XML.

Los límites de parámetro listados en las tablas de soporte se describen en el "Apéndice D. Límites del XML Extender" en la página 295.

Tabla de referencia DTD

El XML Extender también actúa como depósito las DTD de XML. Cuando se habilita una base de datos para XML, se crea una tabla de referencia DTD, DTD_REF. Cada fila de esta tabla representa una DTD, junto con información adicional de metadatos. Los usuarios pueden acceder a esta tabla e insertar sus propias DTD. Las DTD de la tabla DTD_REF se utilizan para validar documentos XML y para ayudar a las aplicaciones a definir un archivo DAD. Su nombre de esquema es db2xml. Una tabla DTD_REF puede tener las columnas que se muestran en la Tabla 51.

Tabla 51. Tabla DTD_REF

Nombre de columna	Tipo de datos	Descripción
DTDID	VARCHAR(128)	Clave primaria (exclusiva y no nula). Se utiliza para identificar la DTD. Cuando se especifica la clave primaria en el archivo DAD, este archivo debe seguir el esquema definido por la DTD.
CONTENT	XMLCLOB	Contenido de la DTD.
USAGE_COUNT	INTEGER	Número de columnas XML y de colecciones XML de la base de datos que utilizan esta DTD para definir sus archivos DAD.
AUTHOR	VARCHAR(128)	Autor de la DTD; información de entrada opcional para el usuario.
CREATOR	VARCHAR(128)	ID de usuario que realiza la primera inserción. La columna CREATOR es opcional.
UPDATOR	VARCHAR(128)	ID de usuario que realiza la última actualización. La columna UPDATOR es opcional.

Restricción: La aplicación sólo puede modificar la DTD cuando USAGE_COUNT es cero.

Tabla de utilización de XML

Almacena información común sobre cada columna habilitada para XML. El nombre de esquema de la tabla XML_USAGE es db2xml, y su clave primaria es (*nombre_tabla*, *nombre_col*). Cuando se habilita la base de datos se crea una tabla XML_USAGE que tiene las columnas mostradas en la Tabla 52.

Tabla 52. Tabla XML_USAGE

Nombre de columna	Descripción
esquema_tabla	En el caso de una columna XML, es el nombre de esquema de la tabla de usuario que contiene una columna XML. En el caso de una colección XML, es el valor "DXX_COLL" como nombre de esquema por omisión.
nombre_tabla	En el caso de una columna XML, es el nombre de la tabla de usuario que contiene una columna XML. En el caso de una colección XML, es un valor "DXX_COLLECTION," el cual identifica la entidad como colección.
nombre_col	Es el nombre de la columna XML o colección XML. Forma parte de la clave compuesta junto con el nombre de tabla.
DTDID	Es el ID de la DTD almacenada en la tabla DTD_REF que sirve para definir el archivo DAD. Es la clave foránea.
DAD	Es el contenido del archivo DAD que está asociado a la columna.
vista_predefinida	Almacena el nombre de la vista predefinida, si existe una.
sufijo_desencadenante	Valor no nulo. Se utiliza para nombres de desencadenantes exclusivos.
Validación	1 para Sí, 0 para No.
modalidad_acceso	1 para la colección XML, 0 para la columna XML

Restricción: La aplicación sólo puede modificar la DTD cuando USAGE_COUNT es cero.

Capítulo 12. Información de diagnóstico

Todas sentencias de SQL incorporado y las llamadas de interfaz de línea de mandatos de DB2 de su programa, incluidas las que invocan las funciones DB2 definidas por el usuario del XML Extender, generan códigos que indican si la sentencia o llamada se ha ejecutado satisfactoriamente.

El programa del usuario puede obtener información que sirve de complemento a estos códigos. Esta información complementaria puede ser información de estado de SQL y mensajes de error. El usuario puede utilizar esta información de diagnóstico para identificar y corregir problemas en un programa.

A veces, la fuente de un problema no se puede diagnosticar con facilidad. En estos casos, puede ser necesario que proporcione información al suministrador de Soporte de software para identificar y corregir el problema. El XML Extender incluye un recurso de rastreo que registra la actividad del XML Extender. La información proporcionada por el rastreo puede ser una información valiosa para el Centro de Soporte de Software de IBM. Debe utilizar el recurso de rastreo sólo de acuerdo con las instrucciones del Centro de Soporte de Software de IBM.

Este capítulo describe cómo acceder a esta información de diagnóstico. Incluye los temas siguientes:

- ¿Cómo utilizar los códigos de retorno de las UDF del XML Extender?
- ¿Cómo controlar la función de rastreo?

Este capítulo también describe los códigos de estado de SQL (SQLSTATE) y los mensajes de error que el XML Extender puede devolver.

Manejo de los códigos de retorno de las UDF

Las sentencias de SQL incorporado devuelven códigos en los campos SQLCODE, SQLWARN y SQLSTATE de la estructura SQLCA. Esta estructura está definida en un archivo INCLUDE de SQLCA. (Para obtener más información sobre la estructura SQLCA y el archivo INCLUDE de SQLCA, vea el manual *DB2 Application Development Guide*.)

Las llamadas CLI de DB2 devuelven valores para SQLCODE y SQLSTATE que el usuario puede recuperar utilizando la función SQLError. (Para obtener más información sobre la recuperación de información de errores mediante la función SQLError, vea el manual *CLI Guide and Reference*.)

Si el valor de SQLCODE es igual a 0, significa que la sentencia se ejecutó satisfactoriamente (con posibles condiciones de aviso). Si el valor de SQLCODE es positivo, significa que la sentencia se ejecutó satisfactoriamente, pero se emitió un aviso. (Cuando el valor de SQLCODE es cero o positivo, las sentencias de SQL incorporado devuelven información sobre avisos en el campo SQLWARN.) Si el valor de SQLCODE es negativo, significa que se ha producido un error.

DB2 asocia un mensaje a cada valor de SQLCODE. Si una UDF del XML Extender detecta una condición de aviso o error, transfiere a DB2 la información asociada a la condición para que se incluya en el mensaje del SQLCODE.

Los valores de SQLSTATE contiene códigos que sirven de complemento a los mensajes del SQLCODE. Vea “Códigos de SQLSTATE” en la página 241 para obtener una descripción de los códigos de SQLSTATE que devuelve el XML Extender.

Las sentencias de SQL incorporado y llamadas CLI de DB2 que arrancan funciones UDF de DB2 del XML Extender pueden devolver mensajes de SQLCODE y valores de SQLSTATE que son exclusivos de estas UDF, pero DB2 devuelve estos valores de la misma manera que lo hace para otras sentencias de SQL incorporado u otras llamadas CLI de DB2. Por tanto, el usuario accede a estos valores de la misma manera que para las sentencias de SQL incorporado o llamadas CLI de DB2 que no arrancan funciones UDF de DB2 del XML Extender.

Vea “Códigos de SQLSTATE” en la página 241 para conocer los valores de SQLSTATE y el número de mensaje asociado a los mensajes que puede devolver el XML Extender. Vea “Mensajes” en la página 246 para obtener información sobre cada mensaje.

Manejo de los códigos de retorno de procedimientos almacenados

El XML Extender proporciona códigos de retorno para ayudar a resolver problemas que se producen en los procedimientos almacenados. Cuando reciba un código de retorno procedente de un procedimiento almacenado, examine el archivo siguiente, el cual asocia el código de retorno con un número de mensaje de error del XML Extender y con una constante simbólica.

```
DXX_INSTALL/include/dxxrc.h
```

Puede buscar el número de mensaje de error en “Mensajes” en la página 246 y utilizar la información de diagnóstico proporcionada en la explicación de los mensajes.

Códigos de SQLSTATE

La Tabla 53 lista y describe los valores de SQLSTATE que puede devolver el XML Extender. La descripción de cada valor de SQLSTATE incluye la representación simbólica del valor. La tabla también lista el número de mensaje que está asociado a cada valor de SQLSTATE. Vea "Mensajes" en la página 246 para obtener información sobre cada mensaje.

Tabla 53. Códigos SQLSTATE y números de mensaje asociados

SQLSTATE	Núm. mensaje	Descripción
00000	DXX*nnnn*I	No se ha producido ningún error.
01HX0	DXXD003W	El elemento o atributo especificado en la expresión de vía de acceso no está en el documento XML.
38X00	DXXC000E	El XML Extender no puede abrir el archivo especificado.
38X01	DXXA072E	El XML Extender ha intentado enlazar automáticamente la base de datos antes de habilitarla, pero no ha encontrado los archivos de enlace.
	DXXC001E	El XML Extender no pudo encontrar el archivo especificado.
38X02	DXXC002E	El XML Extender no puede leer datos del archivo especificado.
38X03	DXXC003E	El XML Extender no puede escribir datos en el archivo.
	DXXC011E	El XML Extender no puede escribir datos en el archivo de control de rastreo.
38X04	DXXC004E	El XML Extender no pudo utilizar el localizador especificado.
38X05	DXXC005E	El tamaño del archivo es mayor que el tamaño de XMLVarchar y el XML Extender no puede importar todos los datos del archivo.
38X06	DXXC006E	El tamaño del archivo es mayor que el tamaño de XMLCLOB y el XML Extender no puede importar todos los datos del archivo.

Tabla 53. Códigos SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Núm. mensaje	Descripción
38X07	DXXC007E	El número de bytes del Localizador de LOB no es igual al tamaño del archivo.
38X08	DXXD001E	Una función de extracción escalar ha utilizado una vía de ubicación que aparece varia veces. Una función escalar sólo puede utilizar una vía de ubicación que no tiene aparición múltiple.
38X09	DXXD002E	La expresión de la vía de búsqueda tiene una sintaxis incorrecta.
38X10	DXXG002E	El XML Extender no pudo asignar memoria del sistema operativo.
38X11	DXXA009E	Este procedimiento almacenado sólo es aplicable a Xcolumn.
38X12	DXXA010E	Mientras intentaba habilitar una columna, el XML Extender no pudo encontrar el DTDID, que es el identificador especificado para la DTD en el archivo de definición de acceso a documento (archivo DAD).
38X14	DXXD000E	Se ha intentado almacenar un documento no válido en una tabla. La validación ha fallado.
38X15	DXXA056E	El elemento de validación contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.
	DXXA057E	El atributo de nombre de una tabla auxiliar contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.
	DXXA058E	El atributo de nombre de una columna contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.

Tabla 53. Códigos SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Núm. mensaje	Descripción
	DXXA059E	El atributo de tipo de una columna contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.
	DXXA060E	El atributo de vía de acceso de una columna del archivo de definición de acceso a documento (DAD) es erróneo o no está presente.
	DXXA061E	El atributo multi_occurrence de una columna contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.
	DXXQ000E	Falta un elemento obligatorio en el archivo de definición de acceso a documento (archivo DAD).
38X16	DXXG004E	Se ha pasado un valor nulo a un procedimiento almacenado de XML para un parámetro obligatorio.
38X17	DXXQ001E	La sentencia de SQL contenida en la definición de acceso a documento (DAD) o la sentencia que la sustituye no es válida. Es necesaria una sentencia SELECT para generar documentos XML.
38X18	DXXG001E	El XML Extender ha encontrado un error interno.
	DXXG006E	El XML Extender ha encontrado un error interno mientras utilizaba CLI.
38X19	DXXQ002E	El sistema se ha quedado sin espacio de memoria o de disco. No hay espacio para contener los documentos XML resultantes.
38X20	DXXQ003W	La consulta SQL definida por el usuario genera más documentos XML que el máximo especificado. Sólo se devuelven el número de documentos especificado.

Tabla 53. Códigos SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Núm. mensaje	Descripción
38X21	DXXQ004E	La columna especificada no es una de las columnas incluidas en el resultado de la consulta SQL.
38X22	DXXQ005E	La correlación de la consulta SQL con XML es incorrecta.
38X23	DXXQ006E	Un elemento nodo_atributo contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un atributo de nombre.
38X24	DXXQ007E	El elemento nodo_atributo contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un elemento de columna ni un nodo_RDB.
38X25	DXXQ008E	Un elemento nodo_texto contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un elemento de columna.
38X26	DXXQ009E	La tabla de resultados especificada no se ha podido encontrar en el catálogo del sistema.
38X27	DXXQ010E	El nodo_RDB del nodo_atributo o del nodo_texto debe tener una tabla.
	DXXQ011E	El nodo_RDB del nodo_atributo o del nodo_texto debe tener una columna.
	DXXQ017E	El XML Extender ha generado un documento XML que es demasiado grande para caber en la columna de la tabla de resultados.
38X28	DXXQ012E	El XML Extender no pudo encontrar el elemento esperado mientras procesaba la DAD.

Tabla 53. Códigos SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Núm. mensaje	Descripción
	DXXQ016E	Todas las tablas deben estar definidas en el nodo_RDB del elemento superior, en el archivo de definición de acceso a documento (archivo DAD). Las tablas de sub-elemento deben coincidir con las tablas definidas en el elemento superior. El nombre de tabla de este nodo_RDB no está en el elemento superior.
38X29	DXXQ013E	El elemento de tabla o columna debe tener un nombre en el archivo de definición de acceso a documento (archivo DAD).
	DXXQ015E	La condición expresada en el elemento de condición, contenido en la definición de acceso a documento (DAD), tiene un formato no válido.
38X30	DXXQ014E	Un elemento nodo_elemento contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un atributo de nombre.
	DXXQ018E	Falta la cláusula ORDER BY en la sentencia de SQL que correlaciona SQL con XML, y que está contenida en un archivo de definición de acceso a documento (archivo DAD).
38X31	DXXQ019E	El elemento objids no tiene un elemento de columna en el archivo DAD (archivo de definición de acceso a documento) que correlaciona SQL con XML.
38X36	DXXA073E	La base de datos no estaba enlazada cuando el usuario ha intentado habilitarla.

Tabla 53. Códigos SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Núm. mensaje	Descripción
38X37	DXXG007E	El entorno local del sistema operativo del servidor no es coherente con la página de códigos de DB2.
38X38	DXXG008E	El entorno local del sistema operativo del servidor no se puede encontrar en la tabla de páginas de códigos.
38x33	DXXG005E	Este parámetro no está soportado en este release; estará soportado en el release futuro.
38x34	DXXG000E	Se ha especificado un nombre de archivo no válido.

Mensajes

El XML Extender proporciona mensajes de error para ayudar en la determinación de problemas.

Mensajes de error

El XML Extender genera los mensajes siguientes cuando finaliza una operación o detecta un error.

DXXA000I **Se está habilitando la columna** <*nombre_columna*>**. Por favor, espere.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA001S **Se ha producido un error inesperado en creación** <*ID_creación*>**, archivo** <*nombre_archivo*> **y línea** <*número_línea*>**.**

Explicación: Se ha producido un error inesperado.

Respuesta del Usuario: Si este error persiste, póngase en contacto con el Suministrador de Servicio de Software. Cuando informe del error, asegúrese de incluir todo el texto del mensaje, el archivo de rastreo y la explicación que permita reproducir el problema.

DXXA002I **Se está conectando con la base de datos** <*base_datos*>**.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA003E **No se puede conectar con la base de datos** <*base_datos*>**.**

Explicación: Es posible que la base de datos especificada no exista o que esté dañada.

Respuesta del Usuario:

1. Compruebe si la base de datos está especificada correctamente.

2. Compruebe si la base de datos existe y es accesible.
3. Determine si la base de datos está dañada. Si está dañada, solicite al administrador de la base de datos su recuperación a partir de una copia de seguridad.

DXXA004E No se puede habilitar la base de datos <*base_datos*>.

Explicación: Puede que la base de datos ya esté habilitada o que esté dañada.

Respuesta del Usuario:

1. Determine si la base de datos está habilitada.
2. Determine si la base de datos está dañada. Si está dañada, solicite al administrador de la base de datos su recuperación a partir de una copia de seguridad.

DXXA005I Se está habilitando la base de datos <*base_datos*>. Por favor, espere.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA006I La base de datos <*base_datos*> se ha habilitado satisfactoriamente.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA007E No se puede inhabilitar la base de datos <*base_datos*>.

Explicación: El XML Extender no puede inhabilitar la base de datos si ésta contiene alguna columna o colección XML.

Respuesta del Usuario: Haga una copia de seguridad de los datos importantes, inhabilite las columnas o colecciones XML existentes y actualice o elimine las tablas hasta que no quede ningún tipo de datos XML en la base de datos.

DXXA008I Se está inhabilitando la columna <*nombre_columna*>. Por favor, espere.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA009E El código Xcolumn no está especificado en el archivo DAD.

Explicación: Este procedimiento almacenado sólo es aplicable a Xcolumn.

Respuesta del Usuario: Asegúrese de que el código Xcolumn esté especificado correctamente en el archivo DAD.

DXXA010E Intente encontrar el ID de DTD <*iddtd*> que ha fallado.

Explicación: Mientras intentaba habilitar una columna, el XML Extender no pudo encontrar el ID de DTD, que es el identificador especificado para la DTD en el archivo de definición de acceso a documento (archivo DAD).

Respuesta del Usuario: Compruebe si está especificado el valor correcto para el ID de DTD en el archivo DAD.

DXXA011E No se ha podido insertar un registro en la tabla DB2XML.XML_USAGE.

Explicación: Mientras intentaba habilitar la columna, el XML no pudo insertar un registro en la tabla DB2XML.XML_USAGE.

Respuesta del Usuario: Asegúrese de que la tabla DB2XML.XML_USAGE existe y no contiene ya un registro del mismo nombre.

DXXA012E No se ha podido actualizar la tabla DB2XML.DTD_REF.

Explicación: Mientras intentaba habilitar una columna, el XML Extender no pudo actualizar la tabla DB2XML.DTD_REF.

Respuesta del Usuario: Asegúrese de que la

tabla DB2XML.DTD_REF existe. Determine si la tabla está dañada o si el ID de usuario de administración tiene la autorización correcta para actualizar la tabla.

DXXA013E No se ha podido modificar la tabla <*nombre_tabla*>.

Explicación: Mientras intentaba habilitar una columna, el XML Extender no pudo modificar la tabla especificada.

Respuesta del Usuario: Compruebe los privilegios necesarios para modificar la tabla.

DXXA014E La columna del ID raíz especificado: <*id_raíz*> no es una clave primaria simple de la tabla <*nombre_tabla*>.

Explicación: El ID raíz especificado no es una clave o no es una clave simple de la tabla *nombre_tabla*.

Respuesta del Usuario: Asegúrese de que el ID raíz especificado es la tabla primaria simple de la tabla.

DXXA015E La columna DXXROOT_ID ya existe en la tabla <*nombre_tabla*>.

Explicación: La columna DXXROOT_ID existe, pero no fue creada por el XML Extender.

Respuesta del Usuario: Especifique una columna de clave primaria para el ID raíz cuando habilite una columna, utilizando un nombre de columna diferente.

DXXA016E La tabla de entrada <*nombre_tabla*> no existe.

Explicación: El XML Extender no pudo encontrar en el catálogo del sistema la tabla especificada.

Respuesta del Usuario: Asegúrese de que la tabla existe en la base de datos y de que está especificada correctamente.

DXXA017E La columna de entrada <*nombre_columna*> no existe en la tabla especificada <*nombre_tabla*>.

Explicación: El XML Extender no pudo encontrar la columna en el catálogo del sistema.

Respuesta del Usuario: Asegúrese de que la columna existe en una tabla de usuario.

DXXA018E La columna especificada no está habilitada para datos XML.

Explicación: Mientras intentaba inhabilitar la columna, el XML Extender no pudo encontrar la columna en la tabla DB2XML.XML_USAGE, lo que indica que la columna no está habilitada. Si la columna no está habilitada para XML, no es necesario inhabilitarla.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA019E Un parámetro de entrada necesario para habilitar la columna es nulo.

Explicación: Un parámetro obligatorio de entrada del procedimiento almacenado enable_column() es nulo.

Respuesta del Usuario: Compruebe todos los parámetros de entrada del procedimiento almacenado enable_column().

DXXA020E No se pueden encontrar columnas en la tabla <*nombre_tabla*>.

Explicación: Mientras intentaba crear la vista predefinida, el XML Extender no pudo encontrar columnas en la tabla especificada.

Respuesta del Usuario: Compruebe si los nombres de columna y de tabla están especificados correctamente.

DXXA021E No se puede crear la vista predefinida <*vista_predefinida*>.

Explicación: Mientras intentaba habilitar una columna, el XML Extender no pudo crear la vista especificada.

Respuesta del Usuario: Compruebe si el nombre de la vista predefinida es exclusivo. Si ya existe una vista con el nombre especificado, especifique un nombre exclusivo para la vista predefinida.

DXXA022I La columna <*nombre_columna*> **está habilitada.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA023E No se puede encontrar el archivo DAD.

Explicación: Mientras intentaba inhabilitar una columna, el XML Extender no pudo encontrar el archivo de definición de acceso a documento (archivo DAD).

Respuesta del Usuario: Compruebe que ha especificado nombres correctos para la base de datos, la tabla y la columna.

DXXA024E El XML Extender ha encontrado un error interno al acceder a las tablas de catálogo del sistema.

Explicación: El XML Extender no pudo acceder a la tabla de catálogo del sistema.

Respuesta del Usuario: Compruebe que la base de datos está en un estado estable.

DXXA025E No se puede eliminar la vista predefinida <*vista_predefinida*>**.**

Explicación: Mientras intentaba inhabilitar una columna, el XML Extender no pudo eliminar la vista predefinida.

Respuesta del Usuario: Compruebe que el ID de usuario del administrador para el XML Extender tiene los privilegios necesarios para eliminar la vista predefinida.

DXXA026E No se puede eliminar la tabla auxiliar <*tabla_auxiliar*>**.**

Explicación: Mientras intentaba inhabilitar una columna, el XML Extender no pudo eliminar la tabla especificada.

Respuesta del Usuario: Compruebe que el ID de usuario del administrador para el XML Extender tiene los privilegios necesarios para eliminar la tabla.

DXXA027E No se pudo inhabilitar la columna.

Explicación: XML Extender no pudo inhabilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA028E No se pudo inhabilitar la columna.

Explicación: XML Extender no pudo inhabilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA029E No se pudo inhabilitar la columna.

Explicación: XML Extender no pudo inhabilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de

servicio de software y proporcione el archivo de rastreo.

DXXA030E No se pudo inhabilitar la columna.

Explicación: XML Extender no pudo inhabilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA031E No se puede redefinir el valor de la columna DXXROOT_ID como NULL en la tabla de aplicación.

Explicación: Mientras intentaba inhabilitar una columna, el XML Extender no pudo establecer en NULL el valor de DXXROOT_ID en la tabla de aplicación.

Respuesta del Usuario: Compruebe que el ID de usuario del administrador para el XML Extender tiene los privilegios necesarios para modificar la tabla de aplicación.

DXXA032E No se pudo disminuir USAGE_COUNT en la tabla DB2XML.XML_USAGE.

Explicación: Mientras intentaba inhabilitar una columna, el XML Extender no pudo disminuir en una unidad el valor de la columna USAGE_COUNT.

Respuesta del Usuario: Compruebe que la tabla DB2XML.XML_USAGE existe, y que el ID de usuario del administrador para el XML Extender tiene los privilegios necesarios para actualizar la tabla.

DXXA033E No se pudo suprimir una fila de la tabla DB2XML.XML_USAGE.

Explicación: Mientras intentaba inhabilitar una columna, el XML Extender no pudo suprimir su fila asociada en la tabla DB2XML.XML_USAGE.

Respuesta del Usuario: Compruebe que la tabla DB2XML.XML_USAGE existe, y que el ID de usuario del administrador para el XML Extender tiene los privilegios necesarios para actualizar la tabla.

DXXA034I El XML Extender ha inhabilitado satisfactoriamente la columna <*nombre_columna*>**.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA035I El XML Extender está inhabilitando la base de datos <*base_datos*>**. Por favor, espere.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA036I El XML Extender ha inhabilitado satisfactoriamente la base de datos <*base_datos*>**.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA037E El nombre especificado de espacio de tablas tiene más de 18 caracteres.

Explicación: El nombre del espacio de tablas no puede tener más de 18 caracteres alfanuméricos.

Respuesta del Usuario: Especifique un nombre de menos de 18 caracteres.

DXXA038E El nombre especificado de vista predefinida tiene más de 18 caracteres.

Explicación: El nombre de la vista predefinida no puede tener más de 18 caracteres alfanuméricos.

Respuesta del Usuario: Especifique un nombre de menos de 18 caracteres.

DXXA039E El nombre especificado para ROOT_ID tiene más de 18 caracteres.

Explicación: El nombre de ROOT_ID no puede tener más de 18 caracteres alfanuméricos.

Respuesta del Usuario: Especifique un nombre de menos de 18 caracteres.

DXXA046E No se puede crear la tabla auxiliar <*tabla_auxiliar*>.

Explicación: Mientras intentaba habilitar una columna, el XML Extender no pudo crear la tabla auxiliar especificada.

Respuesta del Usuario: Compruebe que el ID de usuario del administrador para el XML Extender tiene los privilegios necesarios para crear la tabla auxiliar.

DXXA047E No se pudo habilitar la columna.

Explicación: XML Extender no pudo habilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA048E No se pudo habilitar la columna.

Explicación: XML Extender no pudo habilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA049E No se pudo habilitar la columna.

Explicación: XML Extender no pudo habilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA050E No se pudo habilitar la columna.

Explicación: XML Extender no pudo habilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA051E No se pudo inhabilitar la columna.

Explicación: XML Extender no pudo inhabilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA052E No se pudo inhabilitar la columna.

Explicación: XML Extender no pudo inhabilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de

servicio de software y proporcione el archivo de rastreo.

DXXA053E No se pudo habilitar la columna.

Explicación: XML Extender no pudo habilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA054E No se pudo habilitar la columna.

Explicación: XML Extender no pudo habilitar una columna a causa de una anomalía en un desencadenante interno. Causas posibles:

Respuesta del Usuario: Utilice el recurso de rastreo para crear un archivo de rastreo e intente corregir el problema. Si el problema persiste, póngase en contacto con el Suministrador de servicio de software y proporcione el archivo de rastreo.

DXXA056E El valor de validación <*valor_validación*> **contenido en el archivo DAD no es válido.**

Explicación: El elemento de validación contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.

Respuesta del Usuario: Asegúrese de que el elemento de validación esté especificado correctamente en el archivo DAD.

DXXA057E El nombre de tabla auxiliar <*nombre_tabla_auxiliar*> **contenido en la DAD no es válido.**

Explicación: El atributo de nombre de una tabla auxiliar contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.

Respuesta del Usuario: Asegúrese de que el atributo de nombre de la tabla auxiliar esté especificado correctamente en el archivo DAD.

DXXA058E El nombre de columna <*nombre_columna*> **contenido en el archivo DAD no es válido.**

Explicación: El atributo de nombre de una columna contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.

Respuesta del Usuario: Asegúrese de que el atributo de nombre de la columna esté especificado correctamente en el archivo DAD.

DXXA059E El tipo <*tipo_columna*> **de la columna** <*nombre_columna*> **contenido en el archivo DAD no es válido.**

Explicación: El atributo de tipo de una columna contenido en el archivo de definición de acceso a documento (archivo DAD) es incorrecto o está ausente.

Respuesta del Usuario: Asegúrese de que el atributo de tipo de la columna esté especificado correctamente en el archivo DAD.

DXXA060E El atributo de vía de acceso <*location_path*> **de** <*nombre_columna*> **del archivo DAD no es válido.**

Explicación: El atributo de vía de acceso de una columna del archivo de definición de acceso a documento (DAD) es erróneo o no está presente.

Respuesta del Usuario: Asegúrese de que el atributo de vía de acceso de una columna se especifique correctamente en el archivo DAD.

DXXA061E El atributo multi_occurrence <*multi_occurrence*> **de** <*nombre_columna*> **contenido en el archivo DAD no es válido.**

Explicación: El atributo multi_occurrence de una columna contenido en el archivo de definición de acceso a documento (archivo DAD)

es incorrecto o está ausente.

Respuesta del Usuario: Asegúrese de que el atributo multi_occurrence de la columna esté especificado correctamente en el archivo DAD.

DXXA062E No se puede recuperar el número de columna para <*nombre_columna*> **en la tabla** <*nombre_tabla*>**.**

Explicación: El XML Extender no pudo recuperar en el catálogo del sistema el número de columna para *nombre_columna* de la tabla *nombre_tabla*.

Respuesta del Usuario: Compruebe que la tabla de aplicación esté bien definida.

DXXA063I Se está habilitando la colección <*nombre_colección*>**. Por favor, espere.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA064I Se está inhabilitando la colección <*nombre_colección*>**. Por favor, espere.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA065E Ha fallado la llamada al procedimiento almacenado <*nombre_procedimiento*>**.**

Explicación: Examine la biblioteca compartida db2xml y compruebe si el permiso es correcto.

Respuesta del Usuario: Compruebe que el cliente tenga permiso para ejecutar el procedimiento almacenado.

DXXA066I El XML Extender ha inhabilitado satisfactoriamente la colección <*nombre_colección*>**.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA067I El XML Extender ha habilitado satisfactoriamente la colección <*nombre_colección*>**.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA068I El XML Extender ha activado satisfactoriamente el rastreo.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA069I El XML Extender ha desactivado satisfactoriamente el rastreo.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA070W La base de datos ya está habilitada.

Explicación: Se ha ejecutado el mandato enable database para una base de datos habilitada.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA071W La base de datos ya está inhabilitada.

Explicación: Se ha ejecutado el mandato disable database para una base de datos inhabilitada.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA072E El XML Extender no pudo encontrar los archivos de enlace. Enlace la base de datos antes de habilitarla.

Explicación: El XML Extender ha intentado enlazar automáticamente la base de datos antes de habilitarla, pero no ha encontrado los archivos de enlace

Respuesta del Usuario: Enlace la base de datos antes de habilitarla.

DXXA073E La base de datos no está enlazada. Por favor, enlace la base de datos antes de habilitarla.

Explicación: La base de datos no estaba enlazada cuando el usuario ha intentado habilitarla.

Respuesta del Usuario: Enlace la base de datos antes de habilitarla.

DXXA074E El tipo de parámetro es erróneo. El procedimiento almacenado espera un parámetro STRING.

Explicación: El procedimiento almacenado espera un parámetro STRING.

Respuesta del Usuario: Declare el parámetro de entrada para que sea del tipo STRING.

DXXA075E El tipo de parámetro es erróneo. El parámetro de entrada debe ser del tipo LONG.

Explicación: El procedimiento almacenado espera que el parámetro de entrada sea del tipo LONG.

Respuesta del Usuario: Declare el parámetro de entrada para que sea del tipo LONG.

DXXA076E ID de instancia de rastreo de XML Extender no válido.

Explicación: No se puede iniciar el rastreo con el ID de instancia que se ha proporcionado.

Respuesta del Usuario: Asegúrese de que el ID de instancia es un ID de usuario de AS/400 válido.

DXXC000E No se puede abrir el archivo especificado.

Explicación: El XML Extender no puede abrir el archivo especificado.

Respuesta del Usuario: Compruebe que el ID de usuario de la aplicación tiene permiso de lectura y escritura para el archivo.

DXXC001E No se encuentra el archivo especificado.

Explicación: El XML Extender no pudo encontrar el archivo especificado.

Respuesta del Usuario: Compruebe que el archivo existe y que la vía de acceso está especificada correctamente.

DXXC002E No se puede leer el archivo.

Explicación: El XML Extender no puede leer datos del archivo especificado.

Respuesta del Usuario: Compruebe que el ID de usuario de la aplicación tiene permiso de lectura para el archivo.

DXXC003E No se puede escribir en el archivo especificado.

Explicación: El XML Extender no puede escribir datos en el archivo.

Respuesta del Usuario: Compruebe que el ID de usuario de la aplicación tiene permiso de escritura para el archivo y que el sistema de archivos tiene espacio suficiente.

DXXC004E No se puede utilizar el Localizador de LOB: cr=<*cr_localizador*>.

Explicación: El XML Extender no pudo utilizar el localizador especificado.

Respuesta del Usuario: Compruebe que el Localizador de LOB está definido correctamente.

DXXC005E El tamaño del archivo de entrada es mayor que el tamaño de XMLVarchar.

Explicación: El tamaño del archivo es mayor que el tamaño de XMLVarchar y el XML Extender no puede importar todos los datos del archivo.

Respuesta del Usuario: Utilice el tipo de columna XMLCLOB.

DXXC006E El archivo de entrada excede el límite de DB2 para LOB.

Explicación: El tamaño del archivo es mayor que el tamaño de XMLCLOB y el XML Extender no puede importar todos los datos del archivo.

Respuesta del Usuario: Descomponga el archivo en objetos más pequeños o utilice una colección XML.

DXXC007E No se pueden recuperar datos del archivo y colocarlos en el Localizador de LOB.

Explicación: El número de bytes del Localizador de LOB no es igual al tamaño del archivo.

Respuesta del Usuario: Compruebe que el Localizador de LOB está definido correctamente.

DXXC008E No se puede eliminar el archivo <*nombre_archivo*>.

Explicación: Existe una violación de acceso compartido para el archivo o todavía está abierto.

Respuesta del Usuario: Cierre el archivo o detenga los procesos que retienen el archivo. Es posible que tenga que detener DB2 y volverlo a iniciar.

DXXC009E No se puede crear el archivo en el directorio <*directorio*>.

Explicación: El XML Extender no puede crear un archivo en el directorio *directorio*.

Respuesta del Usuario: Compruebe que el directorio existe, que el ID de usuario de la aplicación tiene permiso de escritura para el directorio y que el sistema de archivos tiene suficiente espacio para el archivo.

DXXC010E Error al escribir en el archivo <*nombre_archivo*>.

Explicación: Se ha producido un error al escribir en el archivo *nombre_archivo*.

Respuesta del Usuario: Compruebe que el sistema de archivos tenga suficiente espacio para el archivo.

DXXC011E No se puede escribir en el archivo de control de rastreo.

Explicación: El XML Extender no puede escribir datos en el archivo de control de rastreo.

Respuesta del Usuario: Compruebe que el ID de usuario de la aplicación tiene permiso de escritura para el archivo y que el sistema de archivos tiene espacio suficiente.

DXXC012E No se puede crear el archivo temporal.

Explicación: No se puede crear el archivo en el directorio temporal del sistema.

Respuesta del Usuario: Compruebe que el ID de usuario de la aplicación tiene permiso de escritura para el directorio temporal del sistema de archivos o que el sistema de archivos tiene espacio suficiente para el archivo.

DXXD000E Se ha rechazado un documento XML no válido.

Explicación: Se ha intentado almacenar un documento no válido en una tabla. La validación ha fallado.

Respuesta del Usuario: Compruebe el documento por comparación con su DTD, utilizando un editor que pueda visualizar caracteres no válidos ocultos. Para corregir este error, desactive la validación en el archivo DAD.

DXXD001E *<vía_ubicación>* **aparece varias veces.**

Explicación: Una función de extracción escalar ha utilizado una vía de ubicación que aparece varia veces. Una función escalar sólo puede utilizar una vía de ubicación que no tiene aparición múltiple.

Respuesta del Usuario: Utilice una función de tabla (añada una 's' al final del nombre de función escalar).

DXXD002E **Se ha producido un error de sintaxis cerca de la posición** *<posición>* **de la vía de búsqueda.**

Explicación: La expresión de la vía de búsqueda tiene una sintaxis incorrecta.

Respuesta del Usuario: Corrija el argumento de vía de búsqueda en la consulta. Consulte la documentación para conocer la sintaxis de las expresiones de vía de búsqueda.

DXXD003W **No se ha encontrado la vía de acceso. Se devuelve un valor nulo.**

Explicación: El elemento o atributo especificado en la expresión de vía de acceso no está en el documento XML.

Respuesta del Usuario: Compruebe que la vía de acceso especificada es correcta.

DXXG000E **El nombre de archivo** *<nombre_archivo>* **no es válido.**

Explicación: Se ha especificado un nombre de archivo no válido.

Respuesta del Usuario: Especifique un nombre de archivo correcto y vuelva a intentarlo.

DXXG001E **Se ha producido un error interno en la creación** *<ID_creación>***, archivo** *<nombre_archivo>* **y línea** *<número_línea>***.**

Explicación: El XML Extender ha encontrado un error interno.

Respuesta del Usuario: Póngase en contacto con el Suministrador de Servicio de Software. Cuando informe del error, asegúrese de incluir todos los mensajes, el archivo de rastreo y la explicación que permita reproducir el error.

DXXG002E **Falta memoria en el sistema.**

Explicación: El XML Extender no pudo asignar memoria del sistema operativo.

Respuesta del Usuario: Cierre algunas aplicaciones y vuelva a intentarlo. Si el problema persiste, consulte la documentación del sistema operativo para obtener ayuda. Algunos sistemas operativos necesitan que el usuario rearranque el sistema para corregir el problema.

DXXG004E **Parámetro nulo no válido.**

Explicación: Se ha pasado un valor nulo a un procedimiento almacenado de XML para un parámetro obligatorio.

Respuesta del Usuario: Compruebe todos los parámetros obligatorios especificados en la lista de argumentos de la llamada de procedimiento almacenado.

DXXG005E **Parámetro no soportado.**

Explicación: Este parámetro no está soportado en este release; estará soportado en el release futuro.

Respuesta del Usuario: Establezca este parámetro en NULL.

DXXG006E **Error interno ESTADOCLI=***<estadocli>***, CR=***<cr_cli>***, creación** *<ID_creación>***, archivo** *<nombre_archivo>***, línea** *<número_línea>* **MSGCLI=***<msg_CLI>***.**

Explicación: El XML Extender ha encontrado un error interno mientras utilizaba CLI.

Respuesta del Usuario: Póngase en contacto con el Suministrador de Servicio de Software. Es posible que este error lo haya causado una

entrada incorrecta del usuario. Cuando informe del error, asegúrese de incluir todos los mensajes de salida, el registro de rastreo y la explicación que permita reproducir el problema. Si es posible, envíe las DAD, los documentos XML y las definiciones de tabla que se apliquen al caso.

DXXG007E El entorno local <*entorno_local*> no es coherente con la página de códigos <*página_códigos*> de DB2.

Explicación: El entorno local del sistema operativo del servidor no es coherente con la página de códigos de DB2.

Respuesta del Usuario: Corrija el entorno local del sistema operativo del servidor y reinicie DB2.

DXXG008E El entorno local <*entorno_local*> no está soportado.

Explicación: El entorno local del sistema operativo del servidor no se puede encontrar en la tabla de páginas de códigos.

Respuesta del Usuario: Corrija el entorno local del sistema operativo del servidor y reinicie DB2.

DXXQ000E El elemento <*elemento*> falta en el archivo DAD.

Explicación: Falta un elemento obligatorio en el archivo de definición de acceso a documento (archivo DAD).

Respuesta del Usuario: Añada el elemento faltante al archivo DAD.

DXXQ001E Sentencia de SQL no válida para la generación XML.

Explicación: La sentencia de SQL contenida en la definición de acceso a documento (DAD) o la sentencia que la sustituye no es válida. Es necesaria una sentencia SELECT para generar documentos XML.

Respuesta del Usuario: Corrija la sentencia de SQL.

DXXQ002E No se puede crear espacio de almacenamiento para contener documentos XML.

Explicación: El sistema se ha quedado sin espacio de memoria o de disco. No hay espacio para contener los documentos XML resultantes.

Respuesta del Usuario: Limite el número de documentos a crear. Disminuya el tamaño de cada documento eliminando algunos nodos de elementos o atributos innecesarios en el archivo de definición de acceso a documento (archivo DAD).

DXXQ003W El resultado excede el máximo.

Explicación: La consulta SQL definida por el usuario genera más documentos XML que el máximo especificado. Sólo se devuelve el número de documentos especificado.

Respuesta del Usuario: No es necesaria ninguna acción.Si necesita todos los documentos, especifique 0 como número máximo de documentos.

DXXQ004E La columna <*nombre_columna*> no está en el resultado de la consulta.

Explicación: La columna especificada no es una de las columnas incluidas en el resultado de la consulta SQL.

Respuesta del Usuario: En el archivo de definición de acceso a documento (archivo DAD), cambie el nombre de columna especificado por el nombre de una de las columnas del resultado de la consulta SQL. O bien, modifique la consulta SQL para que su resultado tenga la columna especificada.

DXXQ004W No se ha encontrado el ID de DTD en la DAD.

Explicación: En la DAD, VALIDATION es YES, pero el elemento DTDID no se ha especificado. No se realiza ninguna comprobación de validez.

Respuesta del Usuario: No es necesaria ninguna acción. Si es necesario realizar la

validación, especifique el elemento DTDID en el archivo DAD.

DXXQ005E Correlación relacional incorrecta. El elemento <*nombre_elemento*> **está en un nivel inferior que su columna hija** <*nombre_columna*>**.**

Explicación: La correlación de la consulta SQL con XML es incorrecta.

Respuesta del Usuario: Asegúrese de que las columnas del resultado de la consulta SQL están en orden descendente en la jerarquía relacional. Asegúrese también de que existe una clave adecuada de columna individual para comenzar cada nivel. Si no existe una clave así en una tabla, la consulta debe generar la clave para esa tabla utilizando una expresión de tabla y la función interna de DB2 generate_unique().

DXXQ006E Un elemento nodo_atributo no tiene nombre.

Explicación: Un elemento nodo_atributo contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un atributo de nombre.

Respuesta del Usuario: Asegúrese de que cada nodo_atributo tenga un nombre en el archivo DAD.

DXXQ007E El nodo_atributo <*nombre_atributo*> **no tiene ningún elemento de columna ni nodo_RDB.**

Explicación: El elemento nodo_atributo contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un elemento de columna ni un nodo_RDB.

Respuesta del Usuario: Asegúrese de que cada nodo_atributo tenga un elemento de columna o nodo_RDB en el archivo DAD.

DXXQ008E Un elemento nodo_texto no tiene ningún elemento de columna.

Explicación: Un elemento nodo_texto contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un elemento de columna.

Respuesta del Usuario: Asegúrese de que cada nodo_texto tenga un elemento de columna en el archivo DAD.

DXXQ009E La tabla de resultados <*nombre_tabla*> **no existe.**

Explicación: La tabla de resultados especificada no se ha podido encontrar en el catálogo del sistema.

Respuesta del Usuario: Cree la tabla de resultados antes de llamar al procedimiento almacenado.

DXXQ010E El nodo_RDB de <*nombre_nodo*> **no tiene una tabla en el archivo DAD.**

Explicación: El nodo_RDB del nodo_atributo o del nodo_texto debe tener una tabla.

Respuesta del Usuario: Especifique la tabla de nodo_RDB para nodo_atributo o nodo_texto en el archivo de definición de acceso a documento (archivo DAD).

DXXQ011E El elemento nodo_RDB de <*nombre_nodo*> **no tiene una columna en el archivo DAD.**

Explicación: El nodo_RDB del nodo_atributo o del nodo_texto debe tener una columna.

Respuesta del Usuario: Especifique la columna de nodo_RDB para nodo_atributo o nodo_texto en el archivo de definición de acceso a documento (archivo DAD).

DXXQ012E Se han producido errores en la DAD.

Explicación: El XML Extender no pudo encontrar el elemento esperado mientras procesaba la DAD.

Respuesta del Usuario: Compruebe que la DAD es un documento XML válido y que contiene todos los elementos necesarios para la

DTD de DAD. Consulte la publicación de XML Extender para obtener información sobre la DTD de DAD.

DXXQ013E El elemento de tabla o columna no tiene un nombre en el archivo DAD.

Explicación: El elemento de tabla o columna debe tener un nombre en el archivo de definición de acceso a documento (archivo DAD).

Respuesta del Usuario: Especifique el nombre del elemento de tabla o columna en el archivo DAD.

DXXQ014E Un elemento nodo_elemento no tiene nombre.

Explicación: Un elemento nodo_elemento contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un atributo de nombre.

Respuesta del Usuario: Asegúrese de que cada elemento nodo_elemento tenga un nombre en el archivo DAD.

DXXQ015E El formato de la condición no es válido.

Explicación: La condición expresada en el elemento de condición, contenido en la definición de acceso a documento (DAD), tiene un formato no válido.

Respuesta del Usuario: Asegúrese de que el formato de la condición sea válido.

DXXQ016E El nombre de tabla de este nodo_RDB no está definido en el elemento superior del archivo DAD.

Explicación: Todas las tablas deben estar definidas en el nodo_RDB del elemento superior, en el archivo de definición de acceso a documento (archivo DAD). Las tablas de sub-elemento deben coincidir con las tablas definidas en el elemento superior. El nombre de tabla de este nodo_RDB no está en el elemento superior.

Respuesta del Usuario: Asegúrese de que la tabla del nodo RDB esté definida en el elemento superior del archivo DAD.

DXXQ017E La columna de la tabla de resultados <*nombre_tabla*> es demasiado pequeña.

Explicación: El XML Extender ha generado un documento XML que es demasiado grande para caber en la columna de la tabla de resultados.

Respuesta del Usuario: Elimine la tabla de resultados. Cree otra tabla de resultados con una columna mayor. Ejecute otra vez el procedimiento almacenado.

DXXQ018E Falta la cláusula ORDER BY en la sentencia de SQL.

Explicación: Falta la cláusula ORDER BY en la sentencia de SQL que correlaciona SQL con XML, y que está contenida en un archivo de definición de acceso a documento (archivo DAD).

Respuesta del Usuario: Edite el archivo DAD. Añada una cláusula ORDER BY que contenga las columnas identificadoras de entidades.

DXXQ019E El elemento objids no tiene ningún elemento de columna en el archivo DAD.

Explicación: El elemento objids no tiene un elemento de columna en el archivo DAD (archivo de definición de acceso a documento) que correlaciona SQL con XML.

Respuesta del Usuario: Edite el archivo DAD. Añada las columnas de clave como sub-elementos del elemento objids.

DXXQ020I XML generado satisfactoriamente.

Explicación: Los documentos XML solicitados se han generado satisfactoriamente a partir de la base de datos.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXQ021E **La tabla** <*nombre_tabla*> **no tiene la columna** <*nombre_columna*>**.**

Explicación: La base de datos de la tabla no tiene la columna especificada.

Respuesta del Usuario: Especifique otro nombre de columna o añada la columna especificada a la base de datos de la tabla.

DXXQ022E **La columna** <*nombre_columna*> **de** <*nombre_tabla*> **debe ser del tipo** <*nombre_tipo*>**.**

Explicación: El tipo de la columna es incorrecto.

Respuesta del Usuario: Corrija el tipo de la columna en la definición de acceso a documento (DAD).

DXXQ023E **La columna** <*nombre_columna*> **de** <*nombre_tabla*> **no puede ser mayor que** <*longitud*>**.**

Explicación: La longitud definida para la columna en la DAD es demasiado grande.

Respuesta del Usuario: Corrija la longitud de la columna en la definición de acceso a documento (DAD).

DXXQ024E **No se puede crear la tabla** <*nombre_tabla*>**.**

Explicación: No se puede crear la tabla especificada.

Respuesta del Usuario: Compruebe que el ID de usuario que está creando la tabla tiene la autorización necesaria para crear una tabla en la base de datos.

DXXQ025I **XML descompuesto satisfactoriamente.**

Explicación: Se ha descompuesto un documento XML y se ha almacenado en una colección satisfactoriamente.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXQ026E **Los datos de XML** <*nombre_xml*> **son demasiado grandes para caber en la columna** <*nombre_columna*>**.**

Explicación: La porción de datos especificada de un documento XML es demasiado grande para caber en la columna especificada.

Respuesta del Usuario: Aumente la longitud de la columna utilizando la sentencia ALTER TABLE o disminuya el tamaño de los datos editando el documento XML.

DXXQ028E **No se ha encontrado la colección** <*nombre_colección*> **en la tabla XML_USAGE.**

Explicación: No se ha encontrado un registro para la colección en la tabla XML_USAGE.

Respuesta del Usuario: Compruebe que ha habilitado la colección.

DXXQ029E **No se ha encontrado la DAD en la tabla XML_USAGE para la colección** <*nombre_colección*>**.**

Explicación: No se ha encontrado un registro de DAD para la colección en la tabla XML_USAGE.

Respuesta del Usuario: Compruebe que ha habilitado la colección correctamente.

DXXQ030E **Sintaxis errónea de alteración temporal de XML.**

Explicación: El valor de XML_override se ha especificado incorrectamente en el procedimiento almacenado.

Respuesta del Usuario: Compruebe que la sintaxis de XML_override sea correcta.

DXXQ031E El nombre de tabla no puede ser mayor que la longitud máxima permitida por DB2.

Explicación: El nombre de tabla especificado por el elemento de condición en la DAD es demasiado largo.

Respuesta del Usuario: Corrija la longitud del nombre de tabla en la definición de acceso a documento (DAD).

DXXQ032E El nombre de columna no puede ser mayor que la longitud permitida por DB2.

Explicación: El nombre de columna especificado por el elemento de condición en la DAD es demasiado largo.

Respuesta del Usuario: Corrija la longitud del nombre de columna en la definición de acceso a documento (DAD).

DXXQ033E El identificador no válido empieza en *<identificador>*

Explicación: La serie no es un identificador de SQL de DB2 válido.

Respuesta del Usuario: Corrija la serie en la DAD para ajustarse a las normas para los identificadores de SQL de DB2.

DXXQ034E Elemento de condición no válido en el nodo_RDB superior de DAD: *<condición>*

Explicación: El elemento de condición debe ser una cláusula WHERE que conste de condiciones de unión conectadas mediante la conjunción AND.

Respuesta del Usuario: Consulte la documentación del XML Extender para obtener la sintaxis correcta de la condición de unión en una DAD.

DXXQ035E Condición de unión no válida en el nodo_RDB superior de DAD: *<condición>*

Explicación: Los nombres de columna del elemento de condición del nodo_RDB superior deben estar calificados con el nombre de tabla si la DAD especifica múltiples tablas.

Respuesta del Usuario: Consulte la documentación del XML Extender para obtener la sintaxis correcta de la condición de unión en una DAD.

DXXQ036E Un nombre de Esquema especificado bajo un código de condición de DAD tiene una longitud mayor que la permitida.

Explicación: Se ha detectado un error al analizar texto bajo un código de condición en la DAD. El texto de condición contiene un ID calificado por un nombre de esquema demasiado largo.

Respuesta del Usuario: Corrija el texto de los códigos de condición en la definición de acceso a documento (DAD).

DXXQ037E No se puede generar *<elemento>* **con múltiples apariciones.**

Explicación: El nodo de elemento y sus descendientes no tienen ninguna correlación con la base de datos, pero el valor multi_ocurrence es igual a YES.

Respuesta del Usuario: Corrija la DAD definiendo multi_occurrence como NO o cree un nodo_RDB en uno de sus descendientes.

Rastreo de diagnóstico

El XML Extender incluye un recurso de rastreo que registra la actividad de los servidores del XML Extender. Debe utilizar el recurso de rastreo sólo de acuerdo con las instrucciones del Centro de Soporte de Software de IBM.

El recurso de rastreo anota información sobre diversos sucesos en un archivo de servidor; estos sucesos pueden ser, por ejemplo, el inicio o fin de la ejecución de un componente del XML Extender o la devolución de un código de error por parte de un componente del XML Extender. Debido a que el recurso de rastreo anota información referente a muchos sucesos, sólo debe utilizarse cuando sea necesario, por ejemplo, al investigar condiciones de error. Además, es aconsejable que limite el número de aplicaciones activas cuando utilice el recurso de rastreo. De esta forma puede ser más fácil identificar la causa de un problema.

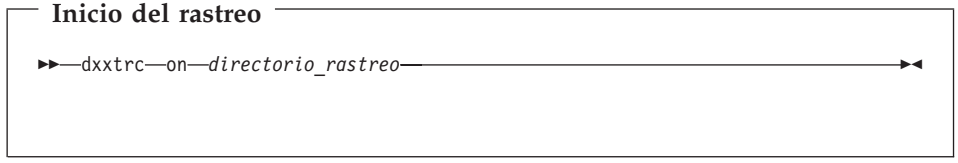
Utilice el mandato **dxxtrc** para iniciar o detener el rastreo. Puede emitir el mandato desde una línea de mandatos en un servidor AIX, Windows NT o Solaris. Debe tener autorización SYSADM, SYSCTRL o SYSMINT para emitir el mandato.

Inicio del rastreo

Propósito

Registra la actividad de los servidores del XML Extender. Para iniciar el rastreo, aplique la opción `on` en **dxxtrc**, junto con el nombre del directorio que contiene el archivo de rastreo. Cuando se activa el rastreo, el archivo, `dxx`*`INSTANCE`*`.trc` se coloca en el directorio especificado. *INSTANCE* es el valor de DB2INSTANCE. Cada instancia de DB2 tiene su propio archivo de anotaciones.

Formato



Parámetros

Tabla 54. Parámetros de rastreo

Parámetro	Descripción
directorio_rastreo	Nombre del directorio donde se coloca `dxx`*`INSTANCE`*`.trc`. Es obligatorio; no es el valor por omisión.

Ejemplos

El siguiente ejemplo muestra el inicio del rastreo para una instancia db2inst1 en AIX. El archivo de rastreo, `dxxdb2inst1.trc`, se coloca en el directorio `/home/db2inst1/sqllib/log`.

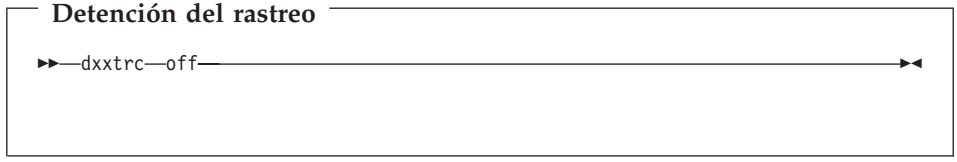
```
dxxtrc on /home/db2inst1/sqllib/log
```

Detención del rastreo

Propósito

Desactiva el rastreo. No se anota ninguna información de rastreo. Debido a que la ejecución del rastreo puede afectar al rendimiento, es aconsejable desactivar el rastreo cuando se trabaja en un entorno de producción.

Formato



Ejemplos

Este ejemplo muestra que el recurso de rastreo se desactiva.

```
dxxtrc off
```

Parte 5. Apéndices

Apéndice A. DTD para el archivo DAD

Esta sección describe las DTD (declaraciones de tipo de documento) del archivo DAD (archivo de definición de acceso a documento). El propio archivo DAD es un documento XML con estructura arborescente y necesita una DTD. El nombre del archivo DTD es `dxxdad.dtd`. La Figura 13 en la página 268 muestra la DTD del archivo DAD. Los elementos de este archivo se describen en la figura siguiente.

```
<?xml encoding="US-ASCII"?>

  <!ELEMENT DAD (dtdid?, validation, (Xcolumn | Xcollection))>
  <!ELEMENT dtdid (#PCDATA)>
  <!ELEMENT validation (#PCDATA)>
  <!ELEMENT Xcolumn (table*)>
  <!ELEMENT table (column*)>
  <!ATTLIST table name CDATA #REQUIRED
                              key CDATA #IMPLIED
                              orderBy CDATA #IMPLIED>
  <!ELEMENT column EMPTY>
  <!ATTLIST column
                      name CDATA #REQUIRED
                      type CDATA #IMPLIED
                      path CDATA #IMPLIED
                       multi_occurrence CDATA #IMPLIED>
  <!ELEMENT Xcollection (SQL_stmt?, objids?, prolog, doctype, root_node)>
  <!ELEMENT SQL_stmt (#PCDATA)>
  <!ELEMENT objids (column+)>
  <!ELEMENT prolog (#PCDATA)>
  <!ELEMENT doctype (#PCDATA | RDB_node)*>
  <!ELEMENT root_node (element_node)>
  <!ELEMENT element_node (RDB_node*,
                                       attribute_node*,
                                       text_node?,
                                       element_node*,
                                       namespace_node*,
                                       process_instruction_node*,
                                       comment_node*)>
  <!ATTLIST element_node
                     name CDATA #REQUIRED
                     ID CDATA #IMPLIED
                     multi_occurrence CDATA "NO"
                     BASE_URI CDATA #IMPLIED>
  <!ELEMENT attribute_node (column | RDB_node)>
  <!ATTLIST attribute_node
                     name CDATA #REQUIRED>
  <!ELEMENT text_node (column | RDB_node)>
  <!ELEMENT RDB_node (table+, column?, condition?)>
  <!ELEMENT condition (#PCDATA)>
  <!ELEMENT comment_node (#PCDATA)>
  <!ELEMENT namespace_node (EMPTY)>
  <!ATTLIST namespace_node
                      name CDATA #IMPLIED
                     value CDATA #IMPLIED>
  <!ELEMENT process_instruction_node (#PCDATA)>
```

Figura 13. Declaraciones DTD de la DAD (definición de acceso a documento)

El archivo DAD tiene cuatro elementos principales:

- DTDID
- validation
- Xcolumn

- Xcollection

Xcolumn y Xcollection tienen elementos y atributos asociados que ayudan a correlacionar datos XML con tablas relacionales de DB2. La lista siguiente describe los elementos principales y sus elementos y atributos asociados. Los ejemplos de sintaxis proceden de la Figura 13 en la página 268.

elemento DTDID
Especifica el ID de la DTD almacenada en la tabla DTD_REF. `DTDID` apunta a la DTD que valida los documentos XML o dirige la correlación entre tablas de colección XML y documentos XML. Es obligatorio especificar el `DTDID` para colecciones XML. Es opcional especificarlo para columnas XML y sólo es necesario si desea crear tablas auxiliares para indexar elementos o atributos, o si desea validar documentos XML de entrada. `DTDID` debe ser igual al ID de sistema (SYSTEM ID) especificado en `doctype` para los documentos XML.

Sintaxis: `<!ELEMENT dtdid (#PCDATA)>`

elemento validation
Indica si el documento XML se debe validar mediante la DTD de la DAD. Si se especifica YES, entonces también debe especificarse el `DTDID`.

Sintaxis: `<!ELEMENT validation(#PCDATA)>`

elemento Xcolumn
Define el esquema de indexación para una columna XML. Consta de 0, 1 o más tablas.

Sintaxis: `<!ELEMENT Xcolumn (table*)>Xcolumn` tiene un elemento hijo, `table`.

elemento table
Define una o más tablas relacionales creadas para indexar elementos o atributos de documentos contenidos en una columna XML.

Sintaxis:

```
<!ELEMENT table (column+)>
 <!ATTLIST table name CDATA #REQUIRED
key CDATA #IMPLIED
orderBy CDATA #IMPLIED>
```

El elemento `table` tiene un solo atributo:

atributo name
Especifica el nombre de la tabla auxiliar.

El elemento `table` tiene un solo elemento hijo:

atributo key
Es la clave única primaria de la tabla

atributo orderBy

Son los nombres de las columnas que determinan el orden secuencial de elementos o atributos que aparecen varias veces al generar documentos XML.

elemento column

Especifica la columna de la tabla que contiene el valor de una vía de ubicación del tipo especificado.

Sintaxis:

```
<!ATTLIST column
                      name CDATA #REQUIRED
                      type  CDATA #IMPLIED
                      path  CDATA #IMPLIED
                      multi_occurrence CDATA #IMPLIED>
```

El elemento `column` tiene los atributos siguientes:

atributo name

Especifica el nombre de la columna. Es el seudónimo de la vía de ubicación que sirve para identificar un elemento o atributo.

atributo type

Define el tipo de datos de la columna. Puede ser un tipo de datos cualquiera del SQL.

atributo path

Muestra la vía de ubicación de un elemento o atributo XML y debe ser la vía de ubicación simple tal como está especificada en la Tabla 3.1.a (enlace fijo).

atributo multi_occurrence

Indica si el elemento o atributo puede aparecer más de una vez en un documento XLM. Los valores son YES o NO.

Xcollection

Define la correlación existente entre documentos XML y una colección XML de tablas relacionales.

Sintaxis: `<!ELEMENT Xcollection(SQL_stmt*, prolog, doctype, root_node)>Xcollection` tiene los elementos asociados siguientes:

SQL_stmt

Especifica la sentencia de SQL que el XML Extender utiliza para definir la colección. En concreto, la sentencia selecciona datos XML de las tablas de colección XML, y utiliza los datos para generar los documentos XML de la colección. El valor de este elemento debe ser una sentencia de SQL válida. Sólo se utiliza en la generación de documentos, y sólo está permitido un único valor de `SQL_stmt`. Para la descomposición de

documentos, se puede especificar más de un valor de SQL_stmt para realizar la creación e inserción de tablas necesaria.

Sintaxis: `<!ELEMENT SQL_stmt  #PCDATA >`

prolog Es el texto del prólogo XML. Se proporciona el mismo prólogo a todos los documentos de la colección completa. El valor de prolog es fijo.

Sintaxis: `<!ELEMENT prolog  #PCDATA>`

doctype
Define el texto para la definición del tipo de documento XML.

Sintaxis: `<!ELEMENT doctype  #PCDATA | RDB_node>` doctype se puede especificar en una de las maneras siguientes:

- Para definir un valor explícito. Este valor se proporciona a todos los documentos de la colección completa.
- Cuando se utiliza la descomposición de documentos, para especificar el elemento hijo RDB_node, que se puede correlacionar con datos de columna de una tabla y almacenar como datos de columna.

doctype tiene un solo elemento hijo:

RDB_node
No está implantado todavía.

root_node
Define el nodo raíz virtual. root_node debe tener un elemento hijo obligatorio, element_node, que se puede utilizar una sola vez. El elemento element_node situado dentro de root_node es en realidad el root_node del documento XML.

Sintaxis: `<!ELEMENT root_node(element_node)>`

element_node
Representa un elemento XML. Debe estar definido en la DTD especificada para la colección. Si se utiliza la correlación de nodo_RDB, el nodo de elemento raíz debe tener un nodo_RDB para especificar todas las tablas que contienen datos XML para el nodo raíz y todos sus nodos asociados. Puede haber 0, 1 o más nodos de atributo y nodos de elemento asociados, así como 0 ó 1 nodos de texto. Para los elementos que no sean el elemento raíz no es necesario ningún nodo RDB.

Sintaxis:

Un nodo de elemento está definido por los elementos asociados siguientes:

RDB_node
(Opcional) Especifica tablas, columnas y condiciones para datos XML. El RDB_node de un elemento sólo necesita definirse para la correlación de nodo_RDB. En este caso, se deben especificar una o más tablas. La columna no es necesaria, pues el contenido del elemento está especificado por su text_node. La condición es opcional, dependiendo de la DTD y de la condición de consulta.

nodos asociados
(Opcional) Un `element_node` pueden también tener los nodos asociados siguientes:

element_node
Representa elementos asociados del elemento XML actual

attribute_node
Representa atributos del elemento XML actual

text_node
Representa el texto CDATA del elemento XML actual

attribute_node
Representa un atributo XML. Es el nodo que define la correlación entre un atributo XML y los datos de columna de una tabla relacional.

Sintaxis:

El `attribute_node` debe tener definiciones para un atributo `name` y un elemento hijo `column` o `RDB_node`. `attribute_node` tiene el atributo siguiente:

name Es el nombre del atributo.

`attribute_node` tiene los elementos asociados siguientes:

Column
Se utiliza para la correlación SQL. La columna se debe especificar en la cláusula SELECT de `SQL_stmt`.

RDB_node
Se utiliza para la correlación de nodo_RDB. El nodo define la correlación entre el atributo y los datos de columna de la tabla relacional. Se deben especificar la tabla y la columna. La condición es opcional.

text_node

Representa el contenido de texto de un elemento XML. Es el nodo que define la correlación entre el contenido de un elemento XML y los datos de columna de una tabla relacional.

Sintaxis: Debe estar definido por un elemento hijo `column` o `RDB_node`:

Column

Es necesario para la correlación SQL. En este caso, la columna se debe especificar en la cláusula SELECT de `SQL_stmt`.

RDB_node

Es necesario para la correlación de nodo_RDB. El nodo define la correlación entre el contenido de texto y los datos de columna de la tabla relacional. Se deben especificar la `tabla` y la `columna`. La condición es opcional.

Apéndice B. Ejemplos

Este apéndice muestra los objetos de ejemplo que se utilizan en los ejemplos de este manual.

DTD de XML

La siguiente DTD se utiliza para el documento `getstart.xml`, el cual aparece en todo este manual y se muestra en la Figura 15 en la página 276.

```
<!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)>
```

Figura 14. DTD de XML de ejemplo: getstart.dtd

Documento XML: getstart.xml

El siguiente documento XML, `getstart.xml`, es el documento XML de ejemplo que se utiliza en los ejemplos de este manual. Contiene los códigos XML para formar una orden de compra.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd">
<Order key="1">
  <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>
  <Part color="black ">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
  </Part>
  <Part color="red   ">
    <key>128</key>
    <Quantity>28</Quantity>
    <ExtendedPrice>38000.00</ExtendedPrice>
    <Tax>7.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-12-30</ShipDate>
      <ShipMode>TRUCK </ShipMode>
    </Shipment>
  </Part>
</Order>
```

Figura 15. Documento XML de ejemplo: `getstart.xml`

Archivos de definición de acceso a documento

Las secciones siguientes contienen archivos de definición de acceso a documento (archivos DAD), que correlacionan datos XML con tablas relacionales DB2, utilizando las modalidades de acceso para columnas XML o para colecciones XML.

- "Archivo DAD: columna XML" en la página 277
- "Archivo DAD: colección XML - correlación SQL" en la página 278 muestra un archivo DAD para una colección XML que hace uso de la correlación SQL.
- "Archivo DAD: XML - correlación de nodo_RDB" en la página 279 muestra un archivo DAD para una colección XML que hace uso de la correlación de nodo_RDB.

Archivo DAD: columna XML

Este archivo DAD contiene la correlación de una columna XML; esta correlación define la tabla, las tablas auxiliares y las columnas donde se almacenarán los datos XML.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>

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

   </Xcolumn>

</DAD>
```

Figura 16. Archivo DAD de ejemplo para una columna XML

Archivo DAD: colección XML - correlación SQL

Este archivo DAD contiene una sentencia SQL, que especifica las tablas DB2, columnas y condiciones donde se almacenarán los datos XML.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO</validation>
<Xcollection>
<SQL_stmt>SELECT o.order_key, customer_name, customer_email, p.part_key, color,
   quantity, price, tax, ship_id, date, mode from order_tab o, part_tab p,
   table (select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab) s
         WHERE o.order_key = 1 and
               p.price > 20000 and
               p.order_key = o.order_key and
               s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id</SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

Figura 17. Archivo DAD de ejemplo para una colección XML que utiliza la correlación SQL (Pieza 1 de 2)

```
<root_node>
<element_node name="Order">
  <attribute_node name="key">
    <column name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node><column name="customer_name"/></text_node>
    </element_node>
    <element_node name="Email">
      <text_node><column name="customer_email"/></text_node>
    </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
      <column name="color"/>
    </attribute_node>
    <element_node name="key">
      <text_node><column name="part_key"/></text_node>
    </element_node>
    <element_node name="Quantity">
      <text_node><column name="quantity"/></text_node>
    </element_node>
    <element_node name="ExtendedPrice">
      <text_node><column name="price"/></text_node>
    </element_node>
    <element_node name="Tax">
      <text_node><column name="tax"/></text_node>
    </element_node>
    <element_node name="Shipment" multi_occurence="YES">
      <element_node name="ShipDate">
        <text_node><column name="date"/></text_node>
      </element_node>
      <element_node name="ShipMode">
        <text_node><column name="mode"/></text_node>
      </element_node>
    </element_node>
  </element_node>
</element_node>
</root_node>
</Xcollection>
</DAD>
```

Figura 17. Archivo DAD de ejemplo para una colección XML que utiliza la correlación SQL (Pieza 2 de 2)

Archivo DAD: XML - correlación de nodo_RDB

Este archivo DAD utiliza elementos <RDB_node> para definir las tablas DB2, columnas y condiciones donde se almacenarán los datos XML.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>
<Xcollection>
     <prolog>?xml version="1.0"?</prolog>
     <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
     <root_node>
         <element_node name="Order">
           <RDB_node>
              <table name="order_tab"/>
              <table name="part_tab"/>
              <table name="ship_tab"/>
              <condition>
                order_tab.order_key = part_tab.order_key AND
                part_tab.part_key = ship_tab.part_key
              </condition>
           </RDB_node>
           <attribute_node name="key">
              <RDB_node>
                 <table name="order_tab"/>
                 <column name="order_key"/>
              </RDB_node>
           </attribute_node>
           <element_node name="Customer">
              <text_node>
                 <RDB_node>
                    <table name="order_tab"/>
                    <column name="customer"/>
                 </RDB_node>
              </text_node>
           </element_node>
           <element_node name="Part">
              <RDB_node>
                 <table name="part_tab"/>
                 <table name="ship_tab"/>
                 <condition>
                    part_tab.part_key = ship_tab.part_key
                 </condition>
              </RDB_node>
              <attribute_node name="key">
                 <RDB_node>
                    <table name="part_tab"/>
                    <column name="part_key"/>
                 </RDB_node>
              </attribute_node>
```

Figura 18. Archivo DAD de ejemplo para una colección XML que utiliza la correlación de nodo_RDB (Pieza 1 de 3)

```
<element_node name="Quantity">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
         <column name="quantity"/>
      </RDB_node>
   </text_node>
</element_node>
<element_node name="ExtendedPrice">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
            <column name="price"/>
            <condition>
               price > 2500.00
            </condition>
      </RDB_node>
   </text_node>
</element_node>
<element_node name="Tax">
   <text_node>
      <RDB_node>
         <table name="part_tab"/>
         <column name="tax"/>
      </RDB_node>
   </text_node>
</element_node>
```

Figura 18. Archivo DAD de ejemplo para una colección XML que utiliza la correlación de nodo_RDB (Pieza 2 de 3)

```
            <element_node name="shipment">
               <RDB_node>
                  <table name="ship_tab"/>
                  <condition>
                     part_key = part_tab.part_key
                  </condition>
               </RDB_node>
               <element_node name="ShipDate">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="date"/>
                        <condition>
                           date > "1966-01-01"
                        </condition>
                     </RDB_node>
                  </text_node>
               </element_node>
               <element_node name="ShipMode">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="mode"/>
                     </RDB_node>
                  </text_node>
               </element_node>
               <element_node name="Comment">
                  <text_node>
                     <RDB_node>
                        <table name="ship_tab"/>
                        <column name="comment"/>
                     </RDB_node>
                  </text_node>
               </element_node>
             </element_node> <! -- end of element Shipment>
           </element_node> <! -- end of element Part --->
         </element_node> <! -- end of element Order --->
       </root_node>
  </Xcollection>
</DAD>
```

Figura 18. Archivo DAD de ejemplo para una colección XML que utiliza la correlación de nodo_RDB (Pieza 3 de 3)

Apéndice C. Consideraciones sobre la página de códigos

Es importante asegurarse de que los documentos XML y otros archivos relacionados estén codificados correctamente para cada cliente o servidor que accede a los archivos. Por consiguiente, es importante comprender las premisas del XML Extender cuando se procesa un archivo y saber cómo maneja las conversiones de la página de códigos. Las consideraciones principales son las siguientes:

- Asegurarse de que la página de códigos real del cliente que recupera un documento de XML desde DB2 coincide con la codificación del documento.
- Asegurarse de que cuando un analizador XML procesa el documento, la declaración de codificación del documento XML también sea coherente con la codificación.

En los siguientes apartados se describen las cuestiones relacionadas con estas consideraciones: cómo se puede preparar ante posibles problemas y cómo XML Extender y DB2 dan soporte a las páginas de códigos cuando los documentos pasan del cliente al servidor y a la base de datos.

Terminología

En este apartado se emplean los siguientes términos:

codificación del documento
Página de códigos real de un documento XML.

declaración de codificación del documento
Nombre de la página de códigos especificada en la declaración XML. Por ejemplo:

```
<?xml version="1.0" encoding="ibm037"?>
```

documento coherente
Documento en el que la codificación del documento coincide con la página de códigos de declaración de codificación del documento.

documento incoherente
Documento en el que la codificación del documento no coincide con la página de códigos de declaración de codificación del documento.

variable (de entorno) de registro DB2CODEPAGE
Especifica la página de códigos de los datos que se presentan a DB2 desde una aplicación cliente de la base de datos. DB2 obtiene la página de códigos del cliente del entorno local del sistema operativo

del cliente, a menos que esta variable esté definida. En DB2, este valor prevalece sobre el entorno local del sistema operativo del cliente, si está definido.

página de códigos del cliente
Página de códigos de la aplicación. Si la variable DB2CODEPAGE está definida, la página de códigos del cliente es el valor de DB2CODEPAGE. De lo contrario, la página de códigos del cliente es el entorno local del sistema operativo del cliente.

página de códigos del servidor o página de códigos del entorno local del sistema operativo del servidor
Entorno local del sistema operativo en el que está instalada la base de datos DB2.

página de códigos de base de datos:
Codificación de los datos almacenados, que se determinan durante el tiempo de creación de la base de datos. En caso de que no se haya especificado explícitamente con la cláusula USING CODESET, este valor adopta el valor por omisión del entorno local del sistema operativo del servidor.

Premisas sobre la página de códigos de DB2 y XML

Cuando DB2 recibe o envía un documento XML, no comprueba la declaración de codificación. En lugar de ello, comprueba la página de códigos del cliente para ver si coincide con la página de códigos de la base de datos. Si son diferentes, DB2 convierte los datos del documento XML para que coincidan con la página de códigos de:

- La base de datos, cuando se importa el documento o un fragmento del documento, a una tabla de bases de datos
- La base de datos, cuando se descompone un documento en una o más tablas de bases de datos
- El cliente, cuando se exporta el documento de la base de datos y se presenta el documento al cliente
- El servidor, cuando se procesa un archivo con una UDF que devuelve datos de un archivo en el sistema de archivos del servidor

Cuando se importa un documento XML a la base de datos, generalmente se importa como documento XML que se almacena en una columna XML o para la descomposición para una colección XML, donde el contenido del elemento y del atributo se guardarán como datos de DB2. Cuando se importa un documento, DB2 convierte la codificación del documento en la de la base de datos. DB2 supone que el documento está en la página de códigos especificada en la columna “Página de códigos fuente” de la siguiente tabla. La Tabla 55 en la página 285 resume las conversiones que DB2 efectúa al

importar un documento XML.

Tabla 55. Utilización de las UDF y los procedimientos almacenados cuando el archivo XML se importa a la base de datos.

Método de proceso	Página de códigos fuente para la conversión	Página de códigos de destino para la conversión	Comentarios
Insertar el archivo DTD en la tabla DTD_REF	Página de códigos del cliente	Página de códigos de base de datos	
Habilitar la columna o habilitar el procedimiento almacenado de la colección, o bien los mandatos de administración que importan el archivo DAD	Página de códigos del cliente	Página de códigos de base de datos	
UDF: • XMLVarcharFromFile() • XMLCLOBFromFile() • Content(): recuperar de XMLFILE y almacenar en CLOB	Página de códigos del servidor	Página de códigos de base de datos	
Procedimientos almacenados de descomposición	Página de códigos del cliente	Página de códigos de base de datos	Se supone que el documento que se va a descomponer está en la página de códigos del cliente. Los datos procedentes de la descomposición se almacenan en tablas en la página de códigos de base de datos.

Cuando se exporta un documento XML desde la base de datos, generalmente se exporta en base a una petición del cliente para presentar un documento; una consulta del contenido de un documento XML o durante la composición de un documento a partir de los datos de DB2. Cuando se exporta un documento, DB2 convierte la codificación del documento en la del cliente o el servidor, en función de dónde se originó la petición y dónde se van a presentar los datos. La Tabla 56 en la página 286 resume las conversiones que DB2 realiza al exportar un documento XML.

Tabla 56. Utilización de las UDF y los procedimientos almacenados cuando el archivo XML se exporta desde la base de datos

Método de proceso	Conversión	Comentarios
UDF • XMLFileFromVarchar() • XMLFileFromCLOB()	Página de códigos de base de datos a página de códigos del cliente cuando los datos se presentan al cliente	
UDF • Content(): recuperar de XMLVARCHAR y almacenar en archivo de servidor externo	Página de códigos de base de datos a página de códigos del servidor	
Procedimiento almacenado de la composición: tablas resultantes almacenadas en la tabla resultante, que se puede consultar y exportar.	Página de códigos de base de datos a página de códigos del cliente cuando los resultados se presentan al cliente	Cuando se componen los documentos, XML Extender copia la declaración de codificación especificada en el código de la DAD, en el documento que se acaba de crear. Debe coincidir con la página de códigos del cliente en el momento de su presentación.

Consideraciones sobre la declaración de codificación

La declaración de codificación declara la codificación del documento XML y aparece en la sentencia de declaración XML. Cuando se utiliza XML Extender, es importante asegurarse de que la codificación del documento coincide con la página de códigos del cliente o el servidor, en función del lugar donde esté situado el archivo.

Declaraciones de codificación permitidas

Puede utilizar cualquier declaración de codificación en documentos XML, conforme a algunas directrices. En este apartado se definen estas directrices junto con las declaraciones de codificación soportadas.

Las codificaciones portátiles recomendadas para los datos XML son UTF-8 y UTF-16, de acuerdo con la especificación de XML. Si utiliza estas codificaciones, la aplicación será interoperable entre empresas. Si utiliza las codificaciones que figuran en la Tabla 57 en la página 287, es probable que la aplicación sea portátil entre sistemas operativos de IBM. Si utiliza otras codificaciones, es menos probable que los datos sean portátiles.

Para todos los sistemas operativos, se da soporte a las siguientes declaraciones de codificación. La lista siguiente describe el significado de cada columna:

- **Codificación** especifica la serie de codificación que se va a utilizar en la declaración de XML.
- **SO** muestra el sistema operativo en el que DB2 da soporte a la página de códigos indicada.
- **Página de códigos** muestra la página de códigos definida por IBM asociada con la codificación indicada

Tabla 57. Declaraciones de codificación soportadas por XML Extender

Categoría	Codificación	SO	Página de códigos
Unicode	UTF-8	AIX, SUN, Linux	1208
	UTF-16	AIX, SUN, Linux	1200
ASCII	iso-8859-1	AIX, Linux, Sun	819
	ibm-1252	Windows NT	1252
	iso-8859-2	AIX, Linux, Sun	912
	iso-8859-5	AIX, Linux	915
	iso-8859-6	AIX	1089
	iso-8859-7	AIX, Linux	813
	iso-8859-8	AIX, Linux	916
	iso-8859-9	AIX, Linux	920
MBCS	gb2312	Windows NT	1386
	ibm-932, shift_jis78	AIX	932
	Shift_JIS	Sólo AIX 4.3, Windows NT	943
	IBM-eucCN	AIX, Sun	1383
	IBM-eucJP, EUC-JP	AIX, Linux, Sun	954, 33722
	euc-tw, IBM-eucTW	AIX, Sun	964
	euc-kr, IBM-eucKR	AIX	970
	big5	AIX, Sun, Windows NT	950

La serie de codificación debe ser compatible con la página de códigos de destino del documento. Si se devuelve un documento desde un servidor a un cliente, la serie de codificación debe ser compatible con la página de códigos del cliente. Consulte el apartado “Declaraciones de codificación y codificaciones coherentes” en la página 288 para conocer las consecuencias de

codificaciones incompatibles. Visite la siguiente dirección URL si desea obtener una lista de las páginas de códigos soportadas por el analizador XML utilizado por el XML Extender:

`http://www.ibm.com/software/data/db2/extenders/xmlext/moreinfo/encoding.html`

Declaraciones de codificación y codificaciones coherentes

Cuando se procesa o se intercambia un documento XML con otro sistema, es importante que la declaración de codificación se corresponda con la codificación real del documento. Es importante asegurarse de que la codificación de un documento sea coherente con la del cliente porque las herramientas de XML, como los analizadores, generan un error para una entidad que incluya una declaración de codificación que no sea la que se indica en la declaración.

La Figura 19 muestra que los clientes tienen páginas de códigos coherentes con la declaración del documento y la codificación declarada.

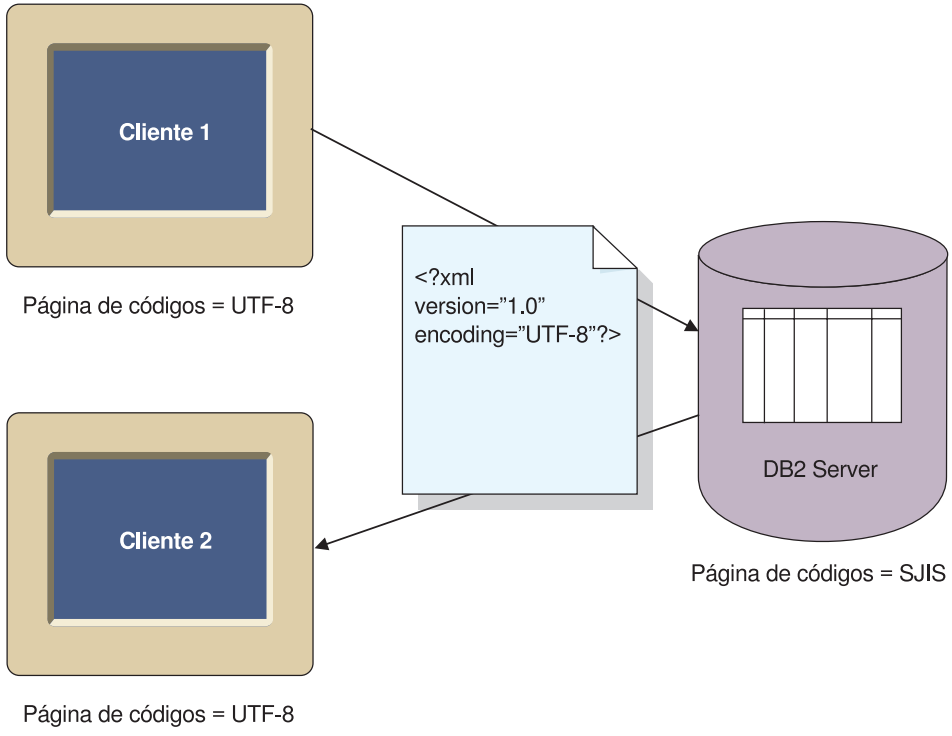


Figura 19. Clientes con páginas de códigos que coinciden

Las consecuencias de tener páginas de códigos diferentes son las siguientes situaciones posibles:

- Una conversión en la que se pueden perder datos, sobre todo si la página de códigos fuente es Unicode y la página de códigos de destino no es Unicode. Unicode contiene el juego completo de conversiones de caracteres. Si un archivo se convierte de UTF-9 a una página de códigos que no da soporte a todos los caracteres que se emplean en el documento, se podrían perder datos durante la conversión.
- Puede que la codificación declarada del documento XML ya no sea coherente con la codificación del documento real, si el documento lo recupera un cliente con una página de códigos distinta de la codificación declarada del documento.

La Figura 20 muestra un entorno en el que las páginas de códigos de los clientes no son coherentes.

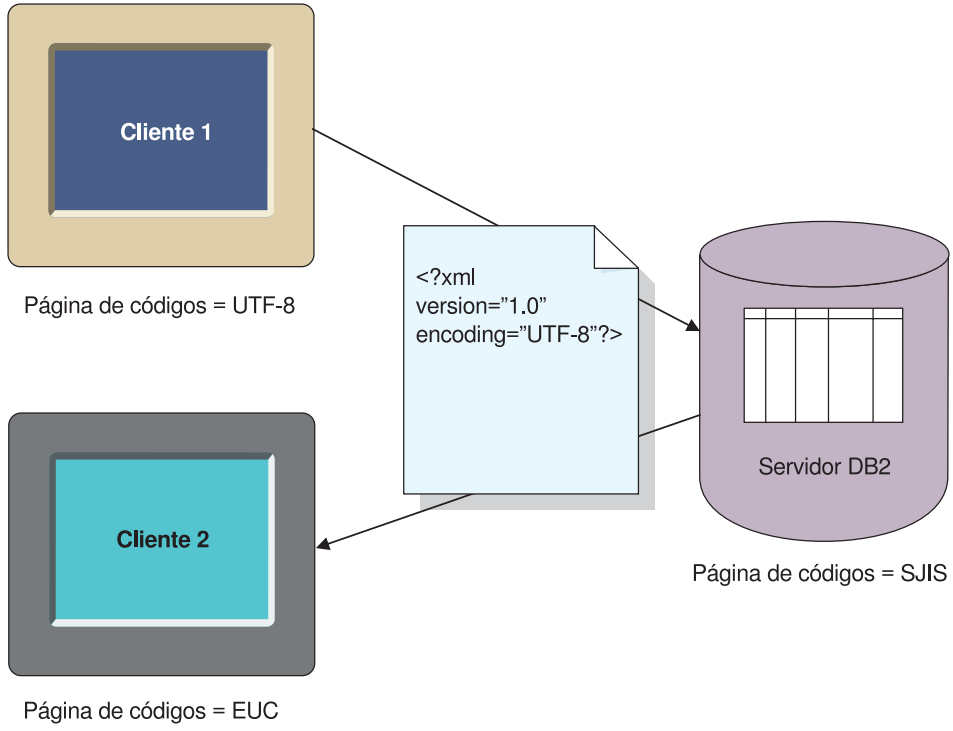


Figura 20. Los clientes tienen páginas de códigos no coincidentes

El cliente 2 recibe el documento en EUC, pero el documento tendrá una declaración de codificación de UTF-8.

Declaración de una codificación

El valor por omisión de la declaración de codificación es UTF-8 y la ausencia de una declaración de codificación significa que el documento está en UTF-8.

Para declarar un valor de codificación:

En la declaración de documento XML, especifique la declaración de codificación con el nombre de la página de códigos del cliente. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Escenarios sobre la conversión

XML Extender procesa documentos XML cuando:

- Almacena y recupera datos de columnas XML, utilizando el almacenamiento de columnas XML y el método de acceso
- Compone y descompone documentos XML

Los documentos se someten a la conversión de la página de códigos cuando pasan de un cliente o un servidor a una base de datos. Es posible que se produzcan incoherencias o que se dañen documentos XML durante las conversiones de páginas de códigos del cliente, servidor y base de datos. A la hora de decidir la declaración de codificación del documento, así como de planificar qué clientes y servidores pueden importar o exportar documentos desde la base de datos, tenga en cuenta las conversiones que se describen en las tablas anteriores y los escenarios que se describen a continuación.

Los siguientes escenarios describen escenarios de conversiones comunes que se pueden producir:

Escenario 1: Este escenario es una configuración con codificaciones coherentes, ninguna conversión de DB2 y un documento importado desde el servidor. La declaración de codificación del documento es UTF-8, el servidor es UTF-8 y la base de datos es UTF-8.

1. El documento se importa a DB2 utilizando la UDF XMLClobfromFile.
2. El documento se extrae en el servidor.
3. DB2 no necesita convertir el documento porque la página de códigos del servidor y la página de códigos de base de datos son idénticas. La codificación y la declaración son coherentes.

Escenario 2: Este escenario es una configuración con codificaciones coherentes, conversión de DB2 y un documento importado desde el servidor y exportado al cliente. La codificación del documento y la declaración es SJIS, la página de códigos del cliente es SJIS y las páginas de códigos del servidor y la base de datos son UTF-8.

1. El documento se importa a DB2 utilizando la UDF XMLClobfromfile del servidor.
2. DB2 convierte el documento desde SJIS y lo almacena en UTF-8. La declaración y la codificación no son coherentes en la base de datos.

3. Un cliente que utiliza SJIS solicita la presentación del documento en el navegador de Web.
4. DB2 convierte el documento a SJIS, la página de códigos del cliente. Ahora la codificación y declaración del documento son coherentes en el cliente.

Escenario 3: Este escenario es una configuración con codificaciones incoherentes, conversión de DB2 y un documento importado desde el servidor y exportado a un cliente. La declaración de codificación del documento es SJIS para el documento entrante. La página de códigos del servidor SJIS y el cliente y la base de datos son UTF-8.

1. El documento se importa a la base de datos utilizando una UDF de almacenamiento.
2. DB2 convierte el documento a UTF-8 desde SJIS. La codificación y la declaración son coherentes.
3. Un cliente con una página de códigos UTF-8 solicita la presentación del documento en un navegador de Web.
4. DB2 no realiza la conversión porque las páginas de códigos del cliente y de la base de datos son las mismas.
5. La codificación y la declaración del documento no son coherentes porque la declaración es SJIS y la codificación es UTF-8. Un analizador u otras herramientas de proceso de XML no pueden procesar el documento.

Escenario 4: Este escenario es una configuración con pérdida de datos, conversión de DB2 y un documento importado desde un servidor UTF-8. La declaración de codificación del documento es UTF-8, el servidor es UTF-8 y la base de datos es SJIS.

1. El documento se importa a DB2 utilizando la UDF XMLClobfromFile.
2. DB2 convierte la codificación a SJIS. Cuando se importa el documento, el documento almacenado en la base de datos podría estar dañado porque puede que los caracteres representados en UTF-8 no tengan una representación en SJIS.

Escenario 5:

Este escenario es una configuración con una limitación de Windows NT. En Windows NT, los entornos locales del sistema operativo no se pueden establecer en el valor UTF-8, sin embargo, DB2 permite que el cliente establezca la página de códigos en el valor UTF-8 utilizando `db2set DB2CODEPAGE=1208`. En este escenario, el cliente y el servidor están en la misma máquina. El cliente es UTF-8, pero el servidor no se puede establecer en el valor UTF-8; la página de códigos es 1252. El documento se codifica como 1252 y la declaración de codificación es ibm-1252. La página de códigos de base de datos es UTF-8.

1. El documento se importa desde el servidor mediante una UDF de almacenamiento y se convierte de 1252 a 1208.
2. El documento se exporta desde DB2 utilizando la UDF Content() del cliente.
3. DB2 convierte el documento de UTF-8 a 1252, aunque el cliente podría obtener 1208 porque el cliente está en el mismo sistema que el servidor y tiene el valor 1208.

Acciones para evitar documentos XML no coherentes

En los apartados anteriores, hemos explicado cómo un documento XML puede tener una codificación no coherente, es decir, la declaración de codificación está en conflicto con la codificación del documento. Las codificaciones no coherentes pueden causar la pérdida de los datos y/o de los documentos XML que no se utilizan.

Siga una de estas recomendaciones para asegurarse de que la codificación del documento XML es coherente con la página de códigos del cliente, antes de entregar el documento a un procesador XML, como por ejemplo, un analizador:

- Cuando exporte un documento desde la base de datos utilizando las UDF de XML Extender, intente una de las técnicas siguientes: (premisa: XML Extender ha exportado el archivo, en la página de códigos del servidor al sistema de archivos del servidor).
  - Convierta el documento a la página de códigos de la codificación declarada
  - Altere temporalmente la codificación declarada, si la herramienta tiene un recurso para alterar temporalmente
  - Cambie manualmente la declaración de codificación del documento exportado a la codificación real del documento (es decir, la página de códigos del servidor)
- Cuando exporte un documento desde la base de datos utilizando los procedimientos almacenados de XML Extender, intente una de las siguientes técnicas: (se supone que el cliente está consultando la tabla resultante, en la que está almacenado el documento compuesto)
  - Convierta el documento a la página de códigos de la codificación declarada
  - Altere temporalmente la codificación declarada, si la herramienta tiene un recurso para alterar temporalmente
  - Antes de consultar la tabla resultante, solicite al cliente que defina la variable de entorno DB2CODEPAGE para que la página de códigos sea una página de códigos compatible con la declaración de codificación del documento XML.

- Cambie manualmente la declaración de codificación del documento exportado a la codificación real del documento (es decir, la página de códigos del cliente)

- **Limitación cuando se utiliza Unicode y Windows NT:** En Windows NT, el entorno local del sistema operativo puede tener el valor UTF-8. Siga estas directrices cuando importe o exporte documentos:
  - Cuando importe archivos las DTD codificadas en UTF-8, establezca la página de códigos del cliente en el valor UTF-8, utilizando:

    ```
    db2set DB2CODEPAGE=1208
    ```

    Utilice esta técnica cuando:
    - Inserte una DTD en la tabla db2xml.DTD_REF
    - Habilite una columna o una colección
    - Descomponga procedimientos almacenados
  - Cuando se utilizan las UDF Content() o XMLxxxfromFile para importar documentos XML, los documentos deben codificarse en la página de códigos del entorno local del sistema operativo del servidor, que no puede ser UTF-8.
  - Cuando exporte un archivo XML desde la base de datos, defina la página de códigos del cliente con el siguiente mandato para que DB2 codifique los datos resultantes en UTF-8:

    ```
    db2set DB2CODEPAGE=1208
    ```

    Utilice esta técnica cuando:
    - Consulte la tabla resultante después de la composición
    - Extraiga datos de una columna XML utilizando las UDF de extracción
  - Cuando se utilizan las UDF Content() o XMLxxxfromFile para exportar documentos XML en archivos del sistema de archivos del servidor, los documentos resultantes se codifican en la página de códigos del entorno local del sistema operativo del servidor, que no puede ser UTF-8.

Apéndice D. Límites del XML Extender

El archivo DAD, los procedimientos almacenados y las tablas de XML Extender tienen los siguientes límites:

Tabla 58. Límites de XML Extender

Valor	Límite
Tabla de una colección de XML de descomposición	1024 filas de cada documento XML descompuesto
Parámetros del procedimiento almacenado:	
CLOB de documento XML	1 MB[2]
CLOB de DAD (Definición de acceso a documento)	100KB[2]
nombreColección	30
nombreCol	30
nombreBd	18
vistaPredefinida	128
ID_raíz	128
nombreTabResul	128
espaciotablas	128
nombreTb	128[1]
Columnas de la tabla db2xml.DTD_REF	
AUTHOR	128
CREATOR	128
UPDATOR	128
DTDID	128
CLOB	100 KB

Notas:

1. Si *nombreTb* está calificada con un nombre de esquema, el nombre completo (incluido el carácter separador) no debe tener una longitud de más de 128 caracteres.
2. Este tamaño se puede modificar. Si desea saber cómo hacerlo, consulte el apartado "Aumento del límite del CLOB" en la página 214.

Los nombres pueden ser objeto de una expansión cuando DB2 los convierte desde la página de códigos del cliente a la página de códigos de la base de datos. Un nombre puede ajustarse al límite de tamaño del cliente, pero puede exceder el límite cuando el procedimiento almacenado obtiene el nombre convertido. Para más información, consulte el apartado "National Language Support Application Development" del capítulo "Programming in Complex Environments" de la publicación *DB2 Application Development Guide*.

Avisos

El presente manual está pensado para productos y servicios ofrecidos en los Estados Unidos. Es posible que IBM no comercialice en otros países los productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre productos y servicios disponibles actualmente en su zona geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar ese producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad intelectual de IBM. Sin embargo, corresponde al usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patente en tramitación que cubran temas descritos en este documento. La posesión de este documento no otorga ninguna licencia sobre dichas patentes. Puede solicitar información sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
E.E.U.U.

Para las consultas sobre licencias referentes a información de doble byte (DBCS), póngase en contacto con el Departamento de la Propiedad Intelectual de IBM de su país, o envíe sus peticiones de información, por escrito, a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país en el que tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías

expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y/o cambios en los productos y/o programas descritos en esta publicación.

Los propietarios de licencias de este programa que deseen tener información sobre él con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido el presente) y (ii) la utilización común de la información intercambiada, deben ponerse en contacto con:

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
E.E.U.U.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos, el pago de una tarifa.

El programa bajo licencia descrito en esta publicación y su material bajo licencia asociado son proporcionados por IBM según los términos del Contrato del Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia IBM o cualquier otro contrato equivalente entre IBM y el cliente.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes de información de uso público. IBM no ha comprobado esos productos y no puede confirmar la veracidad del rendimiento, la compatibilidad ni ninguna otra declaración relacionada con productos que no son de IBM. Las preguntas referentes a las prestaciones de productos que no son de IBM se deben dirigir a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan metas y objetivos.

LICENCIA DE COPYRIGHT:

La presente publicación contiene programas de aplicación de ejemplo escritos en lenguaje fuente, que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo en cualquier forma, sin pago a IBM, con el fin de desarrollar,

utilizar, comercializar o distribuir programas de aplicación conformes a la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni suponer la fiabilidad, eficiencia ni función de estos programas.

Marcas registradas

Los términos siguientes son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países:

DB2	Net.Data
DB2 Extenders	OS/2
DB2 Universal Database	OS/390
IBM	OS/400
IMS	VTAM

Java y todas las marcas comerciales y logotipos basados en Java son marcas comerciales o marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

UNIX es una marca registrada en los Estados Unidos y/o en otros países y se ofrece bajo licencia exclusivamente a través de X/Open Company Limited.

Otros nombres de empresas, productos y servicios pueden ser marcas registradas o marcas de servicio de terceros.

Glosario

aparición múltiple. Indicación de si un elemento o atributo de columna se puede utilizar más de una vez en un documento. La aparición múltiple se especifica en el archivo DAD.

API. Véase *interfaz de programación de aplicaciones*.

archivo externo. Archivo que existe en un sistema de archivos externo a DB2.

atributo. Véase *atributo XML*.

atributo XML. Cualquier atributo especificado por la lista de atributos (ATTLIST) debajo del elemento XML en la DTD. El XML Extender utiliza la vía de ubicación para identificar un atributo.

búsqueda completa de texto. Mediante el uso del Text Extender de DB2, búsqueda de cadenas de texto en cualquier punto de un documento sin tener en cuenta la estructura del documento.

búsqueda por secciones. Permite realizar una búsqueda de texto dentro de una sección que puede estar definida por la aplicación. Para dar soporte a la búsqueda estructural de texto, se puede definir una sección mediante la vía de ubicación abreviada de Xpath.

clave foránea. Clave que forma parte de la definición de una restricción de referencia y que consta de una o más columnas de una tabla dependiente.

clave primaria. Clave exclusiva que forma parte de la definición de una tabla. La clave primaria es la clave padre por omisión de una definición de restricción de referencia.

CLOB. Character large object (gran objeto de caracteres).

código XML. Cualquier código de marcación válido de XML, principalmente el elemento XML. Los términos ″código″ y ″elemento″ se utilizan de forma indistinta.

colección XML. Colección de tablas relacionales que contiene los datos para componer documentos XML o resultantes de la descomposición de documentos XML.

columna XML. Columna de la tabla de aplicación que ha sido habilitada para los UDT del XML Extender.

componer. Generar documentos XML a partir de datos relacionales de una colección XML.

condición. Especificación de los criterios para seleccionar datos XML o de la forma de unir las tablas de la colección XML.

conjunto resultante. Conjunta de filas devueltas por un procedimiento almacenado.

consulta. Petición de información a una base de datos que se realiza de acuerdo con unas condiciones determinadas; por ejemplo, una consulta puede solicitar una lista de todos los clientes especificados en una tabla cuyo saldo sea mayor que 1000.

correlación de nodo_RDB. Ubicación del contenido de un elemento XML o del valor de un atributo XML, que están definidos por el nodo_RDB. El XML Extender utiliza esta correlación para determinar dónde almacenar o recuperar los datos XML.

correlación SQL. Definición de la relación existente entre el contenido de un elemento XML o valor de un atributo XML y datos relacionales, mediante la utilización de una o más sentencias de SQL y el modelo de datos XSLT. El XML Extender utiliza esta definición para determinar dónde almacenar o recuperar los datos XML. La

correlación SQL se define mediante el elemento SQL_stmt en el archivo DAD.

DAD (document access definition). Véase *Definición de acceso a documento*.

datalink (enlace de datos). Tipo de datos de DB2 que permite establecer referencias lógicas entre la base de datos y un archivo que está almacenado fuera de ella.

datos de columna. Datos que están almacenados en una columna DB2. El tipo de los datos puede ser cualquiera de los soportados por DB2.

DBCLOB. Double-byte character large object (gran objeto de caracteres de doble byte).

Definición de acceso a documento (document access definition, DAD). Sirve para definir el esquema de indexación de una columna XML o el esquema de correlación de una colección XML. Se puede utilizar para habilitar una columna XML de una colección XML, la cual tiene formato de XML.

Definición de tipo de documento (Document type definition, DTD). Conjunto de declaraciones para elementos y atributos XML. La DTD define qué elementos se utilizan en el documento XML, en qué orden se pueden utilizar y qué elementos pueden contener otros elementos. Puede asociar una DTD a un archivo DAD (archivo de definición de acceso a documento) para validar documentos XML.

depósito DTD. Tabla DB2, llamada DTD_REF, en la que cada fila representa una DTD, junto con información adicional de metadatos.

descomponer. Separar documentos XML para obtener el conjunto de tablas relacionales de una colección XML.

documento bien formado. Documento XML que no contiene una DTD. Aunque siga la especificación XML, un documento con una DTD válida debe también estar bien formado.

documento válido. Documento XML que tiene una DTD asociada. Para ser válido, el documento XML no debe violar las reglas sintácticas especificadas en su DTD.

DTD (document type definition). Véase *Definición de tipo de documento*.

EDI. Electronic Data Interchange (intercambio electrónico de datos).

elemento. Véase *elemento XML*.

elemento raíz. Elemento de nivel superior de un documento XML.

elemento XML. Código o elemento de XML tal como aparece especificado en la DTD de XML. El XML Extender utiliza la vía de ubicación para identificar un elemento.

espacio de tablas. Concepto abstracto consistente en una colección de contenedores donde se almacenan objetos de base de datos. El espacio de tablas proporciona un nivel de indirección entre una base de datos y las tablas contenidas en la base de datos. Un espacio de tablas:

- Tiene espacio en dispositivos de almacenamiento magnético asignados a él.
- Tiene tablas creadas dentro de él. Estas tablas ocuparán espacio en los contenedores pertenecientes al espacio de tablas. Las porciones de una tabla correspondientes a los datos, el índice, los campos largos y los LOB pueden estar almacenados en el mismo espacio de tablas, o se pueden desglosar individualmente en espacio de tablas separados.

esquema. Colección de objetos de base de datos, tales como tablas, vistas, índices o desencadenantes. Proporciona una clasificación lógica de objetos de base de datos.

esquema de correlación. Define cómo se representan los datos XML en una base de datos relacional. El esquema de correlación se especifica en el archivo DAD. El XML Extender proporciona dos tipos de esquemas de correlación: la *correlación SQL* y la *correlación de base de datos relacional (correlación de nodo_RDB)*.

examinador. Véase *navegador Web*.

expresión de vía. Véase *vía de ubicación*.

Extensible Stylesheet language (XSL) (lenguaje de Hoja de Estilo Ampliable). Lenguaje utilizado para representar hojas de estilo. XSL consta de dos partes: un lenguaje para transformar documentos XML, y un vocabulario XML para especificar semánticas de formato.

Extensive Stylesheet Language Transformation (XSLT). Lenguaje utilizado para transformar documentos XML en otros documentos XML. XSLT está diseñado para ser utilizado como parte de XSL, que es un lenguaje de hoja de estilo de XML.

fuente de datos. Gestor de datos, local o remoto, relacional o no relacional, que permite el acceso a los datos a través de un controlador ODBC que da soporte a las API ODBC.

función de conversión de datos. Función que se utiliza para convertir un tipos de datos (origen) en un tipo de datos (destino) diferente. En general, la función de conversión adopta el nombre del tipo de datos destino. Esta función tiene un solo argumento, cuyo tipo es el tipo de datos origen; el tipo devuelto es el tipo de datos destino.

función definida por el usuario (user-defined function, UDF). Función que se define para el sistema de gestión de bases de datos y que luego se puede hacer referencia a ella en consultas SQL. Puede ser una de las funciones siguientes:

- Una función externa, cuyo cuerpo está escrito en un lenguaje de programación que utiliza valores escalares como argumentos, y se obtiene un resultado escalar en cada invocación.
- Una función derivada, implantada por otra función interna o definida por el usuario que ya es reconocida por el DBMS. Esta función puede ser una función escalar o una función de columna (función de agregación), y devuelve un valor individual a partir de un conjunto de valores (por ejemplo, MAX o AVG).

función escalar. Operación de SQL que obtiene un valor individual a partir de otro valor y que se expresa mediante un nombre de función seguido de una lista de argumentos entre paréntesis.

función predefinida de conversión de datos. Función que convierte el tipo de datos base del SQL en un UDT.

función sobrecargada. Nombre de función para el que existen varias instancias de función.

gran objeto de caracteres (character large object, CLOB). Cadena de caracteres de un solo byte cuya longitud puede ser de hasta 2 GB. Los CLOB tienen una página de códigos asociada. Los objetos de texto que contienen caracteres de un solo byte se almacenan en una base de datos DB2 en forma de CLOB.

gran objeto de caracteres de doble byte (double-byte character large object, DBCLOB). Cadena de caracteres de doble byte, o combinación de caracteres de un solo byte y de doble byte, cuya longitud puede ser de hasta 2 GB. Los DBCLOB tienen una página de códigos asociada. Los objetos de texto que contienen caracteres de doble byte se almacenan en una base de datos DB2 en forma de DBCLOB.

gran objeto (large object, LOB). Secuencia de bytes cuya longitud puede ser de hasta 2 GB. Un LOB puede ser de tres tipos: *gran objeto binario* (BLOB), *gran objeto de caracteres (CLOB)* y *gran objeto de caracteres de doble byte (DBCLOB)*.

ID raíz. Identificador exclusivo que asocia todas las tablas auxiliares con la tabla de aplicación.

indexación de árbol binario. Método de indexación nativo proporcionado por DB2. Crea entradas de índice en la estructura de árbol binario. Da soporte a los tipos de datos base de DB2.

indexar, índice. Conjunto de punteros que siguen un orden lógico de acuerdo con los valores de una clave. Los índices proporcionan un rápido acceso a los datos y pueden asegurar la unicidad de las filas de la tabla.

índice estructural de texto. Indexar cadenas de texto basándose en la estructura arborescente del documento XML, utilizando el Text Extender de DB2.

intercambio de datos. Compartimiento de datos entre aplicaciones. XML da soporte al intercambio de datos sin necesidad de transformar primero el formato exclusivo de los datos.

Intercambio electrónico de datos (Electronic Data Interchange, EDI). Estándar para el intercambio electrónico de datos para aplicaciones inter-empresariales.

interfaz de programación de aplicaciones (API).

(1) Interfaz funcional proporcionada por el sistema operativo o por un programa bajo licencia que se puede solicitar por separado. Una API permite que un programa de aplicación escrito en un lenguaje de alto nivel utilice determinados datos o funciones del sistema operativo o programas bajo licencia.

(2) En DB2, una función incluida dentro de la interfaz como, por ejemplo, la API de mensajes de errores.

(3) En DB2, una función incluida dentro de la interfaz. Por ejemplo, la API de mensajes de errores de obtención.

Java Database Connectivity (JDBC). Interfaz de programación de aplicaciones (API) que tiene las mismas características que Open Database Connectivity (ODBC), pero está diseñada específicamente para ser utilizada por aplicaciones Java de base de datos. Además, para las bases de datos que no tienen un controlador JDBC, JDBC incluye un puente JDBC-ODBC, que es un mecanismo para convertir JDBC a ODBC; JDBC muestra la API JDBC a las aplicaciones Java de base de datos y la convierte a ODBC. JDBC fue creado por Sun Microsystems, Inc. y diversos colaboradores y proveedores.

JDBC. Java Database Connectivity.

LOB. Large object (gran objeto).

localizador. Puntero que se puede utilizar para localizar un objeto. En DB2, el localizador de LOB (large object block) es el tipo de datos que sirve para localizar los LOB.

localizador uniforme de recursos (uniform resource locator, URL). Dirección donde se especifica un servidor HTTP y opcionalmente un nombre de directorio y de archivo; por ejemplo: http://www.ibm.com/data/db2/extenders.

método de acceso y almacenamiento. Asocia documentos XML a una base de datos DB2, utilizando dos métodos de acceso y almacenamiento principales: columnas XML y colecciones XML. Véase también *columna XML* y *colección XML*.

modelo de datos XPath. Estructura en árbol utilizada para modelar un documento XML y navegar por el mismo utilizando nodos.

navegador Web. Programa cliente que inicia peticiones dirigidas a un servidor Web y visualiza la información que el servidor devuelve.

nodo. En el particionamiento de bases de datos, sinónimo de partición de base de datos.

nodo_de_atributo (attribute_node). Representación de un atributo de un elemento.

nodo de base de datos relacional (nodo_RDB). Nodo que contiene una o más definiciones de elemento para tablas, columnas opcionales y condiciones opcionales. Las tablas y columnas se utilizan para definir cómo deben almacenarse los datos XML en la base de datos. La condición especifica los criterios para seleccionar datos XML o la forma de unir las tablas de la colección XML.

nodo_de_elemento (element_node). Representación de un elemento. Un nodo de elemento puede ser el elemento raíz o un elemento hijo.

nodo_de_elemento superior. Representación del elemento raíz del documento XML en el archivo DAD.

nodo_de_texto (text_node). Representa el texto CDATA de un elemento.

objeto. En la programación orientada a objetos, concepto abstracto que consta de datos y las operaciones asociadas a esos datos.

objeto XML. Equivalente a un documento XML.

ODBC. Open Database Connectivity.

Open Database Connectivity. Interfaz de programación de aplicaciones (API) estándar que se utiliza para acceder a datos en sistemas de gestión de bases de datos, relacionales y no relacionales. Mediante esta API, las aplicaciones de base de datos pueden acceder a los datos almacenados en sistemas de gestión de bases de datos, en diversos tipos de máquinas, aunque el sistema de gestión de bases de datos utilice un formato de almacenamiento de datos y una interfaz de programación diferentes. ODBC está basado en la especificación CLI (interfaz de nivel de llamada) de X/Open SQL Access Group y ha sido desarrollado por Digital Equipment Corporation (DEC), Lotus, Microsoft y Sybase. Compárese con *Java Database Connectivity*.

partición. División de almacenamiento, de tamaño fijo.

predicado. Elemento de una condición de búsqueda que expresa o implica una operación de comparación.

procedimiento. Véase *procedimiento almacenado*.

procedimiento almacenado. Bloque de estructuras de procedimiento y sentencias de SQL incorporado que se almacena en una base de datos y se puede invocar mediante un nombre. El procedimiento almacenado permite que un programa de aplicación se ejecute en dos partes. Una parte se ejecuta en el cliente y la otra parte se ejecuta en el servidor. Esto permite que una sola llamada produzca varios accesos a la base de datos.

RDB_node. Relational database node (nodo de base de datos relacional).

sistema de archivos local. Sistema de archivos que existe dentro de DB2.

SQL estático. Sentencias de SQL que están incorporadas dentro de un programa, y que se preparan durante el proceso de preparación del programa antes de ejecutar el programa. Una vez preparada, la sentencia de SQL estático no cambia, pero pueden cambiar los valores de las variables de lenguaje principal especificadas por la sentencia.

SQL incorporado. Sentencias de SQL codificadas dentro de un programa de aplicación. Véase *SQL estático*.

subconsulta. Sentencia SELECT completa que se utiliza dentro de una condición de búsqueda en una sentencia de SQL.

tabla auxiliar. Tablas adicionales que el XML Extender crea para mejorar el rendimiento cuando se buscan elementos o atributos en una columna XML.

tabla de metadatos. Véase *tabla de soporte de administración*.

Tabla de referencia de DTD (tabla DTD_REF). Tabla que contiene definiciones DTD, las cuales se utilizan para validar documentos XML y para ayudar a las aplicaciones a definir una DAD. Los usuarios pueden insertar sus propias DTD en la tabla DTD_REF. Esta tabla se crea al habilitar una base de datos para XML.

tabla de usuario. Tabla que se crea para ser utilizada por una aplicación.

tabla DTD_REF. Tabla de referencia de DTD.

tabla resultante. Tabla cuyas filas son el resultado de una consulta SQL o de la ejecución de un procedimiento almacenado.

tablas de soporte de administración. Tablas que un DB2 Extender utiliza para procesar peticiones de usuario referentes a objetos XML. Algunas tablas de soporte de administración definen tablas de usuario y columnas que están habilitadas para un Extender. Otras tablas de soporte de administración contienen información

sobre atributos referente a objetos contenidos en columnas habilitadas. Sinónimo de tabla de metadatos.

tabla XML. Tabla de aplicación que contiene una o más columnas de XML Extender.

tipo de datos. Atributo de columnas y literales.

tipo definido por el usuario (user-defined type, UDT). Tipo de datos que no es nativo del gestor de bases de datos y que es creado por un usuario. Véase *tipo diferenciado*.

tipo diferenciado. Véase *tipo definido por el usuario*.

UDF de XML. Función de DB2 definida por el usuario que proporciona el XML Extender.

UDF (user-defined function). Véase *función definida por el usuario*.

UDT de XML. Tipo DB2 definido por el usuario que proporciona el XML Extender.

UDT (user-defined type). Véase *tipo definido por el usuario*.

UNION. Operación de SQL que combina los resultados de dos sentencias SELECT. UNION se utiliza a menudo para fusionar listas de valores que proceden de varias tablas.

unión. Operación relacional que permite recuperar datos de dos o más tablas basándose en valores de columna coincidentes.

URL. Uniform resource locator (localizador uniforme de recursos).

validación. Proceso consistente en utilizar una DTD para asegurar la validez del documento XML y para permitir búsquedas estructuradas sobre datos XML. La DTD se almacena en el depósito de DTD.

vía de ubicación. Secuencia de códigos XML que identifican un elemento o atributo de XML. La vía de ubicación identifica la estructura del documento XML, indicando el contexto del elemento o atributo. Una vía con una barra inclinada (/) individual indica que el contexto es el documento completo. La vía de ubicación se utiliza en las UDF de extracción para identificar los elementos y atributos que se deben extraer. La vía de ubicación también se utiliza en el archivo DAD para especificar la correlación entre un elemento o atributo XML y una columna DB2, cuando se define el método de indexación para la columna XML. Además, la vía de ubicación es utilizada por el Text Extender para realizar búsquedas estructurales de texto.

vía de ubicación absoluta. Es la vía de acceso completa de un objeto. La vía de acceso absoluta comienza en el nivel más alto, el elemento "raíz", el cual está representado por una barra inclinada (/) o una barra inclinada invertida (\).

vía de ubicación simple. Secuencia de nombres de tipos de elementos que están conectados por una barra inclinada simple (/).

vista de unión. Vista DB2 creada por la sentencia "CREATE VIEW" que une una o más tablas entre sí.

vista predefinida. Representación de datos en la que se unen una tabla XML y todas sus tablas auxiliares asociadas.

XML. eXtensible Markup Language (lenguaje de marcación ampliable).

XML Path Language. Lenguaje para direccionar partes de un documento XML. XML Path Language está diseñado para ser utilizado por XSLT. Cada vía de ubicación se puede expresar utilizando la sintaxis definida para XPath.

XPath. Lenguaje para direccionar partes de un documento XML.

XSL. XML Stylesheet Language (lenguaje de hoja de estilo XML).

XSLT. XML Stylesheet Language Transformation (transformación de lenguaje de hoja de estilo XML).

Índice

D

E

F

N

O

P

R

S

T

U

V

W

X

Cómo ponerse en contacto con IBM

Si tiene un problema técnico, repase y lleve a cabo las acciones que se sugieren en la *Guía de resolución de problemas* antes de ponerse en contacto con el Centro de Asistencia al Cliente de DB2. Dicha guía sugiere información que puede reunir para ayudar al Centro de Asistencia a proporcionarle un mejor servicio.

Para obtener información o para solicitar cualquiera de los productos de DB2 Universal Database, consulte a un representante de IBM de una sucursal local o a un concesionario autorizado de IBM.

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-237-5511 para obtener soporte técnico
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles

Información sobre productos

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) o 1-800-3IBM-OS2 (1-800-342-6672) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

http://www.ibm.com/software/data/
Las páginas Web de DB2 ofrecen información actual sobre DB2 referente a novedades, descripciones de productos, planes de formación, etc.

http://www.ibm.com/software/data/db2/library/
La biblioteca técnica de servicio y de productos DB2 ofrece acceso a las preguntas más frecuentes (FAQ), arreglos de programa, manuales e información técnica actualizada sobre DB2.

Nota: Puede que esta información sólo esté disponible en inglés.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
El sitio Web para el pedido de publicaciones internacionales proporciona información sobre cómo hacer pedidos de manuales.

http://www.ibm.com/education/certify/
El Programa de homologación profesional contenido en el sitio Web de IBM proporciona información de prueba de homologación para diversos productos de IBM, incluido DB2.

ftp.software.ibm.com
Conéctese como anónimo (anonymous). En el directorio `/ps/products/db2` encontrará programas de demostración, arreglos de programa, información y herramientas referentes a DB2 y a muchos otros productos.

comp.databases.ibm-db2, bit.listserv.db2-l
En estos foros de discusión de Internet los usuarios pueden explicar sus experiencias con los productos DB2.

En Compuserve: GO IBMDB2
Entre este mandato para acceder a los foros referentes a la familia de productos DB2. Todos los productos DB2 tienen soporte a través de estos foros.

Para conocer cómo ponerse en contacto con IBM desde fuera de los Estados Unidos, consulte el Apéndice A del manual *IBM Software Support Handbook*. Para acceder a este documento, vaya a la página Web siguiente: http://www.ibm.com/support/ y luego seleccione el enlace "IBM Software Support Handbook", cerca del final de la página.

Nota: En algunos países, los distribuidores autorizados de IBM deben ponerse en contacto con su organización de soporte en lugar de acudir al Centro de Asistencia de IBM.