IBM® DB2® Universal Database™

XML Extender Administración y programación

Versión 8.2

SC10-3750-01

IBM® DB2® Universal Database™

XML Extender Administración y programación

Versión 8.2

SC10-3750-01

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

Este manual es la traducción del original inglés *IBM DB2 Universal Database XML Extender Administration and Programming Version 8.2*, (SC27-1234-01).

Puede realizar pedidos de publicaciones en línea o a través del representante de IBM de su localidad.

- Para realizar pedidos de publicaciones en línea, vaya a IBM Publications Center en www.ibm.com/shop/publications/order
- Para encontrar el representante de IBM correspondiente a su localidad, vaya a IBM Directory of Worldwide Contacts en www.ibm.com/planetwide

Para realizar pedidos de publicaciones en márketing y ventas de DB2 de los EE.UU. o de Canadá, llame al número 1-800-IBM-4YOU (426-4968).

Contenido

Acerca de este manual

Esta sección contiene la siguiente información:

A quién va dirigido este manual

Este manual está pensado para las siguientes personas:

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

  http://www.w3.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 familiarizados con los conceptos, herramientas y técnicas de administración de DB2.
- Programadores de aplicaciones DB2 familiarizados con SQL y con uno o más lenguajes de programación que se pueden utilizar con aplicaciones de DB2 UDB.

Cómo obtener una versión actual de este manual

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

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

Cómo utilizar este manual

Este manual esta estructurado de la siguiente manera:

Parte 1. Introducción
Esta parte proporciona una visión general de XML Extender y cómo utilizarlo en las aplicaciones comerciales. Contiene un caso 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 UDB para datos XML. Lea esta parte si necesita administrar una base de datos DB2 UDB que contenga datos XML.

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

Parte 4. Consulta
Esta parte describe cómo utilizar los mandatos de administración de XML Extender, los tipos definidos por el usuario, las funciones definidas por el usuario y los procedimientos almacenados. También lista los mensajes y códigos que XML Extender emite. Lea esta parte si está familiarizado con los conceptos y tareas de 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 la DTD para la definición de acceso a documentos, para los ejemplos y el caso de iniciación, así como otros productos XML de IBM®.

Convenios de resaltado

Este manual utiliza los convenios siguientes:

El texto en negrita indica:

- Mandatos
- Nombres de campos
- Nombres de menús
- Pulsadores

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

Las letras en mayúsculas indican:

- Tipos de datos
- Nombres de columnas
- Nombres de tablas

El texto de ejemplo indica:

- Mensajes del sistema
- Valores que escribe el usuario
- Ejemplos de codificación
- Nombres de directorios
- Nombres de archivos

Cómo leer los diagramas de sintaxis

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

Para leer los diagramas de sintaxis, siga estas directrices:

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

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

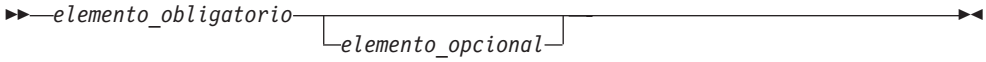
  El símbolo —► indica que la sintaxis de la sentencia continúa en la línea siguiente.

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

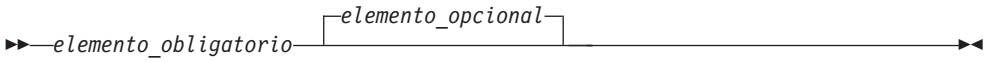
  El símbolo —►◄ indica el final de una sentencia.

  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).

- Los elementos opcionales aparecen debajo de la ruta principal.

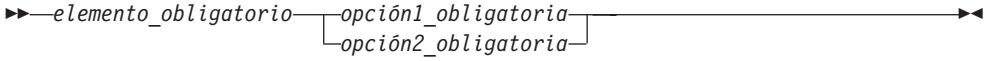


  Si un elemento opcional aparece encima de la ruta principal, ese elemento no afecta a la ejecución de la sentencia y se utiliza sólo por razones de legibilidad.

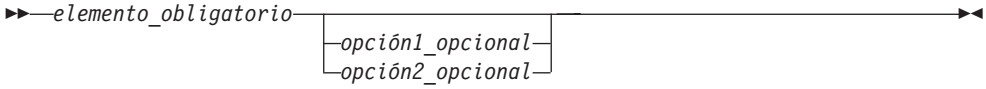


- Si puede elegir entre dos o más elementos, aparecen apilados verticalmente.

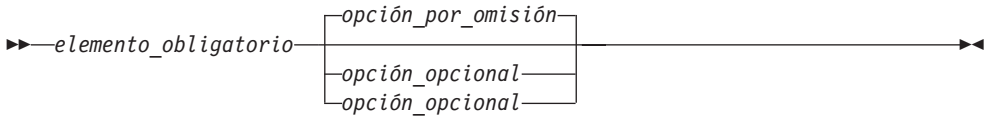
  Si *debe* seleccionar uno de los elementos, un elemento de la pila aparece en la ruta principal.



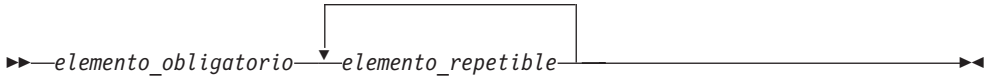
  Si es opcional elegir uno de los elementos, la pila completa aparece debajo de la ruta principal.



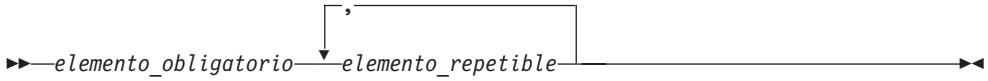
  Si uno de los elementos es la opción por omisión, aparece encima de la ruta principal y las demás opciones aparecen debajo.



- 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 XML Extender, las palabras clave se pueden utilizar indistintamente en mayúsculas y minúsculas. 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.

Parte 1. Introducción

Esta parte proporciona una visión general de XML Extender y cómo utilizarlo en las aplicaciones comerciales.

Capítulo 1. Introducción

Introducción a XML Extender

DB2® XML Extender proporciona la posibilidad de almacenar y acceder a documentos XML, generar documentos XML a partir de datos relacionales ya existentes e insertar filas en tablas relacionales desde documentos XML. XML Extender proporciona nuevos tipos de datos, funciones y procedimientos almacenados para gestionar los datos XML en DB2 UDB.

XML Extender está disponible en los siguientes sistemas operativos:

- Windows® NT
- Windows 2000
- AIX®
- Entorno operativo Solaris™
- Linux
- HP-UX en una plataforma PA-RISC
- OS/390® y z/OS™
- iSeries™

La única plataforma de 64 bits que soporta DB2 XML Extender en la actualidad es la plataforma HP-UX.

Antes de comenzar a trabajar con DB2 UDB XML Extender, es posible que desee establecer el lugar en que se guardarán los archivos temporales. La variable de entorno DB2DXXTEMP controla la ubicación de los archivos temporales de XML Extender. Si la variable no está establecida, el valor de TMP determinará la ubicación de los archivos temporales. Para establecer el valor de DB2DXXTEMP en Windows 2000:

1. Asegúrese de haber iniciado sesión con el ID de usuario que utilice con DB2.
2. Pulse **Inicio—>Configuración—>Panel de control.**
3. Efectúe una doble pulsación sobre el icono **Sistema**.
4. Pulse **Variables de entorno** en la pestaña Avanzado del cuaderno **Propiedades del sistema**.
5. Pulse **Nueva** en la sección **Variables del sistema**. Entre `DB2DXXTEMP` como nombre de la variable y entre un valor para dicha variable como, por ejemplo, `C:\temp`.
6. Cierre todas las ventanas y reinicie el sistema.

Conceptos relacionados:

- "Documentos XML" en la página 4
- "Características de XML Extender" en la página 5
- "Lección: Almacenamiento de un documento XML en una columna XML" en la página 8
- "Lección: Composición de un documento XML" en la página 19
- "Lecciones de la guía de aprendizaje de XML Extender" en la página 8

Documentos XML

Las empresas suelen compartir datos entre las distintas aplicaciones, por ello dichas empresas deben afrontar continuamente el problema de replicar, transformar, exportar o guardar los datos en formatos que puedan ser importados en otras aplicaciones. 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 asegurarse de la coherencia de los datos. Esta comprobación manual consume tiempo y dinero.

Una de las formas de abordar este problema es que el desarrollador de aplicaciones escriba aplicaciones *ODBC (Open Database Connectivity)*, una API (interfaz de programación de aplicaciones) estándar para el acceso a datos para sistemas de gestión de bases de datos relacionales y no relacionales. Estas aplicaciones ODBC guardan 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 conveniente para otra aplicación. Es necesario escribir las aplicaciones de bases de datos para convertir los datos en el formato requerido por la aplicación. Las aplicaciones cambian rápidamente y se vuelven obsoletas con la misma rapidez. Las aplicaciones que convierten datos a HTML proporcionan soluciones de presentación, pero en la práctica los datos presentados no se pueden utilizar para ninguna otra finalidad. Se hace necesario un método que separe los datos de la presentación para poder proporcionar una manera práctica de intercambiar datos entre aplicaciones.

XML—*eXtensible Markup Language*—aborda este problema. El lenguaje XML es extensible porque es un metalenguaje que permite al usuario crear su propio lenguaje dependiendo de las necesidades de la empresa. XML se utiliza para capturar no tan sólo los datos para una aplicación determinada, sino también la estructura de datos. Aunque no es el único formato de intercambio de datos, XML ha surgido como el estándar aceptado. Las aplicaciones que cumplen este estándar pueden compartir datos sin transformarlos primero utilizando formatos exclusivos.

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 se está utilizando una aplicación de gestión de proyectos determinada y desea compartir parte de sus datos con una aplicación de agenda. Su aplicación de gestión de proyectos podría exportar tareas en formato XML, que a continuación, podrían importarse tal cual a la aplicación de agenda. En el mundo interconectado de hoy en día, los proveedores de aplicaciones tienen muchos motivos para convertir el intercambio mediante XML en una característica básica de sus aplicaciones.

Manejo de los datos XML en DB2

Aunque XML soluciona muchos problemas al proporcionar un formato estándar para el intercambio de datos, existen todavía cuestiones por abordar. Al crear una aplicación de datos para la empresa, es necesario plantearse cuestiones como las siguientes:

- ¿Con qué frecuencia es necesario replicar los datos?
- ¿Qué clase de información hay que compartir entre aplicaciones?
- ¿Con qué rapidez se puede buscar la información necesaria?

- ¿Cómo puedo hacer que una determinada acción, como por ejemplo, añadir una nueva entrada, desencadene un intercambio automático de datos entre todas mis 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 de manera más eficaz los resultados XML que las otras aplicaciones necesitan. Con XML Extender, puede sacar provecho de la potencia de DB2® en muchas aplicaciones XML.

Con el contenido de los documentos XML estructurados dentro de una base de datos DB2 UDB, puede combinar la información XML estructurada con datos relacionales normales. Dependiendo de la aplicación, puede elegir si desea almacenar documentos XML completos en DB2 usando los tipos definidos por el usuario proporcionados para datos XML (tipos de datos XML), o bien puede correlacionar el contenido XML como tipos de datos base en tablas relacionales. Para tipos de datos XML, XML Extender añade la posibilidad de busca diversos tipos de datos de valores o elementos XML, además de la búsqueda de texto estructural que DB2 Universal Database™ proporciona.

XML Extender proporciona dos formas de almacenar y acceder a los datos XML en DB2:

Método de columna XML

Almacena documentos XML completos como datos de columna o bien externamente como un archivo y al mismo tiempo extrae el valor de elemento o atributo XML necesario y lo almacena en *tablas auxiliares*, tablas indexadas para búsquedas de alta velocidad. Con el almacenamiento de documentos en forma de datos de columna, puede:

- Realizar búsquedas rápidas en elementos o atributos XML que hayan sido extraídos o almacenados en tablas auxiliares como tipos de datos básicos SQL y que se hayan indexado.
- Actualizar el contenido de un elemento XML o del valor de un atributo XML.
- Extraer elementos o atributos XML dinámicamente utilizando consultas SQL.
- Validar documentos XML durante la inserción o actualización.
- Realizar búsquedas de texto estructural con Net Search Extender.

Método de colección XML

Compone y descompone el contenido de documentos XML con una o más tablas relacionales.

Características de XML Extender

XML Extender proporciona las siguientes características para ayudarle a gestionar y a hacer uso de los datos XML con DB2®:

- Herramientas de administración para ayudarle a gestionar la integración de datos XML en tablas relacionales
- Métodos de almacenamiento y acceso para datos XML dentro de la base de datos
- Un depósito de DTD (definición de tipos de datos), para almacenar las definiciones de tipos de datos que se utilizan para validar datos XML
- La posibilidad de validar documentos XML utilizando un esquema

- Un archivo de correlación llamado archivo DAD (Definición de acceso a documento), que se utiliza para correlacionar documentos XML con datos relacionales
- Vías de ubicación para especificar la ubicación de un elemento o atributo dentro de un documento XML

Herramientas de administración: Las herramientas de administración de XML Extender le ayudan a habilitar la base de datos y columnas de tabla para XML y a correlacionar los datos XML con estructuras relacionales de DB2.

Puede utilizar las herramientas siguientes para realizar tareas de administración en XML Extender:

- El mandato **dxxadm** proporciona una opción para realizar tareas de administración desde la línea de mandatos.
- Los procedimientos almacenados de administración de XML Extender permiten invocar mandatos de administración desde un programa.

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

Método de columna XML
Este método le ayuda a almacenar documentos XML completos en DB2. La columna XML es un método adecuado para archivar documentos. Los documentos se insertan en columnas habilitadas para XML y se pueden actualizar, recuperar y efectuar búsquedas en ellos. Los datos de elementos y atributos pueden correlacionarse con tablas de DB2 UDB (tablas auxiliares), que a su vez pueden indexarse para realizar búsquedas rápidas.

Método de colección XML
Este método le ayuda a correlacionar estructuras de documentos XML con tablas de DB2 UDB para poder componer documentos XML a partir de datos de DB2 UDB existentes o descomponer documentos XML, almacenando los datos sin códigos en tablas de DB2 UDB. Este método es ideal para aplicaciones de intercambio de datos, en especial cuando se actualiza con frecuencia el contenido de los documentos XML.

DTD: XML Extender también permite almacenar las DTD, el conjunto de declaraciones para elementos y atributos XML. Cuando se *habilita* para XML, se crea una tabla de depósito de 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 se utilizan para validar la estructura de los documentos XML.

Archivos DAD: El usuario especifica la forma en que XML Extender procesa los documentos XML estructurados utilizando un archivo de *definición de acceso de documentos (DAD)*. El archivo DAD es un documento XML que correlaciona la estructura del documento XML con una tabla de DB2 UDB. El archivo DAD se utiliza para almacenar documentos XML en una columna o al componer o descomponer datos XML. El archivo DAD especifica si el usuario está almacenando documentos utilizando el método de columna XML o está definiendo una colección XML para componer o descomponer documentos.

Vías de ubicación: Una *vía de ubicación* especifica la ubicación de un elemento o atributo dentro de un documento XML. XML Extender utiliza la vía de ubicación para recorrer la estructura del documento XML y localizar elementos y atributos.

Por ejemplo, una vía de ubicación de `/Order/Part/Shipment/ShipDate` apunta al elemento ShipDate, que es un elemento hijo de los elementos Shipment, Part y Order, tal y como se muestra en el siguiente ejemplo:

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

La Figura 1 muestra un ejemplo de una vía de ubicación y su relación con la estructura del documento XML.

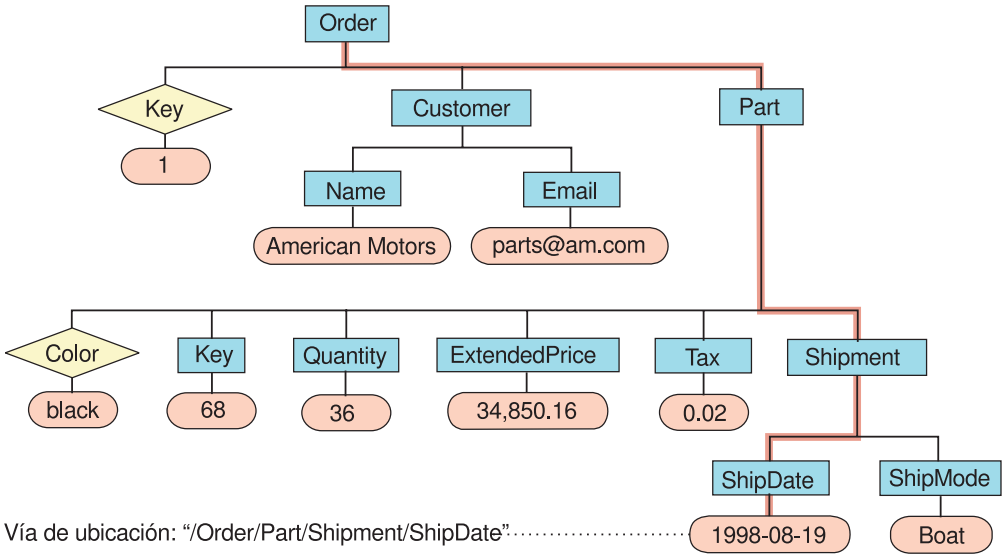


Figura 1. Almacenamiento de documentos como documentos XML estructurados en una columna de una tabla de DB2 UDB

La vía de ubicación se utiliza en los casos siguientes:

Columnas XML

- Se utilizan para identificar los elementos y atributos a extraer o actualizar cuando se utilizan las funciones definidas por el usuario de XML Extender.
- También se utilizan para correlacionar el contenido de un elemento o atributo XML con una tabla auxiliar.

Colecciones XML

Se utilizan para alterar temporalmente los valores en el archivo DAD de un procedimiento almacenado.

Para especificar la vía de ubicación, XML Extender utiliza un subconjunto del *XML Path Language (XPath)*, el lenguaje para indicar la ubicación de las partes de un documento XML.

Para obtener más información acerca de Xpath, consulte la siguiente página Web:

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

Conceptos relacionados:

- "Manejo de los datos XML en DB2" en la página 4
- "Lección: Almacenamiento de un documento XML en una columna XML" en la página 8
- "Lección: Composición de un documento XML" en la página 19
- "Lecciones de la guía de aprendizaje de XML Extender" en la página 8

Lecciones de la guía de aprendizaje de XML Extender

Esta guía de aprendizaje muestra cómo utilizar XML Extender para acceder y modificar los datos XML para las aplicaciones. Se proporcionan tres lecciones:

- Almacenamiento de un documento XML en una columna XML
- Composición de un documento XMLComposición de un documento XML
- Limpieza de la base de datosLimpieza de la base de datos

Siguiendo las lecciones de la guía de aprendizaje, puede configurar una base de datos utilizando los datos de ejemplo proporcionados, correlacionar datos SQL con un documento XML, almacenar documentos XML en la base de datos y, a continuación, buscar y extraer datos de los documentos XML.

En las lecciones de administración, utilizará la ventana de mandatos de DB2® con los mandatos de administración de XML Extender. En las lecciones de gestión de datos XML, utilizará las UDF y procedimientos almacenados de XML Extender. La mayoría de los ejemplos en el resto de esta guía se basan en los datos de ejemplo que se utilizan en esta sección.

Requisitos previos

Caso 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. A continuación, extraerá la información de una base de datos existente de pedidos de compras, SALES_DB.

Conceptos relacionados:

- "Herramientas de administración para XML Extender" en la página 37
- "Visión general de la administración de XML Extender" en la página 39
- "Lección: Almacenamiento de un documento XML en una columna XML" en la página 8
- "Lección: Composición de un documento XML" en la página 19

Lección: Almacenamiento de un documento XML en una columna XML

XML Extender proporciona un método para almacenar y acceder a todos los documentos XML de la base de datos. El método de columna XML le permite almacenar el documento utilizando los tipos de archivos XML, indexar la columna en tablas auxiliares y, a continuación, consultar el documento XML o efectuar búsquedas en el mismo. Este método de almacenamiento es especialmente útil para las aplicaciones de archivo en las que los documentos no se actualizan con frecuencia.

Esta lección muestra cómo utilizar el método de almacenamiento y acceso de columna XML.

El caso:

Se le ha asignado la tarea de archivar los datos de ventas del departamento de servicios. Los datos de ventas que necesita utilizar se almacenan en documentos XML que utilizan la misma DTD.

El departamento de servicios ha proporcionado una estructura recomendada para los documentos XML y ha especificado los datos de elementos que serán consultados con mayor frecuencia. El departamento de servicio quiere que los documentos XML se almacenen en la tabla SALES_TAB de la base de datos SALES_SB y desearía poder buscarlos con rapidez. La tabla SALES_TAB contendrá dos columnas con datos sobre cada venta y una tercera contendrá el documento XML. Esta columna se llama ORDER.

Para almacenar este documento XML en la tabla SALES_TAB, deberá:

1. Determinar los tipos definidos por el usuario (UDT) de XML Extender en los cuales se va a almacenar el documento XML, así como los elementos y atributos que se van a consultar con frecuencia.
2. Configurar la base de datos SALES_DB para XML.
3. Crear la tabla SALES_TAB y habilitar la columna ORDER para que pueda almacenar el documento intacto en DB2®.
4. Insertar una DTD para el documento XML para la validación.
5. Almacenar el documento como un tipo de datos XMLVARCHAR.

Cuando habilite la columna, deberá definir 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.

La tabla SALES_TAB se describe en la Tabla 1. La columna XML que se va a habilitar para XML, ORDER, se muestra en cursiva.

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

Scripts y ejemplos:

Para esta guía de aprendizaje, utilizará un conjunto de scripts para configurar su entorno y ejecutar los pasos de las lecciones. Estos scripts se encuentran en el directorio *instalación_dxx*/samples/db2xml/cmd (donde *instalación_dxx* es el directorio en el que se han instalado los archivos de XML Extender).

Estos scripts son:

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

getstart_prep.cmd
Vincula la base de datos con los procedimientos almacenados de XML Extender y habilita la base de datos para XML Extender.

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_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.

Planificación del modo de almacenar el documento:

Antes de utilizar XML Extender para almacenar los documentos, necesitará:

- Comprender la estructura del documento XML.
- Determinar el tipo definido por el usuario XML en el que almacenar el documento XML.
- Determinar los elementos y atributos XML que el departamento de servicios va a buscar con más frecuencia, para que su contenido pueda almacenarse en tablas e indexarse para mejor el rendimiento.

Las siguientes secciones describen cómo tomar estas decisiones.

Estructura del documento XML:

La estructura del documento XML de esta lección reúne información para un pedido específico que está estructurado mediante la clave de pedido como el nivel superior, después la información de cliente, pieza y envío en el siguiente nivel.

Esta lección proporciona la DTD de ejemplo para que el usuario comprenda y valide la estructura del documento XML.

Determinación del tipo de datos XML para la columna XML:

XML Extender proporciona los tipos XML definidos por el usuario XML que se pueden utilizar para definir una columna para albergar documentos XML. Estos tipos de datos son los siguientes:

XMLVARCHAR
Utilizado para documentos menores de 3 kilobytes almacenados en DB2. El tamaño máximo de los documentos XMLVARCHAR puede definirse como 32672 bytes de tamaño.

XMLCLOB
Utilizado para documentos mayores de 3 kilobytes almacenados en DB2. El tamaño máximo de documento es de 2 gigabytes.

XMLFILE
Se utiliza para documentos almacenados fuera de DB2.

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

Determinación de los elementos y atributos que deben buscarse:

Cuando haya comprendido la estructura del documento XML y las necesidades de la aplicación, podrá determinar qué elementos y atributos se buscarán o extraerán con más frecuencia o aquellos cuya consulta sea la más costosa. El departamento de servicios consultará frecuentemente la clave de pedido, el nombre del cliente, precio y la fecha de envío de un pedido; necesitarán también un rendimiento rápido de dichas 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
nombre de cliente	/Order/Customer/Name
precio	/Order/Part/ExtendedPrice
fecha de envío	/Order/Part/Shipment/ShipDate

Correlación del documento XML con las tablas auxiliares:

Para correlacionar los documentos XML con una tabla auxiliar, debe crear un archivo DAD para la columna XML. El archivo DAD se utiliza para almacenar el documento XML en DB2. También correlaciona el contenido del elemento y atributo XML con las tablas auxiliares de DB2 UDB utilizadas para el indexado, lo cual mejora el rendimiento de la búsqueda.

Después de identificar los elementos y atributos que desea buscar, debe determinar cómo deben organizarse en las tablas auxiliares, cuántas tablas utilizar y qué columnas habrá en cada tabla. Organice las tablas auxiliares colocando información similar en la misma tabla. La estructura del documento 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 el documento, el elemento pieza puede repetirse varias veces y, por lo tanto, los elementos precio y fecha pueden aparecer varias veces. Los elementos que pueden aparecer varias veces deben estar en sus propias tablas auxiliares.

Debe asimismo determinar qué tipos base de DB2 USB deben utilizar los valores del elemento o atributo, tarea que viene determinada según el formato de los datos.

- Si los datos son de tipo texto, utilice VARCHAR.
- Si los datos son de tipo entero, utilice INTEGER.
- Si los datos son de tipo fecha y desea realizar búsquedas de intervalos de fechas, utilice DATE.

En esta guía de aprendizaje, los elementos y atributos se correlacionan con ORDER_SIDE_TAB, PART_SIDE_TAB o SHIP_SIDE_TAB. Las tablas siguientes muestran con qué tabla se correlaciona cada elemento o atributo.

ORDER_SIDE_TAB

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

SHIP_SIDE_TAB

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

Creación de la base de datos SALES_DB:

En esta tarea, debe crear una base de datos de ejemplo y crear la base de datos para XML.

Para crear la base de datos:

1. Asegúrese de que el administrador de DB2 UDB ha habilitado el servidor de bases de datos.
2. Pase al directorio `instalación_dxx/samples/db2xml/cmd`, donde `instalación_dxx es el directorio en el que ha instalado los archivos de XML Extender.` Los archivos de ejemplo contienen referencias a archivos que utilizan nombres de vías de acceso absolutos. Examine los archivos de ejemplo y cambie esas vías de acceso para que apunten a sus directorios.
3. En las plataformas Windows®, abra una ventana de mandatos de DB2 UDB escribiendo:

   ```
   DB2CMD
   ```

4. Ejecute el mandato **getstart_db**:

Habilitación del servidor:

Para guardar información de XML en la base de datos, debe habilitarla para XML Extender. Al habilitar una base de datos para XML, XML Extender:

- Crea los tipos definidos por el usuario (UDT), funciones definidas por el usuario (UDF) y los procedimientos almacenados
- Crea y llena las tablas de control con los metadatos necesarios para XML Extender
- Crea el esquema DB2XML y asigna los privilegios necesarios

Para habilitar la base de datos para XML:

Utilice uno de los métodos siguientes para habilitar la base de datos.

Ejecute el script siguiente:

```
getstart_prep.cmd
```

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

```
dxxadm enable_dbSALES_DB
```

Habilitación de la columna XML y almacenamiento del documento:

En esta lección, habilitará una columna para XML Extender y almacenará un documento XML en la columna. Para realizar estas tareas:

1. Almacene la DTD en el depósito de DTD.
2. Cree un archivo DAD para la columna XML.
3. Cree la tabla SALES_TAB.
4. Añada la columna de tipo XML.
5. Habilite la columna XML.
6. Visualice la columna y las tablas auxiliares.
7. Indexe las tablas auxiliares para una búsqueda estructural.
8. Almacene el documento XML.
9. Consulte el documento XML.

Almacenamiento de la DTD en el depósito de DTD:

Puede utilizar una DTD para validar datos XML en una columna XML. XML Extender crea una tabla en la base de datos habilitada para XML, denominada DTD_REF. Dicha tabla se conoce como el depósito de DTD y se encuentra disponible para almacenar las DTD. Cuando valida documentos XML, debe almacenar la DTD en este depósito. La DTD de esta lección se encuentra en

```
instalación_dxx/samples/db2xml/dtd/getstart.dtd
```

donde *instalación_dxx* es el directorio donde se ha instalado DB2 XML Extender.

- Entre el siguiente mandato SQL INSERT en la misma línea de mandatos de DB2:

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

- Ejecute el siguiente archivo de mandatos para insertar la DTD:

```
getstart_insertDTD.cmd
```

Creación de un archivo DAD para la columna XML:

Esta sección explica cómo crear un archivo DAD para la columna XML. En el archivo DAD, se especifica el método de acceso y almacenamiento utilizado en la columna XML. Asimismo, se definen las tablas y columnas para el indexado.

En los pasos siguientes, se hace referencia a los elementos de la DAD como *códigos* y a los elementos del documento XML como *elementos*. Un ejemplo de archivo DAD similar al que creará está en `instalación_dxx/samples/db2xml/dad/getstart_xcolumn.dad` . Presenta algunas pequeñas diferencias respecto al archivo que se genera en los pasos siguientes. Si lo utiliza para la lección, tenga en cuenta que puede que las vías del archivo sean diferentes de las de su entorno; el valor <validation> está establecido en NO, en vez de en YES.

Para crear un archivo DAD para su uso con una columna XML:

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

   Todos los códigos que se utilizan en el archivo DAD distinguen entre mayúsculas y minúsculas.
2. Cree la cabecera de la DAD, con las declaraciones XML y DOCTYPE.

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

   El archivo DAD es un documento XML y requiere declaraciones XML.
3. Inserte los códigos de apertura y cierre (<DAD> y </DAD>) para el documento. Los demás códigos se encuentran dentro de estos códigos.
4. Inserte los códigos de apertura y cierre (<DTDID> y </DTDID>) con un ID de DTD para especificar si el documento será validado:

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

   Verifique que esta serie coincide con el valor utilizado como primer valor del parámetro al insertar la DTD en la tala del depósito de DTD. Por ejemplo, la vía de acceso que utilice para el DTDID puede ser distinta de la serie que inserte en la tabla de referencia de DTD si se encuentra trabajando en una unidad diferente.
5. Inserte los códigos de apertura y cierre (<validation> y </validation>) y la palabra clave YES o NO para indicar si desea que XML Extender valide la estructura del documento XML utilizando la DTD que ha insertado en la tabla de referencia de DTD. Por ejemplo:

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

   El valor del código <validation> puede estar en mayúsculas o en minúsculas.
6. Inserte los códigos de apertura y cierre (<Xcolumn> y </Xcolumn>) para especificar que el método de almacenamiento es la columna XML.
7. Cree tablas auxiliares. Para cada tabla auxiliar que desee crear:
   a. Inserte los códigos de apertura y cierre (<table> y </table>) para cada tabla auxiliar que vaya a generarse, y especifique el nombre de la tabla auxiliar entre comillas dobles utilizando el atributo ″name=″ como se muestra a continuación:

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

b. Dentro de los códigos de la tabla, inserte un código <column> para cada columna que desee que la tabla auxiliar contenga. Cada columna tiene cuatro atributos: name, type, path y multi_occurrence:

name Especifica el nombre de la columna que se crea en la tabla auxiliar.

type Indica el tipo de datos de la tabla auxiliar para cada elemento o atributo indexado.

path Especifica la vía de ubicación del documento XML para cada elemento o atributo que vaya a indexarse.

multi_occurrence
Indica si el elemento o el atributo al que se hace referencia mediante el atributo de vía (path) puede aparecer más de una vez en el documento XML. Los posibles valores de *multi_occurrence* son *YES* o *NO.* Si el valor es *NO*, podrá mencionar más de un código de columna en la tabla auxiliar. Si el valor es *YES*, sólo puede mencionar una columna en la tabla auxiliar.

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

8. Asegúrese de que tiene los códigos de cierre necesarios:
   - Un código de cierre </Xcolumn> después del último código </table>
   - Un código de cierre </DAD> después de código </Xcolumn>
9. Guarde el archivo con el siguiente nombre:

   ```
   getstart_xcolumn.dad
   ```

Puede comparar el archivo que acaba de crear con el archivo de ejemplo, instalación_dxx/samples/db2xml/dad/getstart_xcolumn.dad . Este archivo es una copia de trabajo del archivo DAD necesario para habilitar la columna XML y para crear las tablas auxiliares. Los archivos de ejemplo contienen referencias a archivos que utilizan vías de acceso absolutas. Examine los archivos de ejemplo y cambie esas vías de acceso para que apunten a sus directorios.

Creación de la tabla SALES_TAB:

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

Para crear la tabla:

Entre la siguiente sentencia CREATE TABLE utilizando uno de los siguientes métodos:

- Entre los siguientes mandatos de DB2 UDB:

```
DB2 CONNECT TO SALES_DB

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

- Ejecute el siguiente archivo de mandatos para crear la tabla:

```
getstart_createTabCol.cmd
```

Adición de la columna de tipo XML:

Añada una nueva columna a la tabla SALES_TAB. Esta columna contendrá el documento XML intacto generado anteriormente y debe ser de la UDT de UDT. XML Extender proporciona varios tipos de datos. En esta lección, almacenará el documento como XMLVARCHAR.

Para añadir la columna de tipo XML:

Ejecute la sentencia de SQL ALTER TABLE utilizando uno de los siguientes tres métodos:

- Entre la sentencia de SQL siguiente:

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

- Ejecute el siguiente archivo de mandatos para modificar la tabla:

```
getstart_alterTabCol.cmd
```

Habilitación de la columna XML:

Después de crear la columna de tipo XML, podrá habilitarla para XML Extender. Al habilitar la columna, 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 contiene el documento XML unido a las columnas de tablas auxiliares. Puede especificar la vista por omisión al consultar el documento XML. En esta lección, especificará una vista con el parámetro `-v`.
- Determinar si desea especificar una clave primaria como *ROOT ID*, el nombre de columna de la clave primaria en la tabla de aplicación y un identificador exclusivo que asocia todas las tablas auxiliares a la tabla de aplicación. Si no especifica una clave primaria, XML Extender añade la columna DXXROOT_ID a la tabla de aplicación y a las tablas auxiliares.

  La columna ROOT_ID se utiliza como clave para unir las tablas de aplicación y auxiliares, lo cual permite que XML Extender actualice 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`. XML Extender utilizará la columna especificada como el ROOT_ID y añadirá la columna a las tablas auxiliares.
- Determinar si desea especificar un espacio de tabla o utilizar el espacio de tabla por omisión. En esta lección, utilizará el espacio de tabla por omisión.

Para habilitar la columna para XML:

Ejecute el mandato **dxxadm enable_column**, utilizando uno de los tres métodos siguientes:

Línea de mandatos:

- Escriba el mandato siguiente:

```
dxxadm enable_column SALES_DB SALES_TAB ORDER getstart_xcolumn.dad
        -v SALES_ORDER_VIEW -r INVOICE_NUM
```

- Ejecute el siguiente archivo de mandatos para habilitar la columna:

```
getstartenableCol.cmd
```

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

Importante: no modifique las tablas auxiliares de este modo. Sólo se deben realizar actualizaciones en las tablas auxiliares mediante las actualizaciones del propio documento XML. XML Extender actualizará automáticamente las tablas auxiliares al actualizar el documento XML en la columna XML.

Visualización de la columna y de las tablas auxiliares:

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

Para visualizar la columna XML y las columnas de las tablas auxiliares:

Entre la siguiente sentencia SELECT de SQL en la línea de mandatos:

```
SELECT * FROM SALES_ORDER_VIEW
```

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

Indexación de tablas auxiliares para la búsqueda estructural:

La creación de índices en tablas auxiliares le permite realizar búsquedas estructural rápidas del documento XML. En esta sección, creará índices en columnas clave de las tablas auxiliares que se crearon al habilitar la columna XML, ORDER. El departamento de servicios ha especificado las columnas que probablemente van a consultar con más frecuencia sus empleados. La Tabla 3 describe estas columnas que se van 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:

Ejecute los siguientes mandatos CREATE INDEX SQL utilizando uno de los tres métodos siguientes:

- Escriba los mandatos siguientes:

```
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)
```

- Ejecute el siguiente archivo de mandatos para crear los índices:

  ```
  getstart_createIndex.cmd
  ```

Almacenamiento del documento XML:

Ahora que se ha habilitado una un columna para que contenga el documento XML y que se han indexado las tablas auxiliares, es posible almacenar el documento utilizando las funciones que XML Extender proporciona. Al almacenar los datos en una columna XML, puede utilizar las funciones de conversión por omisión o las UDF de XML Extender. Debido a que almacenará un objeto del tipo base VARCHAR en una columna perteneciente al tipo XMLVARCHAR de XML, utilizará la función por omisión de conversión de datos.

Para almacenar un documento XML:

1. Abra el documento XML `instalación_dxx/samples/db2xml/xml/getstart.xml` Asegúrese de que la vía de acceso del archivo de DOCTYPE coincide con el ID de DTD especificado en la DAD y cuando insertó la DTD en el depósito de DTD. Puede verificar que coincidan consultando la tabla DB2XML.DTD_REF y comprobando el elemento DTDID del archivo DAD. Si está utilizando una unidad y estructura de directorios distintas de las asignadas por omisión, necesitará cambiar la vía de acceso en la declaración DOCTYPE para que coincida con la estructura de directorios.
2. Ejecute el mandato INSERT de SQL, utilizando uno de los métodos siguientes:
   - Escriba el siguiente mandato INSERT de SQL:

     ```
     DB2 INSERT INTO SALES_TAB (INVOICE_NUM, SALES_PERSON, ORDER) VALUES
     ('123456', 'Sriram Srinivasan', DB2XML.XMLVarcharFromFile
     ('instalación_dxx/samples/db2xml/
     /xml/getstart.xml'))
     ```

   - Ejecute el siguiente archivo de mandatos para almacenar el documento:

     ```
     getstart_insertXML.cmd
     ```

Verifique que efectivamente las tablas se han actualizado. Ejecute las siguientes sentencias SELECT para las tablas en la línea de mandatos.

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

Consulta del documento XML:

Puede buscar el documento XML mediante una consulta directa en 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:

Ejecute la sentencia SELECT de SQL, utilizando uno de los métodos siguientes:

- Escriba la siguiente sentencia SELECT de SQL:

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

- Ejecute el siguiente archivo de mandatos para buscar el documento:

```
getstart_queryCol.cmd
```

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

Ha completado la guía de iniciación para almacenar documentos XML en tablas de DB2 UDB. Por ejemplo:

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

Sriram Srinivasan
```

Conceptos relacionados:

- "Introducción a XML Extender" en la página 3
- "Lección: Composición de un documento XML" en la página 19
- "Lecciones de la guía de aprendizaje de XML Extender" en la página 8

Lección: Composición de un documento XML

En esta lección aprenderá a componer un documento XML a partir de datos de DB2® existentes.

El caso:

Se le ha asignado la tarea de obtener información en una base de datos de pedidos de compra llamada SALES_DB, y de extraer la información solicitada de la misma y almacenarla en documentos XML. El departamento de servicios utilizará estos documentos XML cuando trabaje con pedidos y reclamaciones de los clientes. El departamento de servicios 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.

Para componer un documento XML necesitará planear y crear un archivo DAD que correlacione las columnas de las tablas relacionadas con una estructura de documento XML que proporcione un registro de pedidos de compra. Debido a que este documento se compone de varias tablas, creará una colección XML y asociará dichas 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 columnas con un asterisco son las columnas que el departamento de servicios ha solicitado que estén en la estructura del 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 utilizar XML Extender para componer los documentos, necesitará determinar la estructura del documento XML y cómo corresponde a la estructura de los datos de la base de datos. Esta sección proporciona una visión general de la estructura del documento XML solicitada por el departamento de servicios, y de la DTD utilizada para definir la estructura del documento XML. Esta sección también muestra cómo se correlaciona este documento con las columnas que contienen los datos utilizados para llenar 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 servicios desea un documento que esté estructurado según el número de pedido como nivel superior y, a continuación, la información de cliente, pieza y envío. El departamento de servicios quiere que la estructura del documento sea intuitiva y flexible, con elementos que 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.)

Después de diseñar la estructura del documento, cree una DTD para describir la estructura del documento XML. Esta lección proporciona un documento XML una

DTD para utilizarlas. Mediante la utilización de las reglas de la DTD y de la estructura jerárquica del documento XML, podrá crear un mapa jerárquico de correlaciones de los datos, como se muestra en la Figura 2.

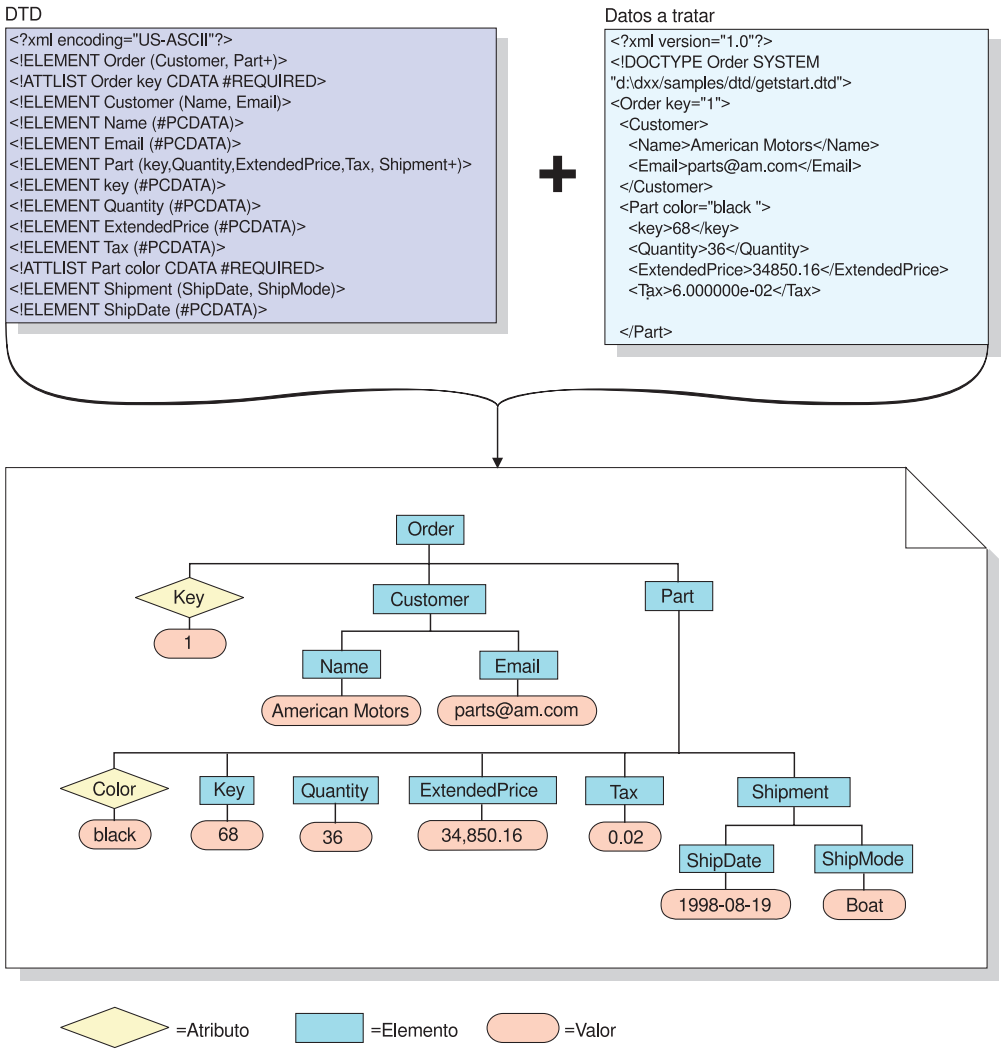


Figura 2. Estructura jerárquica de la DTD y el documento XML

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

Después de diseñar la estructura y crear la DTD, es necesario mostrar cómo se relaciona la estructura del documento con las tablas de DB2 UDB que utilizará para llenar los elementos y atributos. Puede correlacionar la estructura jerárquica con determinadas columnas de las tablas relacionales, como se muestra en la Figura 3 en la página 22.

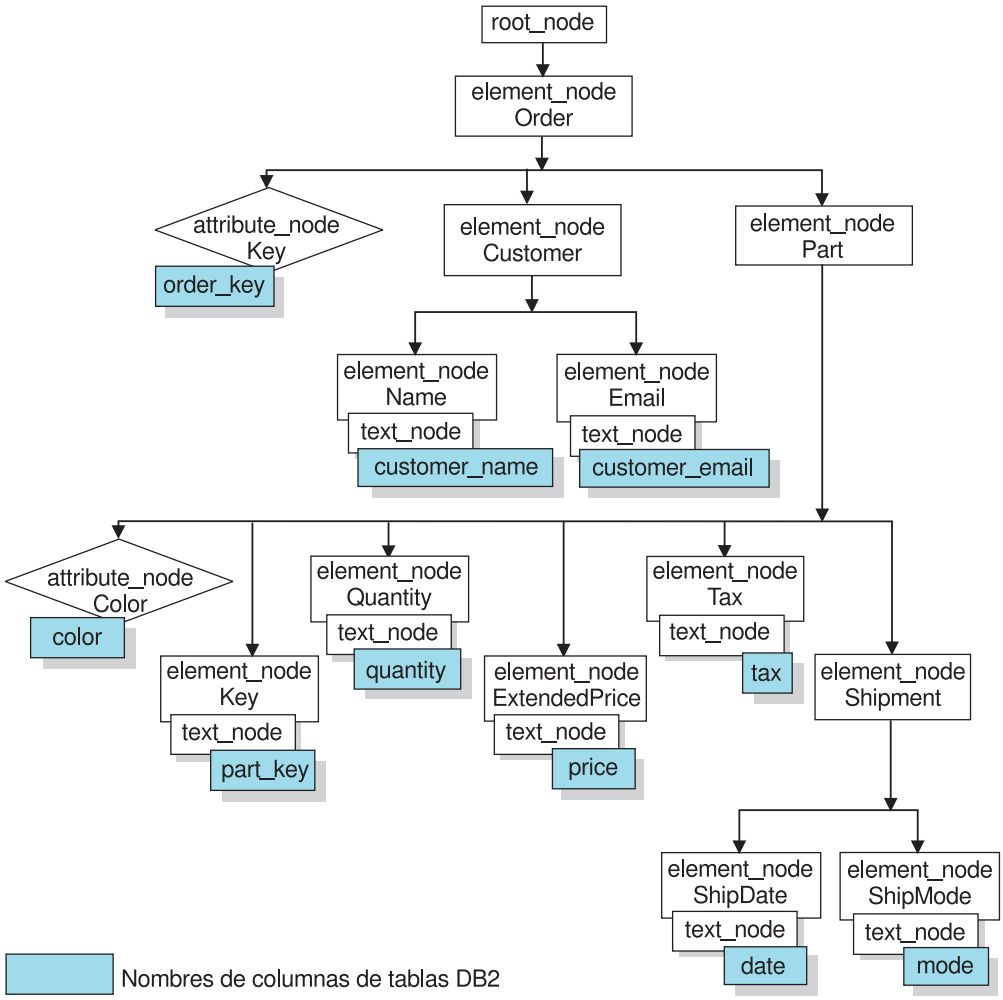


Figura 3. Documento XML correlacionado con columnas de tabla relacional

En esta figura se utilizan nodos para mostrar elementos, atributos y texto dentro de la estructura de documentos XML. Estos nodos se utilizan en el archivo DAD y se explican con más detalle en los pasos posteriores.

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 3, 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.

Scripts y ejemplos:

Esta lección proporciona un conjunto de scripts para configurar el entorno. `Estos scripts se encuentran en el directorio instalación_dxx/samples/db2xml/xml (donde instalación_dxx es el directorio donde se han instalado los archivos de XML)`

Los scripts son:

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

getstart_prep.cmd
Vincula la base de datos con los procedimientos almacenados de 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 del entorno de las lecciones:

En esta sección, se prepara la base de datos para su utilización con XML Extender. Realizará 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.

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 *instalación_dxx*`/samples/db2xml/xml, donde` *instalación_dxx* `es el directorio donde se han instalado los archivos de XML Extender.` Los archivos de ejemplo contienen referencias a los archivos que utilizan nombres de vías de acceso absolutas. Examine los archivos de ejemplo y cambie esas vías de acceso para que apunten a sus directorios.
2. Abra la ventana de mandatos de DB2 UDB:
   ```
   DB2CMD
   ```
3. Ejecute el archivo de mandatos de base de datos, utilizando uno de los métodos siguientes:

   Escriba el mandato siguiente:
   ```
   getstart_db.cmd
   ```

Habilitación de la base de datos:

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

- Crea tipos definidos por el usuario (UDT), funciones definidas por el usuario (UDF) y procedimientos almacenados.
- Crea y llena 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, utilice uno de los siguientes métodos:

Ejecute el siguiente script para habilitar la base de datos SALES_DB:

```
getstart_prep.cmd
```

Estos scripts vinculan la base de datos a los procedimientos almacenados de XML Extender y a la CLI de DB2 UDB. También ejecuta la opción del mandato **dxxadm** que habilita la base de datos:

```
dxxadm enable_db SALES_DB
```

Creación del archivo DAD para la colección XML:

Dado que los datos ya existen en varias tablas, creará una colección XML, que asocia las tablas con el documento XML. La colección se define creando un archivo DAD.

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*. Puede encontrar un ejemplo de archivo DAD similar al que se va a crear en *instalación_dxx*`/samples/db2xml/dad/getstart_xcollection.dad.`

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 y puede que deba actualizar el archivo de ejemplo.

Para crear el archivo DAD para componer un documento XML:

1. Desde el directorio `instalación_dxx/samples/db2xml/xml`, abra un editor de texto y cree un archivo denominado `getstart_xcollection.dad`.
2. Cree la cabecera de la DAD, utilizando el texto siguiente:

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

   Cambie `instalación_dxx` por el directorio donde se ha instalado DB2 XML Extender.
3. Inserte los códigos <DAD></DAD>. Los demás códigos se encuentran dentro de estos códigos.
4. Especifique los códigos <validation> </validation> para indicar si XML Extender valida la estructura del documento XML cuando inserta una DTD en la tabla de depósito de DTD. Esta lección no requiere ninguna DTD y el valor es **NO**.

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

   El valor de los códigos <validation> puede estar en mayúsculas o minúsculas.

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

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

6. A continuación del código <Xcollection>, incluya una sentencia de SQL para especificar las tablas y columnas que se utilizan para la colección XML. Este método se denomina correlación de SQL y es una de las dos maneras de correlacionar los datos relacionales con la estructura del documento XML. Escriba la sentencia siguiente:

```
<Xcollection
<SQL_stmt>
SELECT o.order_key, customer_name, customer_email, p.part_key, color,
   quantity, price, tax, ship_id, date, mode from order_tab o, part_tab p,
   table (select substr(char(timestamp(db2xml.generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab)  s
         WHERE o.order_key = 1 and
              p.price > 20000 and
              p.order_key = o.order_key and
              s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id
</SQL_stmt> </Xcollection>
```

   Esta sentencia de SQL sigue las siguientes directrices cuando utiliza la correlación de SQL. Consulte la Figura 3 en la página 22 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 de los elementos pedido y cliente van primero, las del elemento pieza en segundo lugar y las del envío en tercero.
   - Se agrupan las columnas de una sección que se repite, o de una sección que no se repite, de la plantilla que necesita datos de la base de datos. 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 clave y p.PART_KEY precede a las columnas del elemento Pieza.
   - La tabla SHIP_TAB no tiene una única columna condicional de clave y, por lo tanto, se utiliza la función incorporada de DB2 generate_unique para generar la columna SHIP_ID.
   - A continuación, se listan las columnas de ID de objeto en orden de superior a inferior en una sentencia ORDER BY. Las columnas en ORDER BY no están calificadas por ningún nombre de esquema y tabla, y coinciden con los nombres de columna en la cláusula SELECT.

7. Añada la siguiente información de prólogo (prolog) 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
"instalación_dxx/samples/db2xml/dtd/getstart.dtd"</doctype>
```

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

    element_node (nodo de elemento)
    Especifica el elemento del documento XML. Los nodos de elemento (element_node) pueden tener nodos de elemento 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 nodos de elemento (element_node) de nivel inferior.

    La Figura 3 en la página 22 muestra la estructura jerárquica del documento XML y las columnas de tabla de DB2 UDB e indica qué tipos de nodos se utilizan. Los recuadros sombreados indican que los nombres de columnas de tabla de DB2 UDB desde donde se extraen los datos para componer el documento XML.

    Para añadir cada tipo de nodo, escriba uno cada 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>
```

    El elemento hijo <Shipment> tiene un atributo 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 del documento XML. Estos atributos se anidan en el element_node correspondiente. Los nodos de atributo 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 elemento de nivel inferior element_node, defina códigos <text_node> para indicar que el elemento XML contiene datos de caracteres a extraer de DB2 UDB 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>. Las columnas definidas en el código <column/> 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. Asegúrese de que tiene los códigos de cierre necesarios:
    - Un código </root_node> final después del último código </element_node>
    - Un código </Xcollection> final después del último código </root_node>
    - Un código </DAD> final después del último código </Xcollection>
12. Guarde el archivo `getstart_xcollection.dad`.

Puede comparar el archivo creado con el archivo de ejemplo *instalación_dxx*`/samples/db2xml/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 vías de ubicación y los nombres de vías de acceso de archivos que pueden necesitar cambiarse en el entorno para que 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. Al habilitar la colección se registra en la tabla XML_USAGE y ayuda a mejorar el rendimiento cuando se especifica el nombre de la colección (en lugar del nombre de archivo DAD) cuando se ejecutan procedimientos almacenados. En estas lecciones no se habilita la colección.

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. Utilice uno de los métodos siguientes para llamar al procedimiento almacenado dxxGenXML:

   Escriba el mandato siguiente:

   ```
   getstart_stp.cmd
   ```

   El procedimiento almacenado compone el documento XML y lo almacena en la tabla RESULT_TAB.

   Si está ejecutando XML Extender en un entorno particionado de DB2 Enterprise Server Edition, compruebe haber creado una tabla de resultados con una clave de particionado calificada o haber creado una tabla de resultados en un grupo de nodos con un nodo único.

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

   - *instalación_dxx*`/samples/db2xml/c/tests2x.sqc` muestra cómo llamar al procedimiento almacenado utilizando SQL incorporado y generar el archivo ejecutable `tests2x`, utilizado por `getstart_stp.cmd`.
   - *instalación_dxx*`/samples/db2xml/cli/sql2xml.c` `dxxsamples/cli/sql2xml.c` muestra cómo llamar al procedimiento almacenado utilizando la CLI.
2. Exporte el documento XML desde la tabla a un archivo utilizando uno de los métodos siguientes para invocar la función de recuperación de XML Extender, Content():
   - Escriba los mandatos siguientes:

```
DB2 CONNECT TO SALES_DB

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

- Ejecute el siguiente archivo de mandatos para exportar el archivo:

  ```
  getstart_exportXML.cmd
  ```

Sugerencia: Este paso muestra cómo generar uno o más documentos XML compuestos utilizando la función de conjunto de resultados del procedimiento almacenados de DB2 UDB. La utilización de un conjunto de resultados permite obtener múltiples filas para generar más de un documento. A medida que vaya generando cada documento, puede exportarlos a un archivo. Este método es la manera más simple para mostrar la utilización de conjuntos de resultados. Para conocer maneras más eficientes de obtener datos, consulte los ejemplos de la CLI en *instalación_dxx*/samples/db2xml/cli.

Transformación de un documento XML en un archivo HTML:

Para mostrar los datos del documento XML en un navegador, debe transformar el documento XML en un archivo HTML utilizando una hoja de estilo y la función XSLTransformToFile.

Utilice los pasos siguientes para la transformación en un archivo HTML:

1. Genere una hoja de estilos utilizando un editor de textos y denomínela getstart.xsl:

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

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

     ...

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

   Puede encontrar un ejemplo de este archivo completado en el siguiente directorio:

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

2. Para cada elemento, cree un código con el formato siguiente:

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

   Este código se utilizará en las instrucciones de la transformación. Cree un código para cada elemento de la jerarquía del documento XML. Por ejemplo:

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

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

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

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

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

3. Para formatear el archivo HTML, utilice una lista que muestre la jerarquía de los elementos XML para hacer que los datos sean más legibles. Cree elementos de texto adicionales para describir los datos. Por ejemplo, puede que el archivo de hoja de estilo tenga el aspecto siguiente:

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

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

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

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

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

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

           <br/><br/>
      <ol type="a">
           <xsl:for-each select="Shipment">
        <li><b>Shipment</b><br/>
        <xsl:for-each select="ShipDate | ShipMode">
```

```
<xsl:value-of select="name()"/>
          <xsl:text> : </xsl:text>
          <xsl:value-of select="."/>
<xsl:text>, </xsl:text>
    </xsl:for-each>
   </li>
       </xsl:for-each>
      </ol><br/>
   </li>
    </xsl:for-each>
   </ol>
   </li>
    </xsl:for-each>
   </ol>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

4. Utilice Xpath para editar los códigos <xsl:value-of select=″xxx″> con datos del documento XML.

   Los códigos de elementos son <xsl:value-of select″.″>, donde el símbolo del punto (″.″) se utiliza para obtener datos de los elementos normales.

   Los códigos de atributos son <xsl:value-of select=″@attributname″>, donde el símbolo del ampersand (@) se añade al nombre del atributo que extraerá el valor del atributo. Puede utilizar <xsl:value-of select=″name()″> para obtener el nombre del código XML.

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

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

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

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

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

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

           <br/><br/>
```

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

5. Guarde la hoja de estilo.
6. Cree el archivo HTML de una de las maneras siguientes:
   - Utilice XSLTransformToFile:

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

   - Utilice el mandato siguiente:

     ```
     Getstart_xslt.cmd
     ```

Sólo es posible grabar el archivo de salida en un sistema de archivos accesible para el servidor de DB2 UDB.

Limpieza del entorno de la guía de aprendizaje:

Si desea limpiar el entorno de la lección, puede ejecutar uno de los scripts proporcionados o entrar los mandatos desde la línea de mandatos para:

- Inhabilitar la columna XML, ORDER.
- Descartar las tablas creadas en las lecciones.
- Suprimir la DTD de la tabla de depósito de DTD.

La base de datos SALES_DB no se inhabilita o descarta; la base de datos todavía se encuentra disponible para su uso con XML Extender. Es posible que reciba mensajes de error si no ha completado las dos lecciones de esta sección. Puede ignorar dichos errores.

Para limpiar el entorno de la guía de aprendizaje:

Ejecute el archivo de mandatos de limpieza, utilizando uno de los métodos siguientes:

- Escriba el mandato siguiente:

  ```
  getstart_clean.cmd
  ```

- Si desea inhabilitar la base de datos, puede ejecutar el siguiente mandato XML Extender desde la línea de mandatos:

  ```
  dxxadm disable_db SALES_DB
  ```

Este mandato descarta las tablas de control de administración DTD_REF y XML_USAGE, y elimina los tipos definidos por el usuario y las funciones proporcionadas por XML Extender.

- Si desea eliminar la base de datos, puede ejecutar el siguiente mandato desde la línea de mandatos

```
db2 drop database SALES_DB
```

Este mandato elimina SALES_DB.

Conceptos relacionados:

- "Introducción a XML Extender" en la página 3
- "Lección: Almacenamiento de un documento XML en una columna XML" en la página 8
- "Lecciones de la guía de aprendizaje de XML Extender" en la página 8

Parte 2. Administración

Esta parte describe cómo realizar tareas administrativas en XML Extender.

Capítulo 2. Administración

Herramientas de administración para XML Extender

Las herramientas de administración de XML Extender le ayudan a habilitar la base de datos y las columnas de tabla para XML, y a correlacionar datos XML con estructuras relacionales de DB2®. XML Extender proporciona para su uso la siguiente herramienta de la línea de mandatos así como interfaces de programación de tareas de administración.

XML Extender también proporciona las siguientes herramientas para completar las tareas de administración:

- El Asistente de administración de XML Extender proporciona una interfaz gráfica de usuario para las tareas de administración.
- El mandato **dxxadm** proporciona una opción para realizar tareas de administración desde la línea de mandatos.
- Los procedimientos almacenados de administración de XML Extender permiten invocar mandatos de administración desde un programa.

Preparación para administrar XML Extender

Para ejecutar XML Extender, deberá instalar el siguiente software.

Software necesario: XML Extender necesita DB2® Universal Database Versión 8.2.

Software opcional:

- Para la búsqueda de texto estructural, DB2 Universal Database Net Search Extender Versión 8.2, disponible con DB2 Universal Database Versión 8.2
- Para el Asistente de administración de XML Extender:
  - DB2 Universal Database Java Database Connectivity (JDBC)
  - SDK 1.1.7 o posterior, o bien RE 1.1.1, disponible con el Centro de control de DB2 UDB
  - JFC 1.1 con Swing 1.1, disponible con el Centro de control de DB2 UDB

Antes de instalar XML Extender, es necesario completar las siguientes tareas:

- Vincular XML Extender a la base de datos de DB2 UDB.

  Es necesario vincular XML Extender a cada base de datos. Para obtener un ejemplo, vea:

  instalación_dxx/samples/db2xml/cmd/getstart_prep.cmd
- Consultar las instrucciones de puesta a punto.
- Crear una base de datos para el acceso mediante XML.

Para realizar tareas de administración utilizando XML Extender, debe disponer de autorización DB2ADM.

Migración de XML Extender desde versiones anteriores

Si utiliza una versión anterior de DB2® XML Extender, deberá migrar cada base de datos habilitada para XML Extender antes de utilizar una base de datos habilitada para XML con XML Extender Versión 8.2.

El programa de migración ejecuta varios pasos dependiendo del nivel base de XML Extender que tenga. Los pasos que puede ejecutar el programa de migración son:

- Crear los UDT (tipos definidos por el usuario) XMLCLOB y las UDF (funciones definidas por el usuario) para su uso con bases de datos de Unicode y DBCS
- Crear procedimientos almacenados adicionales para dxxGenXML y dxxRetrieveXML, para dar soporte al uso de tablas temporales
- Modificar los procedimientos almacenados para utilizar el estilo de parámetro SQL como convenio de enlace para pasar parámetros
- Crear nuevas funciones definidas por el usuario para la validación de esquemas y DTD y la función XSLT.
- Crear nuevos procedimientos almacenados (dxxGenXMLCLOB y dxxRetrieveXMLCLOB) que devuelvan CLOB.
- Descartar y volver a crear las UDF (funciones definidas por el usuario) que permitan utilizar la posibilidad de paralelismo de las UDF escalares.

Nota: dxxEnableColl se denomina ahora dxxEnableCollection y dxxDisableColl, dxxDisableCollection. También se ha añadido un procedimiento almacenado db2xml.dxxDisableDB.

Al llamar a los procedimientos almacenados, utilice un punto (.) en lugar de un signo de exclamación (!) en el nombre del procedimiento. Por ejemplo, utilice *db2xml.dxxEnableColumn* en lugar de *db2xml!dxxEnableColumn*.

Procedimiento:

Para migrar una base de datos habilitada para XML y columnas habilitadas para XML:

1. Instale DB2 UDB XML Extender Versión 8.1.
2. Desde la línea de mandatos de DB2 UDB, entre:

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

Migración a XML Extender con fixpacks desde versiones anteriores

El programa de migración migra XML Extender a los fixpack actualizados desde releases anteriores. Debe realizar una copia de seguridad de la base de datos antes de ejecutar el programa de migración.

Para migrar la base de datos:

Los archivos de migración se encuentran en la imagen de instalación del CD del fixpack. Debe seguir estos pasos desde el mismo directorio en el que haya extraído los archivos del fixpack.

Si se encuentra en UNIX o Windows, siga los siguientes pasos.

1. Desde la línea de mandatos de DB2 UDB, entre:

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

2. Desde la línea de mandatos de DB2 UDB, entre:

   ```
   dxxMigv nombre_base_datos
   ```

Si no se realiza el paso de migración, es posible que se produzcan problemas y resultados imprevisible como anomalías al inhabilitar bases de datos y no poder acceder a las nuevas UDF.

Conceptos relacionados:

• "Migración de XML Extender desde versiones anteriores" en la página 38

Visión general de la administración de XML Extender

XML Extender proporciona tres métodos de administración: el Asistente de administración de XML Extender, el mandato de administración de XML Extender y los procedimientos almacenados de XML Extender.

- El mandato de administración, **dxxadm**, proporciona opciones para las diversas tareas de administración.
- Las tareas de administración se pueden ejecutar invocando los procedimientos almacenados para la administración desde un programa.
- El Asistente de XML Extender le guía a través de las tareas de administración. Puede utilizarlo desde una estación de trabajo cliente.

Cuando planifique una aplicación que utilice documentos XML, primero debe decidir si:

- Compondrá documentos XML a partir de datos de la base de datos.
- Almacenará documentos XML existentes. Si desea almacenar documentos XML, deberá también decidir si los desea almacenar como documentos XML intactos en una columna o bien descompuestos en datos relacionales.

Después de tomar esta decisión podrá:

- Decidir si desea validar o no los documentos XML
- Decidir si desea indexar datos de columna XML para realizar una búsqueda y recuperación rápidas
- Decidir cómo correlacionar la estructura del documento XML con tablas relacionales de DB2® UDB

Configuración del Asistente de administración

Las tareas de administración de XML Extender consisten en habilitar las columnas de la base de datos para XML y en correlacionar los datos XML con estructuras relacionales de DB2 UDB. Puede utilizar el Asistente de XML Extender para realizar estas tareas de administración. Esta sección explica cómo configurar e invocar el Asistente de administración. Puede invocar el asistente a través del menú Inicio de Windows o desde un indicador de la línea de mandatos.

Requisitos previos:

Antes de poner a punto el asistente, deberá instalarlo y configurarlo como se explica en el archivo README correspondiente a su sistema operativo. Es necesario incluir los archivos de clase necesarios en la variable de entorno CLASSPATH.

A excepción de los saltos de línea, asegúrese de que la variable de entorno CLASSPATH es similar a la del siguiente ejemplo:

```
C:\instalación_db2\java\db2java.zip;C:\instalación_db2\xml-apis.jar;
C:\instalación_db2\dxxadmin.jar;C:\instalación_db2\xerces.jar;
```

Donde *instalación_db2* es el directorio de instalación.

Procedimiento:

Para configurar el Asistente de administración de XML Extender:

1. Invoque el asistente utilizando el SDK. Puede utilizar el SDK (Software Development Kit) o bien el JRE (Java Runtime Environment).
   - Para utilizar el JRE, entre:

     ```
     jre -classpath vía-clases com.ibm.dxx.admin.Admin
     ```

   - Para utilizar el SDK, entre:

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

   Donde *vía-clases* especifica la variable de entorno %CLASSPATH% que especifica en qué lugar están situados los archivos de clases del asistente de administración. Si utiliza esta opción, la variable de sistema CLASSPATH debe apuntar a los directorios `instalación_dxx/tools` y `instalación_dxx/java`, que contienen los siguientes archivos: `dxxadmin.jar`, `xerces.jar`, `xml-apis.jar` y `db2java.zip`. Por ejemplo:

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

   classpath puede también especificar una alteración temporal de la variable de entorno %CLASSPATH% con punteros a archivos en el directorio `instalación_dxx/dxxadmin`, desde el cual se está ejecutando el Asistente de administración de XML Extender. Por ejemplo:

   ```
   java -classpath dxxadmin.jar;xml4j.jar;db2java.zip com.ibm.dxx.admin.Admin
      url=jdbc:db2:mydb2 userid=db2xml password=db2xml
      driver=COM.ibm.db2.jdbc.app.DB2Driver
   ```

2. Desde la ventana Conexión, conéctese a la base de datos que desea usar para trabajar con datos XML.
3. En el campo **Dirección**, escriba el URL JDBC calificado al completo de la fuente de datos a que se está conectando. La dirección tiene la sintaxis siguiente:

   Para configuraciones autónomas: (valores por omisión y recomendados)

   ```
   jdbc:db2:nombre_base_datos
   ```

   Donde *nombre_base_datos* es la base de datos a la que se está conectando y en la que almacena 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
El nombre del servidor donde se encuentra XML Extender.

número_puerto
Es el número de puerto que se utiliza para la conexión con el servidor. Para determinar el número de puerto, escriba el mandato siguiente en la línea de mandatos de la máquina servidor:

```
db2jstrt número_puerto
```

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

nombre_base_datos
Es la base de datos con la que se conecta y que contiene los documentos XML.

Por ejemplo:

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

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

   Para configuraciones autónomas: (valores por omisión y recomendados)

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

   Para configuraciones de red:

   ```
   COM.ibm.db2.jcc.DB2DRIVER
   ```
6. Pulse **Finalizar**. Invoque el asistente y avance hasta la ventana Área de ejecución.

Después de completar este procedimiento puede invocar el asistente en la ventana del Área de ejecución. Con el asistente, podrá realizar las siguientes funciones:

- Habilitar o inhabilitar una base de datos.
- Añadir una DTD al depósito de DTD.
- Trabajar con columnas XML.
- Trabajar con colecciones XML.

Métodos de acceso y almacenamiento

XML Extender proporciona dos métodos de acceso y almacenamiento para utilizar DB2® como depósito XML: la columna XML y la colección XML. Es necesario decidir cuál de los métodos responde mejor a las necesidades de la aplicación para acceder y manipular los datos XML.

Columna XML
Almacena y recupera documentos XML completos como datos de columna de DB2 UDB. 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 la aplicación determina qué método de acceso y almacenamiento es el más adecuado y cómo estructurar los datos XML.

Utilice el archivo DAD para asociar datos XML con tablas de DB2 UDB mediante ambos métodos de acceso y almacenamiento. La Figura 4 muestra cómo la DAD especifica los métodos de acceso y almacenamiento.

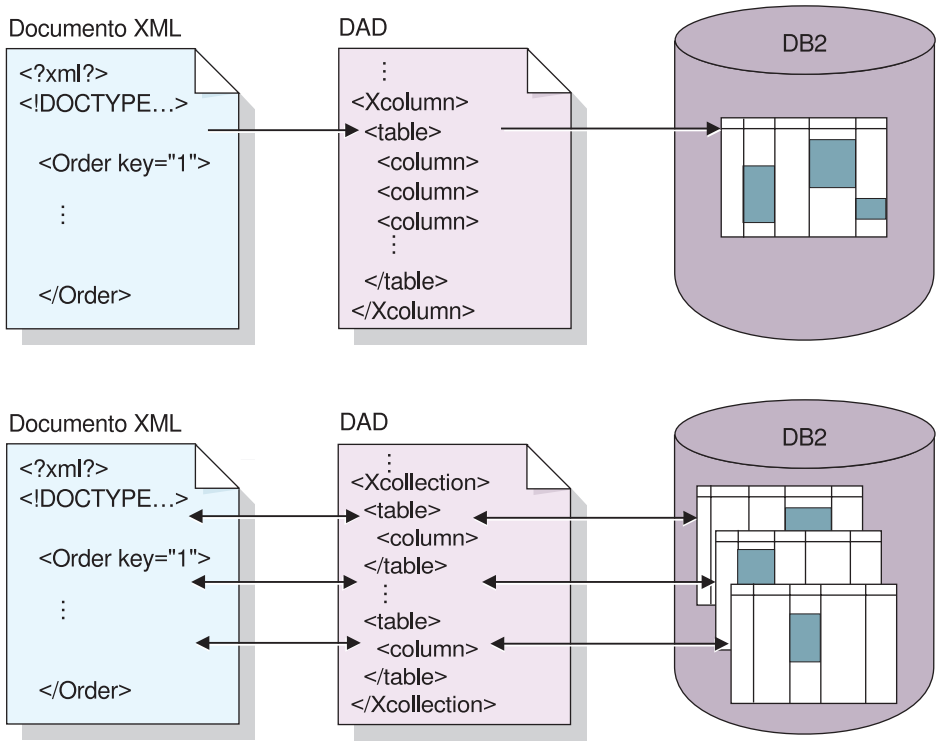


Figura 4. El archivo DAD correlaciona la estructura del documento XML con una estructura de datos relacionales de DB2 UDB y especifica el método de acceso y almacenamiento.

El archivo DAD define la ubicación de los archivos clave, como la DTD, y especifica cómo la estructura del documento XML se relaciona con los datos de DB2 UDB. Y lo que es más importante, define el método de acceso y almacenamiento que el usuario utiliza en la aplicación.

Conceptos relacionados:

- "Cuándo utilizar el método de columna XML" en la página 42
- "Cuándo utilizar el método de colección XML" en la página 43

Información relacionada:

- "Visión general de las funciones de almacenamiento en XML Extender" en la página 140

Cuándo utilizar el método de columna XML

Utilice las columnas XML en cualquiera de las siguientes situaciones:

- Los documentos XML ya existen o proceden de una fuente externa y prefiere almacenarlos en el formato XML nativo. Desea almacenarlos en DB2® para su integridad, archivado y auditoría.
- Los documentos XML se leen con frecuencia, pero no se actualizan.

- Desea utilizar tipos de datos de nombres de archivo para almacenar los documentos XML (externos a DB2 UDB) en el sistema de archivos local o remoto y utilizar DB2 UDB para las operaciones de gestión y búsqueda.
- Necesita realizar búsquedas por rango basadas en los valores de los elementos de los elementos o atributos XML y necesita conocer qué elementos o atributos se utilizan frecuentemente en los argumentos de búsqueda.
- Los documentos tienen elementos con bloques de texto de gran tamaño y desea utilizar DB2 UDB Text Extender para la búsqueda de texto estructural manteniendo el documento completo intacto.

Cuándo utilizar el método de colección XML

Utilice las colecciones XML en cualquiera de las siguientes situaciones:

- Las tablas relacionales existentes 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; además, desea tener la flexibilidad para decidir si guarda los datos en tablas existentes o en tablas nuevas.

Planificación de columnas XML

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

- El tipo definido por el usuario XML en el que va a almacenar el documento XML
- Los elementos y atributos XML que la aplicación va a buscar con frecuencia, para que su contenido se pueda almacenar en tablas auxiliares e indexar para mejorar el rendimiento
- Si desea o no validar los documentos XML en la columna con una DTD
- La estructura de las tablas auxiliares y cómo éstas se indexarán

Tipos de datos XML para las columnas XML

XML Extender proporciona tipos XML definidos por el usuario que se pueden utilizar para definir una columna para guardar documentos XML. Estos tipos de datos se describen en la Tabla 4.

Tabla 4. UDT de XML Extender

Columna de tipo definido por el usuario	Tipo de datos fuente	Descripción
XMLVARCHAR	VARCHAR(*long_varchar*)	Almacena un documento XML completo como un tipo de datos VARCHAR dentro de DB2®. Utilizado para aquellos documentos de menor tamaño (menos de 3 K) almacenados en DB2.

Tabla 4. UDT de XML Extender (continuación)

Columna de tipo definido por el usuario	Tipo de datos fuente	Descripción
XMLCLOB	CLOB(*long_clob*)	Almacena un documento XML completo en forma de tipo de datos CLOB dentro de DB2. Utilizado para documentos grandes (más de 3 K) almacenados en DB2.
XMLFILE	VARCHAR(512)	Almacena el nombre de archivo de un documento XML en DB2 y el documento XML en un archivo local para el servidor de DB2. Utilizado para documentos almacenados fuera de DB2.

Elementos y atributos a indexar para columnas XML

Cuando haya comprendido la estructura del documento XML y las necesidades de la aplicación, podrá determinar qué elementos y atributos se buscarán o extraerán con mayor frecuencia o aquellos cuya consulta supondrá un coste más elevado. El archivo DAD de una columna XML puede correlacionar las vías de ubicación de cada elemento y atributo con las tablas relacionales (tablas auxiliares) que contienen dichos objetos. A continuación, se indexan las tablas auxiliares.

Por ejemplo, la Tabla 5 muestra un ejemplo de tipos de datos y de vías de ubicación de elementos y atributos contenidos en la situación de Iniciación para columnas XML. Los datos se han especificado como información que se busca con frecuencia y las vías de ubicación apuntan a los elementos y atributos que contienen los datos. El archivo DAD puede correlacionar dichas vías de ubicación con tablas auxiliares.

Tabla 5. 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

El archivo DAD para columnas XML

Para las columnas XML el archivo DAD principalmente especifica cómo se deben indexar los documentos almacenados en una columna XML. El archivo DAD especifica una DTD a utilizar para validar documentos insertados en la columna XML. El tamaño de este archivo puede ser de hasta 2 GB.

El archivo DAD para columnas XML proporciona una correlación de cualquier dato XML que se vaya a almacenar en las tablas auxiliares para su indexado.

Para especificar el método de acceso y almacenamiento de la columna XML, utilice el código <Xcolumn> en el archivo DAD. El código <Xcolumn> especifica que los datos XML se almacenarán y recuperarán como documentos XML completos en columnas de DB2 UDB que estén habilitadas para datos XML.

Una columna habilitada para XML es de la UDT de XML Extender. Las aplicaciones pueden incluir la columna en cualquier tabla de usuario. Puede acceder a los datos de la columna XML principalmente mediante sentencias de SQL y las UDF de XML Extender.

Conceptos relacionados:

- "Planificación de tablas auxiliares" en la página 61

Planificación de colecciones XML

Cuando planifique colecciones XML, debe tener en cuenta diferentes consideraciones sobre la composición de documentos a partir de datos de DB2®, la descomposición de un documento XML en datos relacionales o ambos procesos. Las secciones siguientes tratan temas acerca de la planificación de colecciones XML y de las consideraciones que se deben tener en cuenta para la composición y la descomposición.

Validación

Una vez elegido el método de acceso y almacenamiento, puede decidir si desea validar los datos. Los datos XML se validan utilizando una DTD o un esquema. La utilización de una DTD o de un esquema para realizar la validación asegura que el documento sea válido.

Para realizar la validación utilizando una DTD, es posible que necesite tener una DTD en el depósito de XML Extender.

Importante: debe decidir si desea validar los datos XML antes de insertar los datos XML en DB2. XML Extender no valida los datos cuando ya están insertados en DB2.

Consideraciones:

- Sólo puede utilizar una DTD para la composición.
- Puede utilizar varios esquemas para la composición.
- Si decide no validar un documento, la DTD especificada por el documento XML no se procesará. Es importante procesar las DTD para resolver los valores por omisión de las entidades y de los atributos incluso cuando se procesan fragmentos de documentos que es posible validar.

El archivo DAD para colecciones XML

Para las colecciones XML, el archivo DAD correlaciona la estructura del documento XML con las tablas de DB2 UDB desde las que se compone el documento o donde se descompone el documento.

Por ejemplo, si tiene un elemento denominado <Tax> en el documento XML, deberá correlacionar <Tax> con una columna denominada TAX. Defina la relación entre los datos XML y los datos relacionales en la DAD.

Se especifica el nombre de archivo DAD cuando se habilita una colección o cuando se utiliza el archivo DAD en procedimientos almacenados de colecciones XML. Si decide validar los documentos XML utilizando una DTD, el archivo DAD se puede asociar a esa DTD. Cuando se utiliza como el parámetro de entrada de los procedimientos almacenados de XML Extender, el archivo DAD tiene un tipo de datos de CLOB. Este archivo puede tener hasta 100 KB.

Para especificar el método de acceso y almacenamiento de una colección XML, utilice el código en el archivo DAD. El código <Xcollection> especifica que los datos XML se deben descomponer desde documentos XML para crear una colección de tablas relacionales o bien componerse para crear documentos XML partiendo de una colección de tablas relacionales.

Una colección XML es un nombre virtual para un conjunto de tablas relacionales que contiene datos XML. Los programas de aplicación pueden habilitar una colección XML de tablas de usuario cualesquiera. Estas tablas de usuario pueden ser tablas o datos comerciales existentes que XML Extender ha creado recientemente.

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

root_node (nodo raíz)
Especifica el elemento raíz del documento.

element_node (nodo de elemento)
Identifica un elemento, que puede ser el elemento raíz o un elemento hijo.

text_node (nodo de texto)
Representa el texto CDATA de un elemento.

attribute_node (nodo de atributo)
Representa un atributo de un elemento.

La Figura 5 en la página 47 muestra un fragmento de la correlación que se utiliza 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
"instalación_dxx/samples/db2xml/dtd/dad.dtd">
<DAD>
  ...
<Xcollection>
<SQL_stmt>
       ...
</SQL_stmt> <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "instalación_dxx/samples/db2xml/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 correlaciona
                                          el elemento y 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 5. Definiciones de nodo en un archivo DAD para una colección XML

En el ejemplo anterior, las primeras dos columnas en la sentencia de SQL tienen elementos y atributos correlacionados con ellos.

XML Extender también da soporte a las instrucciones de proceso para hojas de estilo, utilizando el elemento <stylesheet>. El elemento <stylesheet> debe encontrarse en el interior 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 Websphere Studio Application Developer para crear y actualizar el archivo DAD. El elemento <stylesheet> no está soportado en la actualidad por el Asistente de administración de XML Extender.

Esquemas de correlación para colecciones XML

Si está utilizando una colección XML, debe seleccionar un *esquema de correlación* que defina cómo se representan los datos XML en una base de datos relacional. Debido a que las colecciones XML deben correlacionar la estructura jerárquica

utilizada en documentos XML con una estructura relacional, es necesario entender cómo se comparan ambas estructuras. La Figura 6 muestra cómo la estructura jerárquica se puede correlacionar con columnas de una tabla relacional.

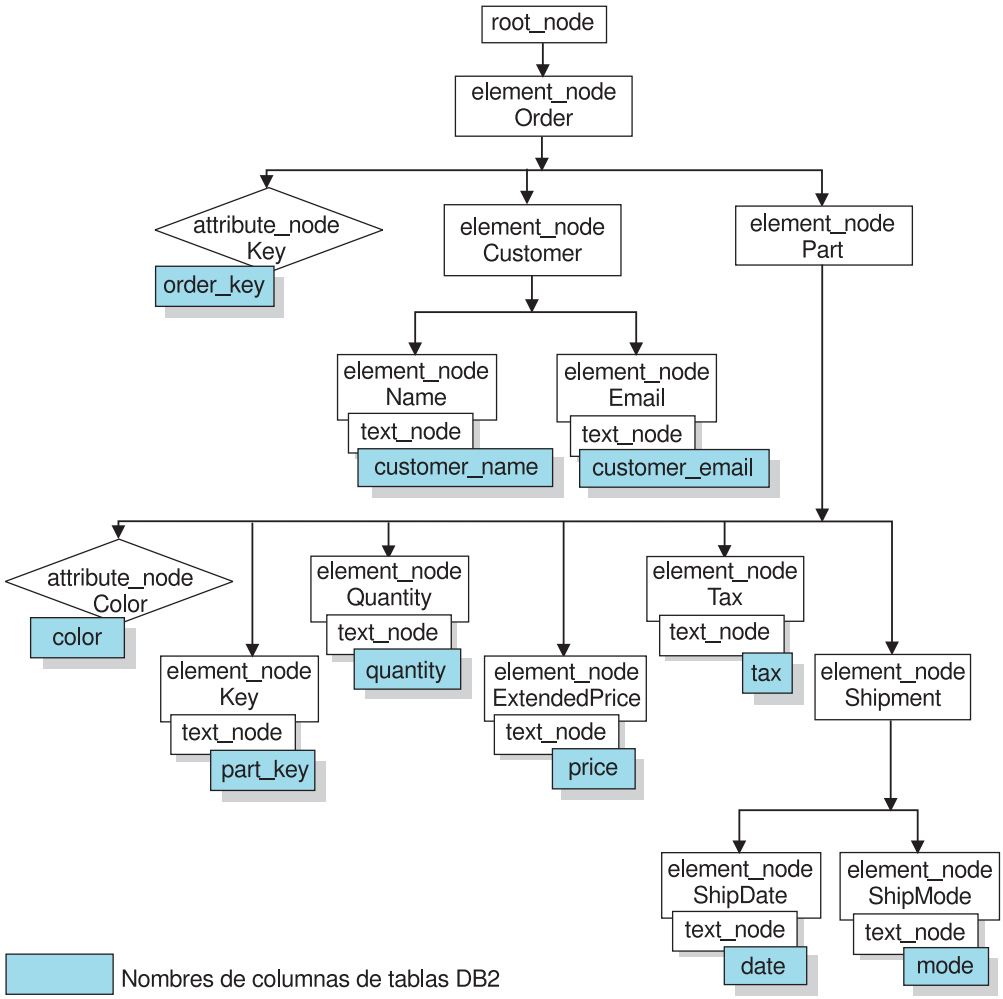


Figura 6. Estructura de un documento XML correlacionado con columnas de tabla relacional

XML Extender utiliza el esquema de correlación al componer y descomponer documentos XML utilizando datos relacionales que se encuentran en varias tablas relacionales. XML Extender proporciona un asistente que ayuda a crear el archivo DAD. No obstante, antes de crear el archivo DAD, debe plantearse cómo se correlacionan los datos XML con la colección XML.

Tipos de esquemas de correlación

El esquema de correlación se especifica en el elemento <Xcollection> en el archivo DAD. XML Extender proporciona dos tipos de esquemas de correlación: *correlación de SQL* y *correlación de base de datos relacional (RDB_node)*.

Correlación de SQL

Permite la correlación directa desde datos relacionales a documentos XML mediante una única sentencia de SQL. La correlación de SQL se utiliza para la composición; no se utiliza para la descomposición. La correlación de SQL se define mediante el elemento SQL_stmt en el archivo DAD. El contenido del elemento SQL_stmt es una sentencia de SQL válida. El elemento SQL_stmt correlaciona la columna en la cláusula SELECT con

elementos o atributos XML utilizados en el documento XML. Los nombres de columna en la cláusula SELECT de la sentencia de SQL se utilizan para definir el valor de un attribute_node o el contenido de un text_node. La cláusula FROM define las tablas que contienen los datos; la cláusula WHERE especifica la condición de unión y búsqueda.

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

Correlación de RDB_node

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

Este método utiliza el *RDB_node* proporcionado por XML Extender, que contiene una o más definiciones de nodo para tablas, columnas opcionales y condiciones opcionales. Las tablas y columnas se utilizan para definir cómo los datos XML deben almacenarse 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 un archivo DAD con un elemento <Xcollection>. La Figura 7 en la página 50 muestra un fragmento de un archivo DAD de ejemplo con una correlación de SQL para una colección XML que compone un conjunto de documentos XML a partir de datos de tres tablas relacionales.

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

               as ship_id, date, node, from ship_tab)  shipid
        WHERE p.price > 2500.00 and s.date > "1996-06-01" AND
              p.order_key = o.order_key and s.part_key = p.part_key
</SQL_stmt> <prolog>?xml version="1.0"?</prolog>
    <doctype>!DOCTYPE DAD SYSTEM "instalación_dxx
     /samples/db2xml/dtd/getstart.dtd"</doctype>
<root_node>
    <element_node name="Order">
    <attribute_node name="key">
   <column_name="order_key"/>
    </attribute_node>
    <element_node name="Customer">
     <text_node>
      <column name="customer"/>
     </text_node>
    <element_node>

  ...

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

Figura 7. Esquema de correlación de SQL

XML Extender proporciona varios procedimientos almacenados que gestionan datos en una colección XML. Estos procedimientos almacenados dan soporte a ambos tipos de correlación, pero requieren que el archivo DAD siga las normas que se describen en el apartado “Requisitos del esquema de correlación”.

Requisitos del esquema de correlación

Las secciones siguientes describen los requisitos para cada tipo de los esquemas de correlación para la colección XML.

Requisitos de esquema de correlación para la correlación de SQL

En este esquema de correlación, debe especificar el elemento SQL_stmt en el elemento <Xcollection> de la DAD. SQL_stmt debe contener una única sentencia de SQL que pueda unir varias tablas relacionales con el predicado de 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 alias para una de las columnas.
  - Agrupe las columnas de la misma tabla y utilice el nivel jerárquico lógico de las tablas relacionales. Es decir, agrupe las columnas de

acuerdo con la manera en que las tablas se correlacionan con la estructura jerárquica del documento XML. En la cláusula SELECT, las columnas de las tablas de nivel superior deben preceder a las columnas de 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 porque están más altas en la estructura de árbol del documento XML. Las columnas `ship_id`, `date` y `mode` de la tabla SHIP_TAB están en el nivel relacional más bajo.

– El algoritmo de composición utiliza la información clave del candidato como pista para ayudarle con el proceso. Utilice una clave adecuada de columna individual para comenzar cada nivel de tabla en la cláusula SELECT. Si dicha clave no se encuentra disponible en una tabla, la consulta debe generar una para esa tabla utilizando una expresión de tabla y la función incorporada, generate_unique(). Las claves generadas no necesitan aparecer en el documento de salida, pero sí necesitan estar en la cláusula SELECT. 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, debe generarse una, en este caso `ship_id`. La clave primaria aparece listada como la primera columna del grupo de 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 incorporada, generate_unique(), para generar una sola clave para tablas que no tengan una sola clave 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, se genera una clave adecuada de columna individual con la función generate_unique() y se le asigna un alias denominado `ship_id`.

  – Utilice un nombre de alias cuando sea necesario diferenciar una columna. Por ejemplo, podría utilizar `o` para ORDER_TAB, `p` para PART_TAB y `s` para SHIP_TAB.

- **Cláusula WHERE**
  – Especifique la condición de unión que asocia entre sí las tablas en la colección. Por ejemplo:

```
WHERE p.price > 2500.00 AND s.date > '2003-06-01' AND
      p.order_key = o.order_key AND s.part_key = p.part_key
```

  – 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 del elemento SQL_stmt.
  – Asegúrese de que los nombres de columna coincidan con la cláusula SELECT.

- Liste las claves adecuadas de una columna en el orden del nivel jerárquico de sus tablas correspondientes.
- 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. La conservación del orden asegura que no haya duplicados incorrectos en los documentos XML que se crearán.
- No califique las columnas en la cláusula ORDER BY con un nombre de esquema o de tabla.

Aunque el elemento SQL_stmt tenga los requisitos anteriores, es potente porque puede especificar cualquier predicado en la cláusula WHERE si la expresión del predicado utiliza las columnas en las tablas.

Requisitos de esquema de correlación para la correlación RDB_node

Cuando utilice el método de correlación, no utilice el elemento SQL_stmt en el elemento <Xcollection> del archivo DAD. En su lugar, utilice el elemento RDB_node como hijo del element_node superior y para cada attribute_node y text_node.

- **RDB_node para el element_node superior**

  El element_node superior del archivo DAD representa el elemento raíz del documento XML. Especifique un RDB_node para el element_node superior basándose en estos requisitos:

  - Los caracteres de final de línea están permitidos en las sentencias condicionales.
  - Los elementos de condición pueden hacer referencia a un nombre de columna un número ilimitado de veces.
  - Especifique todas las tablas que están asociadas con los documentos XML. Por ejemplo, la correlación siguiente especifica tres tablas en el RDB_node del element_node <Order>, que es el element_node 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>
```

    No existen restricciones de clasificación respecto a los predicados de la condición de nodo raíz. 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, especifique una clave primaria para cada tabla. La clave primaria consta de una o varias columnas; en este segundo caso se denomina clave compuesta. La clave primaria se especifica añadiendo una clave de atributo al elemento de tabla del RDB_node. Cuando proporcione una clave compuesta, el atributo de clave se especificará mediante los nombres de las columnas de clave separadas por un espacio. Por ejemplo:

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

La información especificada para la descomposición se ignorará al componer un documento.

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

- **RDB_node para cada attribute_node y text_node**

  Necesita especificar un RDB_node para cada attribute_node y text_node, que indiquen al procedimiento almacenado de qué tabla, qué columna y bajo qué condición de consulta obtener los datos. Debe especificar los valores de tabla y de columna; el valor de condición es opcional.

  – Especifique el nombre de la tabla que contiene los datos de columna. El nombre de tabla debe incluirse en el RDB_node del element_node superior. En este ejemplo, para el text_node del elemento <Price>, la tabla se especifica 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 que los documentos XML se generen utilizando la condición de consulta. La sintaxis permitida para <condition> es la siguiente:

    - nombrecolumna
    - operador
    - literal

    En el ejemplo anterior, la condición se especifica como `price > 2500.00`. Sólo los datos que cumplan la condición estarán en los documentos generados. La condición debe ser una cláusula WHERE válida.

  – Si está descomponiendo un documento o habilitando la colección XML especificada en el archivo DAD, especifique el tipo de columna de cada attribute_node y text_node. 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 se ignorará al componer un documento.

Con el método de correlación de RDB_node, no es necesario suministrar sentencias de SQL. Sin embargo, establecer condiciones de consulta complejas en el elemento de RDB_node puede ser más difícil.

Requisitos de tamaño de tabla de descomposición para la correlación RDB_node

La descomposición utiliza la correlación RDB_node para especificar cómo se descompone un documento XML en tablas de DB2 UDB extrayendo los valores de elemento y atributo en filas de tabla. Los valores de cada documento XML se almacenan en una o más tablas de DB2. Cada tabla puede tener un máximo de 10240 filas descompuestas de cada documento. Por ejemplo, si se descompone un documento XML en cinco tablas, cada una de las cinco tablas puede tener hasta 10240 filas para dicho documento.

La utilización de elementos de aparición múltiple (elementos con vías de ubicación que pueden aparecen más de una vez en la estructura XML) afecta al número de filas insertadas para cada documento. Por ejemplo, un documento que contiene un elemento <Part> que aparece 20 veces, puede descomponerse como 20 filas en una tabla. Al utilizar elementos de aparición múltiple, tenga en cuenta que pueden descomponerse un máximo de 10240 filas en una tabla partiendo de un único documento.

Conceptos relacionados:

• "Archivos DAD para colecciones XML" en la página 171

Tareas relacionadas:

• "Almacenamiento de una DTD en la tabla de depósito" en la página 56

Validación automática de documentos XML

Después de elegir un método de acceso y almacenamiento, ya sea Columna XML o Colección XML, podrá determinar si *validar* los documentos XML. También puede validar los documentos XML compuestos a partir de las colecciones XML.

Puede hacer que los datos XML se validen automáticamente especificando YES para la validación en el archivo DAD. Para hacer que un documento se valide al almacenarlo en DB2®, deberá especificar una DTD dentro del elemento `<dtdid>` o en la especificación `<!DOCTYPE>` en el documento original. Para hacer que un documento se valide cuando está compuesto a partir de una colección XML en DB2, deberá especificar una DTD en el elemento `<dtdid>` o en el elemento `<doctype>` del archivo DAD.

Deben tenerse en cuenta los siguientes factores al decidir si validar los documentos.

- El DTDID o esquema sólo es válido si decide validar el documento XML.

  Para validar la DAD con un esquema, inserte los códigos de esquema que asocian el archivo DAD con el archivo de esquema. Por ejemplo:

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

- No es necesaria una DTD para almacenar ni archivar documentos XML.
- Tanto si decide validar como si no, puede que sea necesario procesar la DTD para establecer valores de entidad y valores por omisión de atributos.
- Si especifica NO para la validación en la DAD, la DTD especificada por el documento XML no se procesará.
- La validación de los datos XML tiene efecto sobre el rendimiento.

Habilitación de bases de datos para XML

Antes de poder almacenar o recuperar documentos XML desde DB2 UDB con XML Extender, debe habilitar la base de datos para XML. XML Extender habilita la base de datos a la que esté conectado utilizando la instancia actual.

Al habilitar una base de datos para XML, XML Extender realiza las siguientes tareas:

- Crea todos los tipos definidos por el usuario (UDT), las funciones definidas por el usuario (UDF) y los procedimientos almacenados para XML Extender
- Crea y llena las tablas de control con los metadatos necesarios para XML Extender
- Crea el esquema DB2XML en los espacios de tabla definidos por el usuario y asigna los privilegios necesarios

  El nombre calificado al completo de una función XML es db2xml.*nombre-función*, donde db2xml es un identificador que proporciona una agrupación lógica para objetos SQL. También puede utilizar el nombre calificado al completo en cualquier lugar en que haga referencia a una UDF o a un UDT. Puede asimismo omitir el nombre de esquema al hacer referencia a una UDF o a un UDT; en este caso DB2 UDB utilice la vía de acceso de la función para determinar el tipo de función o de datos.

Procedimiento:

Puede habilitar una base de datos mediante el Asistente de administración o desde una línea de mandatos. Para realizar esta tarea, desde la línea de mandatos, escriba **dxxadm** y especifique la base de datos a habilitar.

El siguiente ejemplo habilita una base de datos llamada SALES_DB.

```
dxxadm enable_db SALES_DB
```

Para habilitar una base de datos utilizando el Asistente de administración, es necesario realizar las siguientes tareas:

1. Inicie el Asistente de administración y pulse **Habilitar base de datos** desde la ventana Área de ejecución.

   Si una base de datos ya se ha habilitado, aparecerá el botón **Inhabilitar base de datos**. Si la base de datos no se ha habilitado aún, aparecerá el botón **Habilitar base de datos**.

   Cuando se habilite la base de datos, volverá a la ventana Área de ejecución.

Después de habilitar una base de datos, puede utilizar los UDT, las UDF y los procedimientos almacenados de XML Extender.

Conceptos relacionados:

- "Migración de XML Extender desde versiones anteriores" en la página 38

Creación de una tabla XML

Esta tarea forma parte de la tarea más amplia de definición y habilitación de una columna XML.

Se utiliza una tabla XML para almacenar documentos XML intactos. Para almacenar documentos completos en la base de datos con DB2 UDB XML Extender, deberá crear una tabla que contenga una columna con un tipo definido por el usuario XML (UDT). DB2 UDB XML Extender proporciona tres tipos definidos por el usuario para almacenar los documentos XML como datos de columna. Estos UDT son: XMLVARCHAR, XMLCLOB y XMLFILE. Cuando una tabla contiene una columna de tipo XML, podrá habilitarla para XML.

Puede crear una nueva tabla para añadir una columna de tipo XML utilizando el asistente de administración o la línea de mandatos.

Procedimiento:

Para crear una tabla con una columna de tipo XML mediante la línea de mandatos:

Abra el indicador de mandatos de DB2 y escriba una sentencia de creación de tabla.

Por ejemplo, en una aplicación de ventas, tal vez desee guardar un pedido de unidades por líneas de formato XML en una columna llamada ORDER de una tabla denominada SALES_TAB. Esta tabla también tiene las columnas INVOICE_NUM y SALES_PERSON. Como se trata de un pedido pequeño, almacene el pedido de ventas utilizando el tipo XMLVARCHAR. La clave primaria es INVOICE_NUM. La siguiente sentencia CREATE TABLE crea una 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);
```

Cuando haya creado una tabla, el siguiente paso es habilitar la columna para datos XML.

Conceptos relacionados:

- "Planificación de tablas auxiliares" en la página 61
- Capítulo 13, "Tablas de soporte de administración de XML Extender", en la página 279

Almacenamiento de una DTD en la tabla de depósito

Puede utilizar una DTD para validar datos XML contenidos en una columna XML o una colección XML. Las DTD se pueden almacenar en la tabla de depósito de DTD, una tabla de DB2 UDB llamada DTD_REF. La tabla DTD_REF tiene un nombre de esquema de DB2XML. Cada DTD de la tabla DTD_REF tiene un ID exclusivo. XML Extender crea la tabla DTD_REF cuando se habilita una base de datos para XML. Puede insertar la DTD desde la línea de mandatos o utilizando el Asistente de administración.

Procedimiento:

Para insertar la DTD utilizando el Asistente de administración:

1. Inicie el Asistente de administración y pulse **Importar una DTD** desde la ventana Área de ejecución para importar un archivo DTD existente en el depósito de DTD de la base de datos actual. Se abrirá la ventana Importar una DTD.

2. Especifique el nombre de archivo DTD en el campo **nombre de archivo DTD**.
3. Escriba el DTDID en el campo **ID de DTD**.

   El ID de DTD es un identificador de la DTD. También puede ser la vía de acceso que especifica la ubicación de una DTD en el sistema local. El ID de DTD debe coincidir con el valor especificado en el archivo DAD para el elemento <DTDID>.
4. Opcional: Escriba el nombre del autor de la DTD en el campo **Autor**.
5. Pulse **Finalizar** para insertar la DTD en la tabla de depósito de DTD, DB2XML.DTD_REF y para volver a la ventana Área de ejecución.

Para insertar una DTD desde la línea de mandato, emita una sentencia INSERT de SQL desde la Tabla 6. Por ejemplo:

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

Tabla 6. Las definiciones de columna de la tabla de depósito de DTD

Nombre de columna	Tipo de datos	Descripción
DTDID	VARCHAR(128)	ID de la DTD.
CONTENT	XMLCLOB	Contenido de la DTD.
USAGE_COUNT	INTEGER	Número de columnas XML y de colecciones XML en la base de datos que utilizan esta DTD.
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.
UPDATOR	VARCHAR(128)	ID de usuario que realiza la última actualización.

Habilitación de columnas XML

Para almacenar un documento XML en una base de datos DB2 UDB, es necesario habilitar la columna que contendrá el documento para XML. La habilitación de una columna permite indexarla para poder efectuar rápidamente búsquedas en ella. Puede habilitar una columna utilizando el Asistente de administración de XML Extender o la línea de mandatos. La columna debe ser de tipo XML.

Cuando XML Extender habilita una columna XML, realiza las siguientes operaciones:

- Lee el archivo DAD para:
  - Comprobar la existencia de la DTD en la tabla DTD_REF, si se ha especificado el DTDID.
  - Crear tablas auxiliares en la columna XML, para la indexación.
  - Preparar la columna para que contenga datos XML.
- Opcionalmente crea una vista por omisión de la tabla XML y de las tablas auxiliares. La vista por omisión muestra la tabla de la aplicación y las tablas auxiliares.
- Especifica una columna ID RAÍZ, si no se ha especificado ninguna.

Cuando haya habilitado la columna XML, podrá:

- Crear índices en las columnas auxiliares.
- Insertar documentos XML en la columna XML.
- Consultar, actualizar o buscar los documentos XML de la columna XML.

Puede inhabilitar las columnas XML utilizando el Asistente de administración o desde una línea de mandatos de DB2.

Procedimiento (utilizando el Asistente de administración):

Para habilitar columnas XML utilizando el Asistente de administración:

1. Configure e inicie el Asistente de administración.
2. Pulse **Trabajar con columnas XML** desde la ventana Área de ejecución para ver las tareas relacionadas con las columnas de XML Extender. Se abrirá la ventana Seleccionar tarea.
3. Pulse **Habilitar una columna** y, a continuación, **Siguiente**.
4. Especifique la tabla y la columna.
   - En el campo **Nombre de tabla**, seleccione la tabla donde reside la columna XML.
   - Seleccione la columna que desea habilitar en el campo **Nombre de columna**.
5. Especifique la vía de acceso y el nombre de archivo DAD en el campo **Nombre de archivo DAD**. Por ejemplo:

   ```
   instalación_dxx/samples/dad/getstart.dad
   ```
6. Opcional: Escriba el nombre de un espacio de tabla existente en el campo **Espacio de tabla**.

   El espacio de tabla por omisión contiene tablas auxiliares creadas por XML Extender. Si especifica un espacio de tabla, las tablas auxiliares se crean en el espacio de tabla especificado. Si no especifica ningún espacio de tabla, las tablas auxiliares se crean en el espacio de tabla por omisión.
7. Opcional: Escriba el nombre de la vista por omisión en el campo **Vista por omisión**.

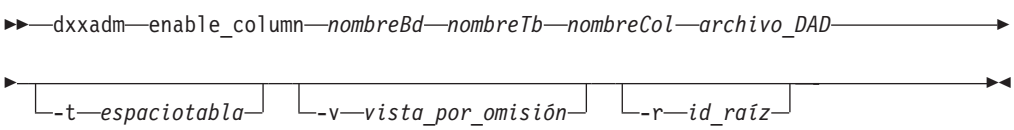
   Si se especifica, la vista por omisión se crea automáticamente al crear la columna. La vista por omisión une la tabla XML y todas las tablas auxiliares relacionadas.
8. Recomendado: Escriba el nombre de la columna correspondiente a la clave primaria de la tabla en el campo **ID raíz**.

   XML Extender utiliza el valor de **ID raíz** como identificador único para asociar todas las tablas auxiliares con la tabla de aplicación. XML Extender añade la columna DXXROOT_ID a la tabla de aplicación y genera un identificador.
9. 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, recibirá el mensaje: `Columna habilitada`.
   - Si la columna no se ha habilita satisfactoriamente, aparecerá un mensaje de error junto con indicaciones para que corrija los valores de los campos de entrada hasta que la columna esté habilitada satisfactoriamente.

Procedimiento (utilizando la línea de mandatos):

Para habilitar una columna XML utilizando la línea de mandatos, utilice el mandato DXXADM enable_column.

Sintaxis:



Parámetros:

nombreBd
Es el nombre de la base de datos.

nombreTb
Es el nombre de la tabla que contiene 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).

vista_por_omisión
Opcional. El nombre de la vista por omisión creada por XML Extender para unir una tabla de aplicaciones y todas las tablas auxiliares relacionadas.

id_raíz Opcional, pero recomendado. 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. DB2 XML Extender utiliza el valor de ROOT_ID como un identificador exclusivo para asociar todas las tablas auxiliares con la tabla de aplicaciones. Si no se especifica el ID RAÍZ, XML Extender añade la columna DXXROOT_ID a la tabla de aplicación y genera un identificador.

Restricción: Si la tabla de aplicación contiene un nombre de columna de DXXROOT_ID, debe especificar el parámetro *id_raíz*; de lo contrario, se producirá un error.

Ejemplo:

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

En este ejemplo, la columna ORDER se habilita en la tabla SALES_TAB. El archivo DAD es `getstart.dad`, la vista por omisión es sales_order_view, y el ID RAÍZ es INVOICE_NUMBER.

De acuerdo con este ejemplo, la tabla SALES_TAB presenta las columnas siguientes:

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

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

ORDER_SIDE_TAB:

Nombre de columna	Tipo de datos	Expresión de vía
ORDER_KEY	INTEGER	/Order/@Key
CUSTOMER	VARCHAR(50)	/Order /Customer /Name
INVOICE_NUM	CHAR(6)	N/D

PART_SIDE_TAB:

Nombre de columna	Tipo de datos	Expresión de vía
PART_KEY	INTEGER	/Order/Part/@Key
PRICE	DOUBLE	/Order/Part /ExtendedPrice
INVOICE_NUM	CHAR(6)	N/D

SHIP_SIDE_TAB:

Nombre de columna	Tipo de datos	Expresión de vía
DATE	DATE	/Order/Part/ Shipment/ShipDate
INVOICE_NUM	CHAR(6)	N/D

Todas las tablas auxiliares tienen la columna INVOICE_NUM del mismo tipo, ya que la clave primaria INVOICE_NUM especifica el ID RAÍZ en la tabla de aplicación. Después de habilitar la columna, el valor de la columna INVOICE_NUM se inserta en las tablas auxiliares cuando se inserta una fila en la tabla principal. Si especifica una vista por omisión al habilitar la columna XML ORDER, XML Extender crea una vista por omisión, sales_order_view. La vista une las tablas anteriores mediante 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_side_tab.order_key, order_side_tab.customer,
            part_side_tab.part_key, part_side_tab.price,
            ship_tab.date
     FROM sales_tab, order_side_tab, part_side_tab, ship_side_tab
     WHERE sales_tab.invoice_num = order_side_tab.invoice_num
               AND sales_tab.invoice_num = part_side_tab.invoice_num
               AND sales_tab.invoice_num = ship_side_tab.invoice_num
```

Si especifica el espacio de tabla al realizar la habilitación, se crearán las tablas auxiliares en el espacio de tabla especificado. Si no se especifica el espacio de tabla, las tablas auxiliares se crean en el espacio de tabla por omisión.

Planificación de tablas auxiliares

Las tablas auxiliares son tablas de DB2® que se utilizan para extraer el contenido de un documento XML en el que se buscará con frecuencia. La columna XML está asociada a las tablas auxiliares que guardan el contenido del documento XML. Cuando se actualiza el documento XML en la tabla de aplicación, los valores de las tablas auxiliares se actualizan automáticamente.

La Figura 8 muestra una columna XML con tablas auxiliares.

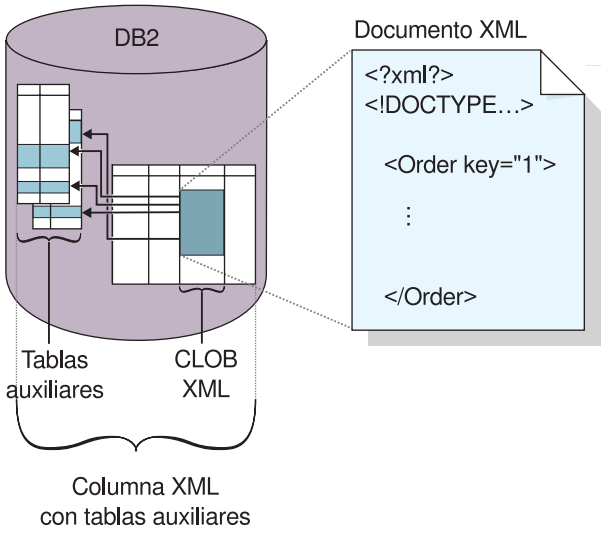


Figura 8. Columna XML cuyo contenido está correlacionado en las tablas auxiliares. Existe un archivo XML en la columna que está asociada a las tablas auxiliares que guardan el contenido del documento XML.

Aparición múltiple:

Cuando los elementos y atributos aparecen varias veces en las tablas auxiliares, tenga en cuenta las cuestiones siguientes en la planificación:

- Para los elementos o atributos de un documento XML de aparición múltiple, debe crear una tabla auxiliar separada para cada elemento o atributo XML con varias apariciones, debido a la estructura completa de los documentos XML. Esto significa que los elementos o atributos tienen vías de ubicación que aparecen varias veces y deben correlacionarse con una tabla con una sola columna. No pueden haber otras columnas en la tabla.
- Cuando un documento tiene varias vías de ubicación de aparición múltiple, XML Extender añade una columna llamada DXX_SEQNO con un tipo INTEGER en cada tabla auxiliar para efectuar el seguimiento del orden de los elementos que aparecen más de una vez. Con DXX_SEQNO, puede recuperar una lista de los elementos en 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, la consulta puede ser parecida a la siguiente:

```
SELECT sales_person FROM sales_order_view
   WHERE price > 2500,00
```

La sentencia de SQL devuelve los nombres de los vendedores de SALES_TAB que tienen pedidos guardados en la columna ORDER y donde la columna 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 de las tablas auxiliares. Sin embargo, cuántas más tablas auxiliares se creen, más costosa será la consulta. Por lo tanto, sólo es recomendable crear la vista por omisión cuando el número total de columnas de tablas auxiliar sea pequeño. Las aplicaciones pueden crear sus propias tablas, uniendo las columnas de tablas auxiliares importantes.

Indexación de tablas auxiliares

Esta tarea forma parte de la tarea más amplia de definición y habilitación de una columna XML.

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

Procedimiento:

Para crear un índice de las tablas auxiliares desde la línea de mandatos de DB2 UDB, utilice la sentencia DB2 CREATE INDEX SQL

desde la línea de mandatos de DB2 UDB.

En el ejemplo siguiente, se crean índices en cuatro tablas auxiliares utilizando el indicador 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)
```

Composición de documentos XML utilizando la correlación de SQL

Puede componer documentos XML utilizando la correlación SQL desde la línea de mandatos o utilizando el Asistente de administración.

Debe utilizar la correlación SQL si está componiendo un documento XML y desea utilizar una sentencia de SQL para definir la tabla y las columnas de las cuales procederán los datos del documento XML. Sólo puede utilizar la correlación SQL para componer documentos XML. Un archivo DAD se crea para componer el documento XML mediante la correlación SQL.

Requisitos previos:

Antes de componer los documentos, primero deberá correlacionar la relación entre las tablas de DB2 UDB y el documento XML. Este paso incluye correlacionar la jerarquía del documento XML y especificar cómo se correlacionan los datos en el documento con una tabla de DB2 UDB.

Procedimiento:

Para componer documentos XML desde la línea de mandatos, siga los siguientes pasos:

1. Cree un documento nuevo en un editor de texto y escriba la sintaxis siguiente:
   ```
   <?XML version="1.0"?>
   ```
2. Inserte los códigos <DAD></DAD>.

   El elemento DAD contendrá los demás elementos.
3. Inserte los códigos utilizados para validar la DAD con una DTD o con un esquema.
   - Para validar el documento XML compuesto con una DTD, inserte los códigos DTDID para asociar el archivo DAD con la DTD del documento. Por ejemplo:
     ```
     <dtdid>vía_acceso/nombre_dtd.dtd>
     ```
   - Para validar el documento XML compuesto con un esquema, inserte los códigos de esquema que asocian el archivo DAD con el archivo de esquema. Por ejemplo:
     ```
     <schemabindings>
     <nonamespacelocation location="vía_acceso/schema_name.xsd"/>
     </schemabindings>
     ```

   La DTD o el esquema sólo son útiles si decide validar el documento XML. Utilice el código de validación para indicar si DB2 UDB XML Extender valida el documento XML:
   - Si desea validar el documento XML, escriba:
     ```
     <validation>YES</validation>
     ```
   - Si no desea validar el tipo de documento XML:
     ```
     <validation>NO</validation>
     ```
4. Escriba los códigos <XCollection> </XCollection> para especificar que está utilizando colecciones XML como método de acceso y almacenamiento para los datos XML.
5. Dentro de los códigos <Xcollection> </Xcollection>, inserte los códigos <SQL_stmt> </SQL_stmt> para especificar la sentencia de SQL que correlacionará los datos relacionales con los documentos XML. Esta sentencia se utiliza para consultar datos de tablas de DB2 UDB. El siguiente ejemplo muestra un ejemplo de consulta SQL:
   ```
   <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>
   ```

La sentencia de SQL de ejemplo para el correlacionado de los datos relacionales con el documento XML cumple los siguientes requisitos:

- Las columnas se especifican en orden de arriba abajo, según la jerarquía de la estructura del documento XML.
- Las columnas de una entidad se agrupan conjuntamente.
- La columna de ID de objeto es la primera columna de cada grupo.
- La tabla Order_tab no tiene una columna de clave única y, por lo tanto, se utiliza la función incorporada de DB2 UDB generate_unique para generar la columna ship_id.
- En la cláusula ORDER BY, las columnas de ID de objeto se listan en orden correspondiente a la jerarquía de arriba abajo a la que pertenecen. La columna en ORDER BY no debe ser calificada por ningún esquema y los nombres de columnas deben coincidir con los nombres de columnas en la cláusula SELECT.

6. Añada la siguiente información de prólogo (prolog) que debe utilizarse en el documento XML compuesto:

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

7. Escriba el código <doctype> </doctype>. Este código contiene la declaración DOCTYPE que hay que insertar en cada documento compuesto. Por ejemplo:

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

8. Especifique el elemento raíz y los elementos y atributos que componen el documento XML:

   a. Añada el código <root></root_node> para definir el elemento raíz. Todos los elementos y atributos que forman el documento XML se especifican dentro del nodo raíz (root_node).

   b. Utilice los códigos `<element_node>`, `<attribute_node>`, `<text_node>` y `<column>` para especificar los nombres de los elementos y atributos en el documento compuesto y para correlacionarlos con las columnas especificadas en la sentencia de SQL.

      Código <column>
      Especifica la columna desde la que se deben recuperar los datos para el valor del elemento o atributo.

      Código <element_node>
      Especifica los elementos en el documento XML. Establece el atributo de nombre del código element_node en el nombre del atributo del elemento. Cada element_node puede tener varios element_node hijos.

      Código <attribute_node>
      Especifica los atributos de un elemento en el documento XML. Los atributos se anidan en el nodo de elemento. Establece el atributo de nombre del código attribute_node en el nombre del atributo.

      Código <text_node>
      Especifica el contenido del texto del elemento y los datos de las columnas de una tabla relacional para nodos de elemento (element_node) de nivel inferior. Para cada elemento de nivel inferior, los códigos <text_node> indican que el elemento contiene datos de caracteres a extraer de DB2 cuando se ha compuesto el documento. Para cada element_node de nivel inferior, utilice un código <column> para especificar desde qué columna extraer los datos cuando se compone el documento XML. Los códigos de

columnas se encuentran normalmente dentro de los códigos <attribute_node> o <text_node>. Todos los nombres de columna definidos deben estar en la cláusula <SQL_stmt> SELECT al principio del archivo DAD.

9. Asegúrese de que los códigos de finalización se encuentran en sus lugares correspondientes:
   a. Asegúrese de que un código de finalización </root_node> se encuentra después del último </element_node>.
   b. Asegúrese de que un código de finalización </Xcollection> se encuentra después del código </root_node>.
   c. Asegúrese de que un código de finalización </DAD> se encuentra después del código </Xcollection>.
10. Guarde el archivo como *archivo.dad*. Donde *archivo* es el nombre del archivo.

El siguiente ejemplo de Windows muestra una DAD completa:

```
<?xml version'"1.0"?>
<!DOCTYPE DAD SYSTEM "C:\dxx_xml\test\dtd\dad.dtd">
<DAD>
<validation>NO</validation>   <Xcollection>
<SQL_stmt> select o.order_key, customer_name, customer_email,
 p.part_key, color, qty, price, tax, ship_id, date, mode from order_tab o,
 part_tab p, (select db2xml.generate_unique() as
ship_id, date, mode, part_key from ship_tab) s where
 o.order_key = 1 and p.price . 20000 and p.order_key
 = o.order_key and s.part_key =p.part_key ORDER BY order_key,
part_key, ship_id</SQL_stmt>
<prolog>?XML version="1.0"?</prolog>
<doctype>!DOCTYPE ORDER SYSTEM
"C:\instalación_dxx\samples\db2xml\dtd/Order.dtd"
 </doctype>
   <root_node>
 <element_node name="Order">
  <attribute_node name="key">
   <column name="order_key"/>
     </attribute_node>
   <element_node name="Customer">
    <element_node name="NAME">
     <text_node><column name="customer_name"/></text_node>
        </element_node>
        </element_node>
    <element_node name="Part">
     <attribute_node name="color">
      <column name="color"/>
     </attribute_node>
     <element_node name="key">
      <text_node><column name="part_key"/></text_node>
        </element_node>
     <element_node name ="Quantity">
      <text_node><column name="qty"/></text_node>
        </element_node>
     <element_node name="ExtendedPrice">
      <text_node><column name="price"/></text_node>
        </element_node>
     <element_node name="Tax">
      <text_node><column name="tax"/></text_node>
        </element_node>
     <element_node name="Shipment" multi_occurrence="YES">
       <element_node name=shipDate">
        <text_node><column name="date"/><text_node>
         <element_node>
        <element_node name="ShipMode">
         <text_node><column name="mode"/></text_node>
```

```
    </element_node>
    </element_node>
    </element_node>
    </element_node>
</root_node>
</Xcollection>
</DAD>
```

Composición de colecciones XML utilizando la correlación de RDB_node

La correlación de RDB_node utiliza los códigos <RDB_node> para especificar las tablas, columnas y condiciones de un nodo de elemento o atributo. Utilice este método si desea componer documentos XML utilizando una estructura de tipo XML. <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 de la columna.

Los elementos hijo que se utilizan en el elemento RDB_node dependen del contexto del nodo y utilizan las siguientes normas:

Si el tipo de nodo es:	**Se utilizan los siguientes elementos hijo de RDB:**		
	Tabla	**Columna**	**Condición**
Elemento raíz	Sí	No	Sí[1]
Atributo	Sí	Sí	Opcional
Texto	Sí	Sí	Opcional

[1] Es obligatorio con varias tablas

Puede utilizar el Asistente de administración o una línea de mandatos para componer documentos XML utilizando la correlación de RDB_node.

Restricciones:

Si compone las colecciones XML utilizando la correlación de RDB_node, todas las sentencias de un elemento determinado deben correlacionarse con columnas de la misma tabla.

Procedimiento:

Para componer un documento XML desde la línea de mandatos utilizando la correlación de RDB_node:

1. Abra un editor de texto y cree una cabecera de la DAD escribiendo la sintaxis siguiente:

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

   Donde *vía*/dad.dtd es la vía de acceso y el nombre de archivo de la DTD para la DAD.
2. Inserte los códigos <DAD></DAD>. Este elemento contendrá los demás elementos.

3. Inserte los códigos utilizados para validar la DAD con una DTD o con un esquema.
   - Para validar la DAD con una DTD, inserte los códigos DTDID que asocian el archivo DAD con la DTD del documento XML. Por ejemplo:

     ```
     <dtdid>vía/nombre_dtd.dtd</dtdid>
     ```

   - Para validar la DAD con un esquema, inserte los códigos de esquema que asocian el archivo DAD con el archivo de esquema. Por ejemplo:

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

   El DTDID o el esquema sólo son útiles si decide validar el documento XML. Utilice el código de validación para indicar si DB2 UDB XML Extender valida el documento XML:
   - Si desea validar el documento XML, escriba:

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

   - Si no desea validar el tipo de documento XML:

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

4. Inserte los códigos <XCollection> </XCollection> para especificar que está utilizando las colecciones XML como método de acceso y almacenamiento de los datos XML.
5. Añada la información de prólogo (prolog) siguiente:

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

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

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

7. Inserte los códigos <root_node></root_node>. Dentro de los códigos de root_node, especifique los elementos y atributos que forman el documento XML.
8. Dentro del código <root_node>, correlacione los elementos y atributos en el documento XML con los nodos de elementos y atributos que corresponden a los datos DB2 UDB. Utilice el elemento RDB_node para el nodo de elemento (element_node), el nodo de texto (text_node) y el nodo de atributo (attribute_node). Estos nodos proporcionan una vía de acceso de los datos XML a los datos de DB2 UDB. Para correlacionar los elementos y atributos del documento XML:
   a. Especifique un RDB_node para el element_node superior. Este elemento especifica todas las tablas que están asociadas al documento XML. Para especificar un RDB_node para el element_node superior, inserte los códigos <RDB_node> después del código root_node.
      - Especifique un RDB_node para el attribute_node.
      - Especifique un RDB_node para el text_node.

   b. Defina un nodo de tabla para cada tabla que contenga datos que deben 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>
      ```

Si está descomponiendo un documento XML, mediante 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. La clave primaria se especifica añadiendo una clave de atributo al elemento de tabla del RDB_node. También debe especificar una clave primaria para cada tabla si va a habilitar una colección. El ejemplo siguiente muestra cómo especificar una columna de clave para cada tabla especificada en el nodo de elemento (element_node).

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

Conceptos relacionados:

- "Esquemas de correlación para colecciones XML" en la página 105
- "Vías de ubicación" en la página 114
- "Archivos DAD para colecciones XML" en la página 171
- "Requisitos para la correlación de RDB_Node" en la página 110
- "Procedimientos almacenados de composición XML Extender - Visión general" en la página 208

Tareas relacionadas:

- "Descomposición de una colección XML utilizando la correlación de RDB_node" en la página 68
- "Gestión de datos en colecciones XML" en la página 94
- "Actualización y supresión de datos en colecciones XML" en la página 102

Descomposición de una colección XML utilizando la correlación de RDB_node

Utilice la correlación de RDB_node para descomponer documentos XML. Este método utiliza el <RDB_node> para especificar las tablas, columnas y condiciones DB2 UDB para un nodo de elemento o 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 sobre la columna.

Los elementos hijo 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**
Elemento raíz	Sí	No	Sí[1]
Atributo	Sí	Sí	opcional
Texto	Sí	Sí	opcional

(1) Necesario si se utilizan varias tablas

Procedimiento utilizando una línea de mandatos::

Para descomponer los documentos XML utilizando una línea de mandatos:

1. Cree un archivo en cualquier editor de texto. Cree una cabecera DAD escribiendo la siguiente sintaxis:

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

   Donde *vía*/dad.dtd es la vía de acceso y el nombre de archivo de la DTD de la DAD.
2. Inserte los códigos <DAD></DAD>.
3. Inserte los códigos para validar la DAD con una DTD o esquema.
   - Para validar la DAD con una DTD, inserte los códigos DTDID que asocian el archivo DAD con la DTD del documento XML. Por ejemplo:

     ```
     <dtdid>vía/nombre_dtd.dtd</dtdid>
     ```
   - Para validar la DAD con un esquema, inserte los códigos de esquema que asocian el archivo DAD con el archivo de esquema. Por ejemplo:

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

   El DTDID o el esquema es útil sólo si decide validar el documento XML. Utilice el código de validación para indicar si DB2 UDB XML Extender valida el documento XML:
   - Si desea validar el documento XML, escriba:

     ```
     <validation>YES</validation>
     ```
   - Si no desea validar el tipo de documento XML:

     ```
     <validation>NO</validation>
     ```
4. Inserte los códigos <XCollection> </XCollection> para especificar que está utilizando colecciones XML como método de acceso y almacenamiento para los datos XML.
5. Añada la información de prólogo (prolog) siguiente:

   ```
   <prolog>?xml version="1.0"?</prolog>
   ```
6. Añada los códigos <doctype></doctype>. Por ejemplo:

   ```
   <doctype>! DOCTYPE Order SYSTEM "instalación_dxx
      /samples/db2xml/dtd/getstart.dtd"</doctype>
   ```
7. Defina el nodo raíz (root_node) utilizando los códigos <root_node></root_node>.
8. Dentro de root_node, correlacione los elementos y atributos del documento XML con los nodos de elementos y de atributos que corresponden a los datos de DB2 USB. Dichos nodos proporcionan una vía de acceso desde los datos XML a los datos de DB2 UDB.
   a. Defina un element_node (nodo de elemento) raíz, de nivel superior. Este element_node contiene:
      - Nodos de tabla con una condición de unión para especificar la colección.
      - Elementos hijo
      - Atributos

      Para especificar los nodos de tabla y la condición:

      1) Cree un elemento RDB_node dentro del elemento element_node. Por ejemplo:

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

2) Defina un table_node para cada tabla que contenga datos a incluir en el documento XML. Por ejemplo, si tiene tres tablas, ORDER_TAB, PART_TAB y SHIP_TAB, con datos de columna 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:

```
table_name.table_column = nombre_tabla.columna_tabla AND
nombre_tabla.columna_tabla = nombre_tabla.columna_tabla ...
```

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>
```

El atributo clave es necesario para la descomposición y habilitación de una colección.

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

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

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

text_node (nodo de texto)
Para especificar que el elemento tiene contenido en una tabla de DB2 UDB; en este caso no tiene elementos hijo.

attribute_node (nodo de atributo)
Para especificar un atributo.

elementos hijo
Hijos de element_node.

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

Los RDB_nodes se utilizan para elementos de nivel inferior que tengan contenido que correlacionar con una tabla de DB2 UDB. Un RDB_node contiene los siguientes elementos hijo:

table Especifica la tabla con la que se ha correlacionado el elemento o atributo.

column Especifica la columna con la que se ha correlacionado el elemento o atributo.

condition Opcionalmente especifica una condición que afecta a la inserción en 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, especifique un código <RDB_node> para correlacionar el elemento XML con la tabla y columna de DB2 UDB.

Archivo DAD:

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

El código <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 código <attribute_node> para cada atributo del documento XML que se correlacione con una columna en una tabla de DB2 UDB. Por ejemplo:

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

El attribute_node tiene un RDB_node para correlacionar el valor del atributo con una tabla y columna de DB2 UDB.

Por ejemplo, es posible que tenga 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 nodo de atributo (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>
```

9. 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 sin códigos. 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>
```

10. Asegúrese de que los códigos de finalización se encuentran en sus lugares correspondientes:
    a. Asegúrese de que un código de finalización </root_node> se encuentra después del último </element_node>.
    b. Asegúrese de que un código de finalización </Xcollection> se encuentra después del código </root_node>.
    c. Asegúrese de que un código de finalización </DAD> se encuentra después del código </Xcollection>.

Conceptos relacionados:

- "Procedimientos almacenados de descomposición de XML Extender - Visión general" en la página 220

Tareas relacionadas:

- "Descomposición de documentos XML en datos de DB2 UDB" en la página 98
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Parte 3. Programación

Esta parte describe técnicas de programación para gestionar los datos XML.

Capítulo 3. Columnas XML

Esta sección describe cómo gestionar los datos en columnas XML utilizando DB2.

Gestión de datos en columnas XML

Cuando se utilizan columnas XML para almacenar datos, se guarda un documento XML completo en el formato nativo como datos de columnas en DB2. Este método de acceso y almacenamiento permite conservar el documento XML inalterado, y al mismo tiempo permite indexar y hacer búsquedas en el documento, recuperar datos de él y actualizarlo.

Una vez habilitada una base de datos para XML, puede utilizar los siguientes UDT (tipos definidos por el usuario) proporcionados por XML Extender:

XMLCLOB
Utilice este UDT para el contenido del documento XML que se almacena como gran objeto de caracteres (CLOB) en DB2.

XMLVARCHAR
Utilice este UDT para el contenido del documento que se almacena como un VARCHAR en DB2.

XMLFILE
Utilice este UDT para un documento XML que está guardado en un archivo en un sistema de archivos local.

Puede crear o modificar tablas de aplicación para que tengan columnas de tipos de datos UDT. Estas tablas se denominan tablas XML.

Después de habilitar una columna en una tabla para XML, puede crear la columna XML y realizar las siguientes tareas de gestión:

- Almacenar documentos XML en DB2
- Recuperar datos o documentos XML desde DB2
- Actualizar documentos XML
- Suprimir datos o documentos XML

Para realizar todas estas tareas, utilice las UDF (funciones definidas por el usuario) proporcionadas por XML Extender. Utilice las funciones de conversión de datos por omisión para almacenar documentos en DB2. Las funciones de conversión de datos convierten el tipo base SQL en tipos definidos por el usuario de XML Extender y convierten instancias de un tipo de datos (origen) en instancias de un tipo de datos diferente (destino).

Conceptos relacionados:

- "Columnas XML como método de almacenamiento y acceso" en la página 76
- "Utilización de índices para datos de columna XML" en la página 77

Columnas XML como método de almacenamiento y acceso

Habrá ocasiones en las que desee almacenar y mantener la estructura del documento tal y como es en la actualidad. XML contiene toda la información necesaria para crear un conjunto de documentos.

Por ejemplo, si se trata de una agencia de noticias que divulga artículos por la Web, puede que desee mantener un archivo de los artículos publicados. En este caso, XML Extender permite almacenar los artículos XML completos o parciales en una columna de una tabla de DB2®, que es la *columna XML*, como se muestra en la Figura 9.

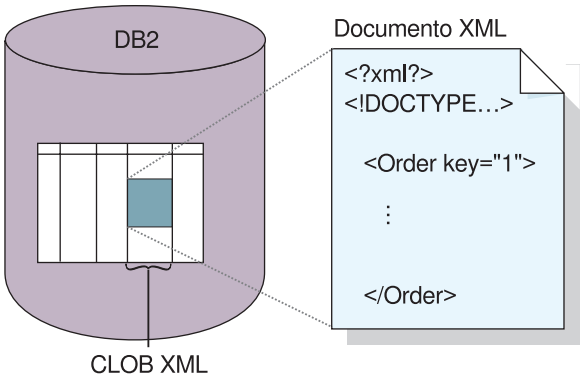


Figura 9. Almacenamiento de documentos XML estructurados en una columna de una tabla de DB2 UDB

El método de almacenamiento y acceso de columna XML permite gestionar los documentos XML utilizando DB2. Puede almacenar documentos XML en una columna de tipo XML y puede consultar el contenido del documento para buscar un elemento o un atributo específico. Puede asociar y almacenar una DTD en DB2 UDB para uno o más documentos. Además, puede correlacionar el contenido de los elementos y atributos con las tablas de DB2 UDB, denominadas *tablas auxiliares*. Dichas tablas pueden indexarse para mejorar el rendimiento de las consultas, pero no se indexan automáticamente. La columna que se utiliza para almacenar el documento se denomina columna XML. Especifica que la columna se utiliza como método de almacenamiento y acceso de columna XML.

En el archivo DAD (Definición de acceso a documento), escriba los códigos <Xcolumn> y </Xcolumn> para indicar que el método de almacenamiento y acceso que va a utilizar es será el de columna XML. A continuación, la DAD correlacionará el contenido del elemento y atributo XML que debe almacenarse en tablas auxiliares.

Antes de empezar a trabajar con XML Extender para almacenar los documentos, es necesario comprender la estructura del documento XML para poder determinar cómo indexar los elementos y atributos en el documento. Cuando planifique cómo indexar el documento, debe determinar:

- El tipo definido por el usuario XML en el que va a almacenar el documento XML
- Los elementos y atributos XML que la aplicación va a buscar con frecuencia, para que su contenido se pueda almacenar en tablas auxiliares e indexar para mejorar el rendimiento
- Si desea o no validar documentos XML en la columna con una DTD

Definición y habilitación de una columna XML

Las columnas XML se utilizan para almacenar y acceder a los documentos XML completos en la base de datos. Este método de almacenamiento permite guardar documentos utilizando los tipos de archivos XML, indexar las columnas en tablas auxiliares y consultar o buscar documentos XML.

Utilice las columnas XML cuando desee almacenar documentos XML completos en una columna de una tabla de DB2 si el documento no se va a actualizar con frecuencia o si desea almacenar documentos XML intactos.

Si desea correlacionar estructuras de documentos XML con tablas de DB2 UDB para poder componer documentos XML a partir de datos DB2 UDB existentes o descomponer documentos XML en datos de DB2, deberá utilizar las colecciones XML en lugar de las columnas XML.

Procedimiento:

Para definir y habilitar una columna XML desde la línea de mandatos:

1. Cree un archivo de definición de acceso a documento (DAD).
2. Cree una tabla donde se guardan los documentos XML.
3. Habilitar la columna para datos XML.
4. Indexar tablas auxiliares.

La columna XML se crea como un tipo de datos del usuario XML. Una vez completadas estas tareas, podrá guardar los documentos XML en la columna. A partir de este punto, estos documentos se pueden actualizar, buscar y extraer.

Conceptos relacionados:

- "Columnas XML como método de almacenamiento y acceso" en la página 76
- "Utilización de índices para datos de columna XML" en la página 77
- "Validación automática de documentos XML" en la página 54
- "Lección: Almacenamiento de un documento XML en una columna XML" en la página 8

Tareas relacionadas:

- "Creación de un archivo DAD para columnas XML" en la página 169
- "Creación de una tabla XML" en la página 55
- "Habilitación de columnas XML" en la página 57
- "Indexación de tablas auxiliares" en la página 62
- "Gestión de datos en columnas XML" en la página 75

Utilización de índices para datos de columna XML

Una decisión de planificación importante cuando se utilizan columnas XML es si se han de indexar las tablas auxiliares para documentos 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 estructural.

Cuando utiliza columnas XML, que contienen documentos XML completos, puede crear tablas auxiliares para contener columnas de valores de atributo o elementos 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 índices de DB2® en el motor de la base de datos para indexar datos de consulta frecuente de tipo general (tales como, enteros, decimales o fechas). XML Extender extrae los valores de los elementos y atributos XML de los documentos XML y los almacena en las tablas auxiliares, permitiendo al usuario crear índices en estas tablas auxiliares. Puede especificar cada columna de una tabla auxiliar utilizando una vía de ubicación, que identifica un elemento o atributo XML y un tipo de datos de SQL.

XML Extender llena automáticamente la tabla auxiliar cuando se almacenan datos XML en la columna XML.

Para poder realizar búsquedas rápidas, cree índices en dichas columnas utilizando la tecnología *indexación de la estructura de árbol B* de DB2 UDB. Consulte la documentación de DB2 UDB para obtener más información acerca de la *indexación de estructura de árbol B*.

Debe tener en cuenta las siguientes cuestiones a la hora de crear un índice:

- 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.
- Puede crear varios índices sobre una columna XML.
- 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 asocie 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, XML Extender modifica la tabla de la aplicación para añadirle una columna DXXROOT_ID, la cual almacena un ID exclusivo creado 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 DB2 UDB Net Search Extender, también podrá utilizar la característica de texto estructural de Net Search Extender. Net Search Extender tiene soporte para *"búsqueda por secciones"*, que amplía las capacidades de una búsqueda de texto completo convencional al permitir buscar palabras que coincidan con un determinado contexto de documento especificado por las vías de ubicación. El *indexado de texto estructural* puede utilizarse con el indexado de XML Extender en tipos de datos SQL generales.

Almacenamiento de datos XML

Utilizando XML Extender, podrá insertar documentos XML intactos en una columna XML. Si define tablas auxiliares, XML Extender actualiza automáticamente dichas tablas. Al almacenar un documento XML directamente, XML Extender almacena el tipo base como un tipo XML.

Requisitos previos:

- Asegúrese de hacer creado o actualizado el archivo DAD.
- Determine el tipo de datos que utilizará para almacenar el documento.
- Elija un método (funciones de conversión de datos o funciones UDF) para almacenar los datos en la tabla de DB2®.

Mediante una sentencia INSERT de SQL, especifique la tabla y columna XML donde residirá el documento XML.

XML Extender proporciona dos métodos para almacenar documentos XML: funciones por omisión de conversión de datos y UDF de almacenamiento.

La Tabla 7 muestra cuándo utilizar cada método.

Tabla 7. Funciones de almacenamiento de XML Extender

Si el tipo base de DB2 UDB es ...	**Almacenar en DB2 UDB como ...**			
	XMLVARCHAR	**XMLCLOB**	**XMLDBCLOB**	**XMLFILE**
VARCHAR	XMLVARCHAR()	N/D	N/D	XMLFile FromVarchar()
CLOB	N/D	XMLCLOB()	XMLDB CLOB, casting function	XMLFile FromCLOB()
FILE	XMLVarcha rFromFile()	XMLCLOB FromFile()	XMLDB CLOBFrom Archivo, UDF	XMLFILE

Funciones de conversión de datos por omisión para almacenar datos XML

Para cada UDT, existe una función por omisión de conversión para convertir el tipo base SQL al UDT. Puede utilizar las funciones de conversión de datos proporcionadas por XML Extender en la cláusula VALUES para insertar datos. La Tabla 8 muestra las funciones de conversión de datos proporcionadas:

Tabla 8. Funciones de conversión de datos por omisión de XML Extender

Función de conversión de datos	**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 almacena el nombre de archivo

Por ejemplo, la siguiente sentencia inserta un tipo convertido VARCHAR en el tipo XMLVARCHAR:

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

Almacenamiento de las UDF para almacenar datos XML

Para cada UDT de XML Extender, existe una UDF de almacenamiento para importar datos a DB2 desde un recurso distinto del tipo base de DB2. Por ejemplo, si desea importar un documento archivo XML a DB2 UDB como tipo XMLCLOB, puede utilizar la función XMLCLOBFromFile().

La Tabla 9 muestra las funciones de almacenamiento proporcionadas por XML Extender.

Tabla 9. UDF de almacenamiento de XML Extender

UDF de almacenamiento	Tipo devuelto	Descripción
XMLVarcharFromFile()	XMLVARCHAR	Lee un documento XML desde un archivo en el servidor y devuelve el valor del tipo de datos XMLVARCHAR. Opcional: Especifique la codificación del archivo.
XMLCLOBFromFile()	XMLCLOB	Lee un documento XML desde un archivo en el servidor y devuelve el valor del tipo de datos XMLCLOB. Opcional: Especifique la codificación del archivo.
XMLFileFromVarchar()	XMLFILE	Lee un documento XML de la memoria en forma de datos VARCHAR, graba el documento en un archivo externo y devuelve el valor del tipo de datos XMLFILE, que es el nombre de archivo. Opcional: Especifique la codificación del archivo externo.
XMLFileFromCLOB()	XMLFILE	Lee un documento XML de la memoria como datos CLOB o como localizador CLOB, graba el documento en un archivo externo y devuelve el valor del tipo de datos XMLFILE, que es el nombre de archivo. Opcional: Especifique la codificación del archivo externo.

Por ejemplo, al utilizar la función XMLCLOBFromFile(), la siguiente sentencia graba un registro en una tabla XML como un XMLCLOB:

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

Este ejemplo importa el documento XML del archivo llamado `instalación_dxx/samples/db2xml/xml/getstart.xml` en la columna ORDER en la tabla SALES_TAB.

Método para recuperar un documento XML

Utilizando XML Extender, podrá recuperar un documento completo o el contenido de elementos y atributos. Cuando recuperar directamente una columna XML, XML Extender devuelve el UDT como tipo de columna. Para obtener más detalles acerca de recuperar datos, consulte las siguientes secciones:

- "Recuperación de un documento XML completo"
- "Recuperación del contenido de elementos y de los valores de atributo de los documentos XML" en la página 83

XML Extender proporciona dos métodos para recuperar datos: funciones por omisión de conversión de datos y la UDF sobrecargada Content(). La Tabla 10 muestra cuándo hay que utilizar cada método.

Tabla 10. Funciones de recuperación de XML Extender

Cuando el tipo XML es ...	**Recuperar de DB2 UDB como ...**			
	VARCHAR	**CLOB**	**DBCLOB**	**FILE**
XMLVARCHAR	**VARCHAR**	N/D	N/D	UDF Content()
XMLCLOB	N/D	**XMLCLOB**	N/D	UDF Content()
XMLFILE	N/D	UDF Content()	N/D	FILE

Recuperación de un documento XML completo

Procedimiento:

Para recuperar un documento XML completo:

1. Asegúrese de que ha almacenado el documento XML en una tabla XML y determine qué datos desea recuperar.
2. Elija un método (funciones de conversión o UDF) para recuperar datos en la tabla de DB2 UDB.
3. Si utiliza la UDF sobrecargada Content(), determine el tipo de los datos que se recuperan y el tipo de datos que se van a exportar.
4. La columna XML desde la que se extrae el elemento o atributo debe estar definida como tipo de datos XMLVARCHAR, XMLCLOB como LOCATOR o XMLFILE.

Especifique una consulta SQL que especifique la tabla y la columna XML desde las que desea recuperar el documento XML.

Funciones de conversión de datos por omisión para recuperar datos XML

La función de conversión por omisión proporcionada por DB2 UDB para UDT convierte un UDT XML en un tipo base SQL y luego realiza operaciones con él. En la sentencia SELECT, puede utilizar funciones de conversión de datos que XML Extender proporciona para recuperar datos. La Tabla 11 en la página 82 muestra las

funciones de conversión de datos proporcionadas.

Tabla 11. Funciones de conversión de datos por omisión de XML Extender

Conversión de datos utilizada en la cláusula SELECT	Tipo devuelto	Descripción
varchar(XMLVARCHAR)	VARCHAR	Documento XML en VARCHAR
clob(XMLCLOB)	CLOB	Documento XML en CLOB
varchar(XMLFile)	VARCHAR	Nombre de archivo XML en VARCHAR

Por ejemplo, la siguiente sentencia recupera XMLVARCHAR y lo almacena en memoria como un tipo de datos VARCHAR:

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

Utilización de la UDF Content() UDF para recuperar datos XML

Utilice la UDF Content() para recuperar el contenido del documento desde el almacenamiento externo a la memoria, o bien para exportar el documento desde el almacenamiento interno a un archivo externo, es decir, un archivo que sea externo a DB2 UDB en el servidor de DB2 UDB.

Por ejemplo, puede que tenga el documento XML almacenado como un tipo de datos XMLFILE. Si desea realizar operaciones con él en la memoria, puede utilizar la UDF Content(), que puede tomar como entrada un tipo de datos XMLFILE y devolver un CLOB.

La UDF Content() realiza dos funciones de recuperación diferentes, dependiendo del tipo de datos especificado. Permite:

- Recuperar un documento del almacenamiento externo y colocarlo en memoria.

  Puede utilizar la UDF Content() UDF para recuperar el documento XML y colocarlo en una memoria intermedia o un *localizador* de CLOB (una variable de lenguaje principal con un valor que representa un valor LOB individual en el servidor de base de datos) cuando el documento se almacena como archivo externo.

  Utilice la siguiente sintaxis de función, donde *objxml* es la columna XML que se está consultando:

  De XMLFILE a CLOB:

  ```
  Content(objxml XMLFile)
  ```

- Recuperar un documento del almacenamiento interno y exportarlo a un archivo externo.

  Puede utilizar la UDF Content() para recuperar un documento XML que se encuentre almacenado dentro de DB2 UDB como tipo de datos XMLCLOB y exportarlo a un archivo en el sistema de archivos del servidor de bases de datos. La UDF Content() devuelve el nombre del archivo como un tipo de datos VARCHAR.

  Utilice la siguiente sintaxis de función:

  Del tipo XML al archivo externo:

  ```
  Content(objxml tipo XML, nombreArchivo varchar(512),
        codificaciónDestino varchar(100))
  ```

  Donde:

objxml Es el nombre de la columna XML desde la cual se va a recuperar el contenido XML. *objxml* puede ser de tipo XMLVARCHAR o XMLCLOB.

nombreArchivo
Es el nombre del archivo externo en el que se deben almacenar los datos XML.

codificaciónDestino
Opcional: Especifica la codificación del archivo de salida.

En el ejemplo siguiente, un pequeño segmento de programa C con sentencias de SQL incorporado (sentencias de SQL codificadas dentro de un programa de aplicación) muestra cómo se recupera un documento XML desde un archivo y se coloca en la memoria. Este ejemplo da por supuesto que el tipo de datos de la columna ORDER es 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 { /* hace lo necesario con el documento XML en el almac. int. */}
       }
    EXEC SQL CLOSE c1;
    EXEC SQL CONNECT RESET;
```

Recuperación del contenido de elementos y de los valores de atributo de los documentos XML

Puede recuperar (extraer) el contenido de un elemento o el valor de un atributo de uno o más documentos XML (búsqueda en documento individual o en documento de colección). XML Extender proporciona funciones de extracción definidas por el usuario que se pueden especificar en la cláusula SELECT de SQL para cada uno de los tipos de datos SQL.

Recuperar el contenido de elementos y valores de atributos es útil al desarrollar aplicaciones ya que permite acceder a los datos XML como datos relacionales. Por ejemplo, suponga que tiene 1000 documentos XML almacenados en la columna ORDER de la tabla SALES_TAB. Para recuperar los nombres de todos los clientes que hayan pedido artículos de más de 2500 dólares, utilice la siguiente sentencia de SQL con la UDF a extraer en la cláusula SELECT:

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
        WHERE price > 2500,00
```

En este ejemplo, la UDF de extracción recupera el contenido del elemento <customer> de la columna ORDER y lo almacena como tipo de datos VARCHAR. La vía de ubicación es `/Order/Customer/Name`. Adicionalmente, el número de valores devueltos se reduce utilizando una cláusula WHERE, la cual especifica que únicamente el contenido del elemento <customer> con un subelemento <ExtendedPrice> tenga un valor mayor de 2500.00.

La Tabla 12 en la página 84 muestra las UDF que pueden utilizarse para extraer el contenido de elementos o valores de atributos, utilizando la siguiente sintaxis funciones de tabla o escalares.

Sintaxis:

```
extract retrieved_datatype(objxml, vía_acceso)
```

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

objxml Es el nombre de la columna XML desde la que se debe extraer el elemento o el atributo. 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 en el documento XML (como por ejemplo, `/Order/Customer/Name`).

Restricción: Las UDF de extracción pueden dar soporte a vías 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 12 muestra las funciones de extracción, en formato escalar y de tabla.

Tabla 12. Funciones de extracción de 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

Tabla 12. Funciones de extracción de XML Extender (continuación)

Función escalar	Función de tabla	Nombre de columna devuelto (función de tabla)	Tipo devuelto
extractTimestamp()	extractTimestamps()	returnedTimestamp	TIMESTAMP

Ejemplo de función escalar: En el ejemplo siguiente, se inserta un valor con el valor de clave de atributo de 1. El valor se extrae como entero y se convierte automáticamente a un tipo DECIMAL.

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

Actualización de datos XML

Con XML Extender, es posible actualizar todo el documento XML sustituyendo los datos de columna XML o actualizar los valores de elementos o atributos especificados.

Procedimiento

Para actualizar los datos XML:

1. El documento XML debe actualizarse en una tabla XML.
2. Es necesario saber qué tipo de datos desea recuperar.
3. Es necesario elegir un método para actualizar los datos en la tabla de DB2 UDB (funciones de conversión o UDF).
4. Mediante una consulta SQL, especifique la tabla y columna XML que se debe actualizar.

Actualización de un documento XML completo

Puede actualizar un documento XML utilizando una función de conversión de datos por omisión o una UDF de almacenamiento.

Actualización con una función por omisión de conversión de datos

Para cada UDT (tipo definido por el usuario), existe una función de conversión de datos por omisión para convertir el tipo base SQL al UDT. Puede utilizar las funciones de conversión de datos proporcionadas por XML Extender para actualizar el documento XML completo.

Por ejemplo, la siguiente sentencia actualiza el tipo XMLVARCHAR a partir del tipo VARCHAR de conversión de datos, dando por supuesto que `xml_buf` es una variable del lenguaje principal que está definida como un tipo VARCHAR.

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

Actualización de documentos XML con una UDF de almacenamiento

Para cada uno de los UDT de XML Extender, existe una UDF de almacenamiento para importar los datos a DB2 UDB desde un recurso distinto de su tipo base. Puede utilizar una UDF de almacenamiento para actualizar el documento XML completo mediante su sustitución.

El ejemplo siguiente actualiza el objeto XML desde el archivo denominado instalación_dxx/samples/db2xml/xml/getstart.xml para la columna ORDER de la tabla SALES_TAB.

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

Actualización de elementos y atributos específicos de un documento XML

Utilice la UDF Update para efectuar cambios específicos en vez de actualizar el documento completo. Cuando utilice esta UDF, especifique la vía de ubicación del elemento o del atributo cuyo valor se va a sustituir. No es necesario editar el documento XML; XML Extender efectúa el cambio automáticamente.

Sintaxis:

```
Update(objxml, vía, valor)
```

La sintaxis contiene los componentes siguientes:

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.

valor Es el nuevo valor que se debe actualizar.

Por ejemplo, la siguiente sentencia sustituye el valor de elemento <xxCustomer> por IBM:

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

Aparición múltiple: Cuando se especifica 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. Si la vía de ubicación aparece en un documento más de una vez, la UDF Update sustituye todos los valores existentes por el valor proporcionado en el parámetro *valor*.

Métodos para la 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 como criterio de recuperación.

XML Extender proporciona varios métodos para buscar documentos XML que están almacenados en una columna XML. El usuario puede:

- Hacer una búsqueda estructural del documento y obtener resultados basados en el contenido de elementos o valores de atributos.
- Efectuar una búsqueda de una vista de la columna XML y las tablas auxiliares.
- Efectuar una búsqueda directa de las tablas auxiliares para obtener un mejor rendimiento.
- Efectuar una búsqueda utilizando las UDF de extracción con cláusulas WHERE.
- Utilice DB2® Net Search Extender para buscar una serie de texto en los datos de columnas dentro del contenido estructural.

Con XML Extender puede utilizar índices para efectuar una búsqueda rápida de las columnas en las tablas auxiliares. Estas columnas contienen el contenido de elementos o los valores de atributos XML que se extraen de los documentos XML. Al especificar el tipo de datos de un elemento o un atributo, puede buscar en un tipo de datos SQL o efectuar búsquedas de rangos. Por ejemplo, en el supuesto del pedido de compra, podría buscar todos los pedidos cuyo precio global sea mayor que 2500,00.

Además, puede utilizar Net Search Extender para realizar búsqueda de texto estructural o de texto completo. Por ejemplo, suponga que tiene una columna denominada RESUME que contiene currículos de personas en formato XML. Si desea buscar los nombres de todos los candidatos que tengan conocimientos de Java™, podrá utilizar DB2 UDB Net Search Extender para buscar en los documentos XML todos los currículos donde el elemento <skill> (conocimientos) contenga la serie de caracteres "JAVA".

Las siguientes secciones describen los métodos de búsqueda:

- "Búsqueda del documento XML por la estructura"
- "Utilización de DB2 UDB Net Search Extender para realizar búsquedas de texto estructural de documentos XML" en la página 89

Búsqueda del documento XML por la estructura

Mediante las funciones de búsqueda de XML Extender, podrá buscar datos XML en una columna basándose en la estructura del documento (los elementos y atributos del documento).

Procedimientos:

Para buscar los datos, puede:

- Consultar directamente las tablas auxiliares.
- Utilizar una *vista de unión.*
- Utilizar las UDF de extracción.

Estos métodos de búsqueda se describen en los siguientes ejemplo y se basan la siguiente situación. La tabla SALES_TAB tiene una columna XML denominada ORDER. Esta columna contiene tres tablas auxiliares, ORDER_SIDE_TAB, PART_SIDE_TAB y SHIP_SIDE_TAB. Se especificó una vista por omisión, sales_order_view, cuando se habilitó la columna ORDER. Esta vista une estas tablas 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
```

Ejemplo: búsqueda con una consulta directa en tablas auxiliares

La búsqueda mediante consulta directa o subconsulta proporciona un rendimiento óptimo para una búsqueda estructural cuando las tablas auxiliares están indexadas.

Procedimiento:

Puede utilizar una consulta o subconsulta para buscar en tablas auxiliares correctamente.

Por ejemplo, la siguiente sentencia utiliza una consulta y subconsulta para buscar directamente 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.

Ejemplo: búsqueda desde una vista de unión

XML Extender puede crear una vista por omisión que una la tabla de aplicación y las tablas auxiliares utilizando un ID exclusivo. Puede utilizar esta vista por omisión o cualquier vista que una tabla de aplicación y las tablas auxiliares, para buscar datos de columnas y consultar las 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 tardará la consulta en ejecutarse.

Sugerencia: Puede crear el ID raíz o DXXROOT_ID (creado por XML Extender), para unir las tablas al crear su propia vista.

Por ejemplo, la siguiente sentencia busca la vista denominada SALES_ORDER_VIEW y devuelve los valores de la columna SALES_PERSON donde los pedidos de unidades por línea tienen un precio superior a 2500,00.

```
SELECT sales_person from sales_order_view
   WHERE price > 2500,00
```

Ejemplo: búsqueda mediante las UDF de extracción

También puede utilizar las UDF de extracción de 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 utilizarse con cláusulas WHERE que restrinjan el número de documentos XML a incluir en la búsqueda.

La siguiente sentencia busca mediante una UDF de extracción de 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 contengan la subserie IBM®.

Ejemplo: búsqueda de elementos o atributos de aparición múltiple

Cuando se realiza una búsqueda de elementos y atributos que tengan varias apariciones, utilice la cláusula DISTINCT para impedir valores duplicados.

La siguiente sentencia realiza una búsqueda con 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` es de aparición múltiple y crea la tabla auxiliar PART_SIDE_TAB para ese elemento. 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 de DB2 UDB Net Search Extender para realizar búsquedas de texto estructural de documentos XML

Si DB2 UDB Net Search Extender se encuentra instalado, puede utilizarlo para realizar una búsqueda de texto estructural.

Procedimiento:

Para utilizar DB2 UDB Net Search Extender:

1. Decida si desea utilizar la búsqueda de texto estructural o la búsqueda de texto completo.
2. Habilite una columna XML para DB2 UDB Net Search Extender.
3. Cree una consulta para efectuar la búsqueda.

Para aprender cómo utilizar la búsqueda de DB2 UDB Net Search Extender, consulte la publicación DB2 Universal Database Extenders: Net Search Extender Administración y programación, Versión 7.

Utilización de búsquedas de texto estructural y búsquedas completas de texto

Al buscar en la estructura de un documento XML, XML Extender busca aquellos elementos convertidos a tipos de datos generales, pero no busca texto. Puede utilizar Net Search Extender para realizar búsquedas de texto estructural o búsquedas de texto completo en una columna habilitada para XML. DB2 UDB Net Search Extender da soporte a la búsqueda de documentos XML en DB2 UDB Versión 6.1 o posterior. Net Search Extender se encuentra disponible en los sistemas operativos AIX®, Windows®, iSeries y en el entorno operativo Solaris™.

Búsqueda estructural de texto

Busca series de texto que están basadas en la estructura de árbol del documento XML. Por ejemplo, en una estructura de documento de `/Order/Customer/Name`, puede utilizar una búsqueda de texto estructural para buscar la serie de caracteres ″IBM″ dentro del subelemento <Customer>. Sin embargo, el documento también puede tener la serie de texto ″IBM″ en un subelemento <Comment> o como parte del nombre de un producto. Una búsqueda de texto estructural sólo busca la serie en el elemento que está especificado. En este ejemplo, sólo se encontrarán los elementos que tienen ″IBM″ en el subelemento </Order/Customer/Name>; no se devolverá ningún documento que contenga ″IBM″ en otros elementos, excepto en el subelemento </Order/Customer/Name>.

Búsqueda completa de texto

Busca series de texto en todos los lugares de la estructura del documento, sin distinguir elementos ni atributos. Siguiendo con el ejemplo anterior, se devolverían todos los documentos que contienen la serie ″IBM″, independientemente del lugar donde aparece la serie.

Habilitación de una columna XML para DB2 UDB Net Search Extender

En una base de datos habilitada para XML, se habilita a DB2 UDB Net Search Extender para que busque el contenido de una columna habilitada para XML. En este ejemplo, la base de datos se denomina SALES_DB, la tabla es ORDER y los nombres de columnas XML son XVARCHAR y XCLOB.

1. Consulte el archivo `install.txt` en el CD de DB2 UDB Extenders para obtener más información acerca de cómo instalar Net Search Extender.
2. Ejecute el mandato **txstart**:
   - En los sistemas operativos UNIX®, escriba el mandato en el indicador de mandatos del propietario de la instancia.
   - En Windows NT®, entre el mandato desde la ventana de mandatos donde está especificado DB2INSTANCE.
3. Abra la ventana de la línea de mandatos de Net Search Extender y conéctese a la base de datos. En el indicador de mandatos de **db2tx**, escriba:

   ```
   connect to SALES_DB
   ```
4. Habilite la base de datos para DB2 UDB Net Search Extender.

   En el indicador de mandatos de **db2tx**, escriba:

   ```
   enable database
   ```
5. Habilite las columnas en la tabla XML para DB2 UDB Net Search Extender y defina los tipos de datos del documento XML, el lenguaje, páginas de códigos y otra información acerca de la columna.
   - Para la columna XVARCHAR, de tipo VARCHAR, escriba:

     ```
     enable text column order xvarchar function
     db2xml.varchartovarchar handle varcharhandle ccsid 1252
     language us_english format xml indextype precise
         indexproperty sections_enabled
     documentmodel (Order) updateindex update
     ```
   - Para la columna XCLOB, de tipo CLOB, escriba:

     ```
     enable text column order xclob
     function db2xml.clob handle clobhandle ccsid 1252
     language us_english indextype precise updateindex update
     ```
6. 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
     ```
7. Defina el modelo de documento XML en un archivo de inicialización de modelos de documentos denominado `desmodel.ini`. Este archivo se encuentra en el directorio `/db2tx/txins000` en UNIX y en el directorio `/instance/db2tx/txins000` en Windows NT. 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/Shipment/ShipMode = /Order/Part/Shipment/ShipMode
```

Búsqueda de texto utilizando DB2 UDB Net Search Extender

Para buscar texto utilizando DB2 UDB Net Search Extender, cree una consulta que especifique el elemento o atributo que desea buscar. A continuación, DB2 UDB Net Search Extender utiliza la consulta para buscar el contenido del elemento o valores del atributo.

Por ejemplo, entre las siguientes sentencias en una ventana de mandatos de DB2 UDB para hacer que DB2 UDB Net Search Extender busque el texto de un documento XML:

```
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() de Net Search Extender busca el texto de un documento XML.

Este ejemplo no contiene todos los pasos necesarios para que DB2 UDB Net Search Extender busque los datos en la columna. Para aprender acerca de los conceptos de búsqueda y capacidad de Net Search Extender, consulte la publicación *DB2 Universal Database Extenders: Net Search Extender Administración y programación*.

Supresión de documentos XML

Utilice la sentencia DELETE de SQL para suprimir la fila que contiene un documento XML de una columna XML. Puede especificar una cláusula WHERE para suprimir documentos específicos.

Por ejemplo, la siguiente sentencia suprime todos los documentos que tienen un valor para <ExtendedPrice> mayor que 2500,00:

```
DELETE from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500,00)
```

Las filas correspondientes en las tablas auxiliares se suprimen automáticamente.

Conceptos relacionados:

- "Columnas XML como método de almacenamiento y acceso" en la página 76

Tareas relacionadas:

- "Gestión de datos en columnas XML" en la página 75

Limitaciones al invocar funciones desde la base de datos de Java (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.

Por ejemplo, JDBC no puede resolver el siguiente código:

```
DB2XML.XML función_por_omisión_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_por_omisión_conversión(CAST(? AS
tipo_conv(longitud))
```

Ejemplos:

En los siguientes ejemplo, la tabla `Sales_Tab` tiene tres columnas. La columna `invoice_num` tiene un tipo de datos de Char(6), la columna `sales_person` tiene un tipo de datos de Varchar(20) y la columna `order` tiene un tipo de datos de XMLVarchar.

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 4. Gestión de datos en colecciones XML

Colecciones XML como método de almacenamiento y acceso

Los datos relacionales se *descomponen* a partir de documentos XML de entrada o se utilizan para *componer* documentos XML de salida. Los datos descompuestos son el contenido sin códigos de un documento XML almacenado en una o más tablas de base de datos. O bien los documentos XML se componen a partir de datos existentes en una o más tablas de base de datos. Si tiene datos que necesita compartir con otras aplicaciones, quizá desee poder componer y descomponer los documentos XML de entrada y salida, y gestionar los datos como sea necesario para sacar provecho de las posibilidades relacionales de DB2®. Este tipo de almacenamiento de documentos XML se denomina *colección XML*.

En la Figura 10 se proporciona un ejemplo de colección XML.

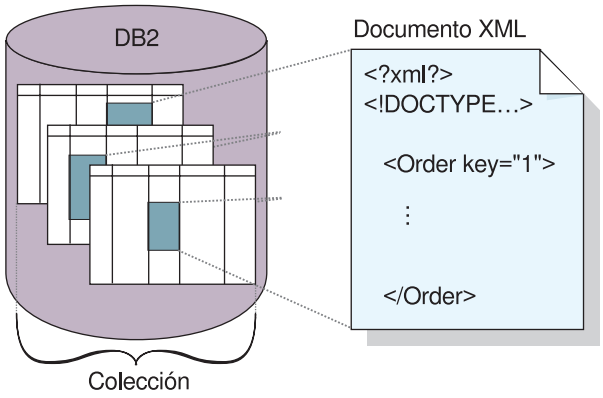


Figura 10. Almacenamiento de documentos como datos sin códigos en tablas de DB2 UDB

La colección XML se define en un archivo DAD, que especifica cómo se correlacionan los elementos y atributos con una o más tablas relacionales. La colección es un conjunto de columnas, asociadas con un archivo DAD, que contiene los datos de un documento XML determinado o un conjunto de documentos XML. Puede definir un nombre de colección habilitándolo y, a continuación, hacer referencia al mismo por su nombre al emitir un procedimiento almacenado para componer o descomponer documentos XML. Se denomina colección XML habilitada. Se asigna un nombre a la colección para que se ejecute fácilmente con los procedimientos almacenados que componen y descomponen los documentos XML.

Cuando se define una colección en el archivo DAD, se utiliza uno de dos tipos de esquemas de correlación, la *correlación de SQL* o la *correlación de RDB_node* que definen las tablas, columnas y condiciones utilizadas para asociar los datos XML con tablas de DB2 UDB. La correlación de SQL utiliza sentencias SQL SELECT para definir las tablas y condiciones de DB2 UDB utilizadas para la colección. La correlación de RDB_node utiliza un nodo de base de datos relacional basado en XPath o RDB_node que tiene elementos hijo.

Los procedimientos almacenados se proporcionan para componer o descomponer documentos XML. Los nombres de procedimientos almacenados utilizan el calificador DB2XML, que es el *nombre de esquema* de XML Extender.

Gestión de datos en colecciones XML

Una colección XML es un conjunto de tablas relacionales que contiene datos que se correlacionan con documentos XML. Este método de acceso y almacenamiento permite componer un documento XML a partir de datos existentes, descomponer un documento XML y utilizar XML como método de intercambio de datos.

Las tablas relacionales que componen la colección pueden ser tablas nuevas o tablas ya existentes que contengan datos a utilizar con XML Extender para componer documentos XML para las aplicaciones. Los datos de las columnas de estas tablas no contienen códigos XML, sino el contenido y los valores asociados a elementos y atributos, respectivamente. Puede utilizar los procedimientos almacenados para almacenar, recuperar, actualizar, buscar y suprimir datos de colección XML.

Preparación para componer documentos XML a partir de datos de DB2

La composición es el proceso de creación de 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, cree una definición de acceso a documento (DAD). Un archivo DAD especifica la correlación entre el documento XML y la estructura de la tabla de DB2. Los procedimientos almacenados utilizan el archivo DAD para componer el documento XML.

Procedimiento::

Antes de comenzar a componer documentos XML:

1. Correlacione la estructura del documento XML con las tablas relacionales que contienen el contenido de los elementos y los valores de atributos.
2. Seleccione un método de correlación: Correlación de SQL o correlación de RDB_node.
3. Prepare el archivo DAD.

XML Extender proporciona cuatro procedimientos almacenados, dxxGenXML(), dxxGenXMLCLOB(), dxxRetrieveXML() y dxxRetrieveXMLCLOB para componer documentos XML. La frecuencia con la que se tiene la intención de actualizar el documento XML es un factor clave para seleccionar el procedimiento almacenado que se va a utilizar.

Composición de documentos XML actualizados frecuentemente

Si el documento sólo se va a actualizar ocasionalmente, utilice el procedimiento almacenado dxxGenXML para componer el documento. Para utilizar este procedimiento almacenado, no tiene que habilitar una colección. En su lugar, el procedimiento almacenado utiliza un archivo DAD.

El procedimiento almacenado dxxGenXML crea documentos XML utilizando datos que están guardados 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 aplicación debe crear la tabla de resultados y debe tener siempre una columna de tipo VARCHAR, CLOB, CHAR o XMLCLOB utilizada para almacenar los datos XML.

Adicionalmente, si el valor del elemento de validación en el archivo DAD es YES, XML Extender añade la columna DXX_VALID del tipo INTEGER a la tabla de resultados si la columna DXX_VALID no se encuentra ya en la tabla. XML Extender inserta un valor de 1 para un documento XML válido y 0 para un documento no válido.

El procedimiento almacenado dxxGenXML también permite especificar el número máximo de filas que se deben generar en la tabla resultante. Esto reduce el tiempo de proceso. El procedimiento almacenado devuelve el número real de filas en la tabla y cualquier código de retorno o mensajes.

El procedimiento almacenado correspondiente para la descomposición es dxxShredXML; también utiliza el archivo DAD como parámetro de entrada y no necesita que la colección XML esté habilitada.

Procedimiento:

Para componer una colección XML utilizando el procedimiento almacenado dxxGenXML, intercale una llamada de procedimiento almacenado en la aplicación utilizando la declaración de procedimiento almacenado siguiente:

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

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

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; se establecerá en 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_ind;
              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);

     /* leer datos de un archivo y ponerlos en un CLOB */
     strcpy(dadfile.name,"instalación_dxx
      /samples/dad/getstart_xcollection.dad");
     dadfile.name_length = strlen("instalación_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';
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 db2xml.dxxGenXML(:dad:dad_ind;
                           :result_tab:rtab_ind,
                           :overrideType:ovtype_ind,:override:ov_ind,
                           :max_row:maxrow_ind,:num_row:numrow_ind,
                           :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Después de llamar el procedimiento almacenado, la tabla resultante contiene 250 filas porque la consulta SQL especificada en el archivo DAD generó 250 documentos XML.

Composición de documentos XML actualizados frecuentemente

Si el documento se va a actualizar con frecuencia, utilice el procedimiento almacenado dxxRetrieveXML para componer el documento. Debido a que se repiten las mismas tareas, es importante conseguir un óptimo rendimiento.

El procedimiento almacenado dxxRetrieveXML funciona de la misma manera que el procedimiento almacenado dxxGenXML, salvo que utiliza 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 lo tanto XML Extender recupera el archivo DAD y lo utiliza para componer el documento de la misma manera que el procedimiento almacenado dxxGenXML.

El procedimiento almacenado dxxRetrieveXML permite que el mismo archivo DAD se utilice tanto para la composición como para la descomposición.

El procedimiento almacenado correspondiente para la descomposición es dxxInsertXML; también utiliza el nombre de una colección XML habilitada.

Procedimiento:

Para componer una colección XML utilizando el procedimiento almacenado dxxRetrieveXML, intercale una llamada de procedimiento almacenado en la aplicación utilizando la declaración de procedimiento almacenado siguiente:

```
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 */
```

Ejemplo: El ejemplo siguiente es de una llamada dxxRetrieveXML(). Se supone que se crea una tabla resultante con el nombre de XML_ORDER_TAB y que la tabla tiene una columna de tipo XMLVARCHAR.

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

            EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
              char    collection;   /* almacenamiento intermedio en dad */
              char    result_tab[32];   /* nombre de la tabla resultante */
              char    override[2];      /* alt. temp; se establecerá en 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   collection_ind;
              short   rtab_ind;
              short   ovtype_ind;
              short   ov_ind;
              short   maxrow_ind;
              short   numrow_ind;
              short   returnCode_ind;
              short   returnMsg_ind;

EXEC SQL END DECLARE SECTION;

              /* crear tabla */
              EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

              /* inicializar variable del 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 db2xml.dxxRetrieveXML(:collection:collection_ind;
                           :result_tab:rtab_ind,
                           :overrideType:ovtype_ind,:override:ov_ind,
                           :max_row:maxrow_ind,:num_row:numrow_ind,
                           :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Conceptos relacionados:

- "Colecciones XML como método de almacenamiento y acceso" en la página 93
- "Esquemas de correlación para colecciones XML" en la página 105
- "Vías de ubicación" en la página 114
- "Archivos DAD para colecciones XML" en la página 171
- "Procedimientos almacenados de composición XML Extender - Visión general" en la página 208

Tareas relacionadas:

- "Composición de colecciones XML utilizando la correlación de RDB_node" en la página 66

- "Hojas de estilo para una colección XML" en la página 113
- "Descomposición de una colección XML utilizando la correlación de RDB_node" en la página 68
- "Actualización y supresión de datos en colecciones XML" en la página 102
- "Búsqueda de colecciones XML" en la página 104

Descomposición de documentos XML en datos de DB2 UDB

Descomponer un documento XML implica desglosar los datos del interior de un documento XML y almacenarlos en tablas relacionales. XML Extender proporciona procedimientos almacenados para descomponer datos XML a partir de documentos XML fuente, en tablas relacionales. Para utilizar dichos procedimientos almacenados, debe crear un archivo DAD, que especifica la correlación entre el documento XML y la estructura de tabla de DB2 UDB. Los procedimientos almacenados utilizan el archivo DAD para descomponer el documento XML.

Habilitación de una colección XML para la descomposición

En la mayoría de los casos, es necesario habilitar una colección XML antes de utilizar los procedimientos almacenados. Los casos en los que debe habilitar las colecciones son:

- Al descomponer documentos XML en tablas nuevas, debe habilitarse una colección XML porque XML Extender crea todas las tablas de la colección cuando se habilita la colección.
- Cuando sea importante mantener la secuencia de elementos y atributos que aparecen varias veces (aparición múltiple). XML Extender sólo conserva el orden de secuencia de los elementos o atributos de aparición múltiple para las tablas que se crean cuando se habilita una colección. Cuando se descomponen documentos XML en tablas relacionales existentes, no se puede garantizar que se conserve el orden de secuencia.

Consulte la sección sobre el mandato de administración dxxadm para informarse acerca de la opción enable_collection.

Si desea transferir el archivo DAD cuando las tablas ya existen en la base de datos, no es necesario habilitar una colección XML.

Antes de descomponer un documento XML en datos de DB2 UDB:

1. Correlacione la estructura del documento XML con las tablas relacionales que contienen el contenido de los valores de atributos y elementos.
2. Prepare el archivo DAD, utilizando la correlación de RDB_node.
3. Opcional: Habilite la colección XML.

Procedimiento::

Utilice uno de los dos procedimientos almacenados que DB2 UDB XML Extender proporciona para descomponer documentos XML, dxxShredXML() o dxxInsertXML.

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. Inserta datos de un documento XML en una colección XML de acuerdo con la especificación <Xcollection> del archivo DAD de entrada. Luego, el procedimiento almacenado dxxShredXML() descompone el documento XML e inserta los datos XML sin códigos en las tablas especificadas en el archivo DAD. 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.

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 con 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 */
```

Ejemplo: El ejemplo siguiente es una llamada a dxxShredXML():

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

EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
      SQL TYPE is CLOB(100K) dad;       /* DAD*/
       SQL TYPE is CLOB_FILE  dadFile;   /* archivo DAD */
      SQL TYPE is CLOB(1M)  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 del lenguaje principal e indicadores */
        strcpy(dadFile.name,
"instalación_dxx/samples/db2xml/dad/
        getstart_xcollection.dad");
        dadFile.name_length=strlen("instalación_dxx
       /samples/db2xml/dad/getstart_xcollection.dad");
        dadFile.file_option=SQL_FILE_READ;
       strcpy(xmlFile.name,"instalación_dxx
       /samples/db2xml/xml/getstart_xcollection.xml");
       xmlFile.name_length=strlen
      ("instalación_dxx/samples/db2xml/xml
      /getstart_xcollection.xml");
```

```
                xmlFile.file_option=SQL_FILE_READ;
                SQL EXEC VALUES (:dadFile) INTO :dad;
                SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
          returnCode = 0;
          returnMsg[0] = '\0';
                dad_ind = 0;
xmlDoc_ind = 0;
          returnCode_ind = -1;
          returnMsg_ind = -1;

/* 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. 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 datos de 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.

Procedimiento:

Para descomponer una colección XML: dxxInsertXML():

Intercale una llamada de procedimiento almacenado en su aplicación, mediante esta declaración:

```
dxxInsertXML(char(
             ) collectionName,   /* entrada */

             CLOB(1M)       xmlobj,          /* entrada */
             long           returnCode,      /* salida */
             varchar(1024)  returnMsg)       /* salida */
```

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(1M)  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 del lenguaje principal e indicadores */
 strcpy(collection,"sales_ord")strcpy
(xmlobj.name,"instalación_dxx/samples/db2xml
      /xml/getstart_xcollection.xml");
xmlobj.name_length=strlen("instalación_dxx/samples/db2xml
/xml/getstart_xcollection.xml");

      xmlobj.file_option=SQL_FILE_READ;
      SQL EXEC VALUES (:xmlFile) INTO (:xmlDoc);
      returnCode = 0;
      returnMsg[0] = '\0';
      collection_ind = 0;
      xmlobj_ind = 0;
      returnCode_ind = -1;
      returnMsg_ind = -1;

/* Llamar al procedimiento almacenado */
 EXEC SQL CALL DB2XML.dxxInsertXML
       (:collection:collection_ind;
                                     :xmlDoc:xmlDoc_ind,
                        :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Límites de tamaño de la tabla de descomposición

La descomposición utiliza la correlación RDB_node para especificar cómo se descompone un documento XML en tablas de DB2 UDB extrayendo los valores de elementos y atributos y almacenándolos en filas de tablas. Los valores de cada documento XML se almacenan en una o más tablas de DB2 UDB. Cada tabla puede tener un máximo de 10240 filas descompuestas de cada documento.

Por ejemplo, si se descompone un documento XML en cinco tablas, cada una de las cinco tablas puede tener hasta 10240 filas para dicho documento. Si la tabla tiene filas para varios documentos, puede tener hasta 10240 filas 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 se utilizan elementos de aparición múltiple, tenga en cuenta que se pueden descomponer un máximo de 1024 filas en una tabla desde un solo documento.

Conceptos relacionados:

- “Procedimientos almacenados de descomposición de XML Extender - Visión general” en la página 220

Tareas relacionadas:

- “Descomposición de una colección XML utilizando la correlación de RDB_node” en la página 68
- “Invocación de procedimientos almacenados de XML Extender” en la página 199

Información relacionada:

- “Procedimiento almacenado dxxInsertXML()” en la página 222
- “Procedimiento almacenado dxxShredXML()” en la página 220

Actualización y supresión de datos en colecciones XML

Puede actualizar, suprimir, buscar y recuperar colecciones XML. No obstante, el propósito de utilizar una colección XML es el de almacenar o recuperar los datos sin códigos en tablas de base de datos. Los datos de las tablas de base de datos existentes no guardan relación alguna con ningún documento XML de entrada; las operaciones de actualización, supresión y búsqueda constan del acceso normal mediante SQL a estas tablas.

XML Extender permite realizar operaciones en los datos desde una vista de colección XML. Es posible ejecutar las sentencias de SQL UPDATE y DELETE para modificar los datos utilizados para componer documentos XML y, por lo tanto, para actualizar la colección XML. Al realizar operaciones de SQL en las tablas de la colección se verán afectados los documentos generados.

- 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 compone el nodo de elemento (element_node) superior especificado en la DAD.

Actualización de datos en una colección XML

XML Extender permite actualizar datos sin códigos que se almacenan en tablas de una colección XML. Cuando se actualizan los valores de las tablas de una colección XML, está actualizando el texto de un elemento XML o el valor de un atributo XML. Las actualizaciones pueden también suprimir una instancia de datos de elementos o atributos que aparezcan varias veces.

Desde el punto de vista de SQL, cambiar el valor del elemento o del atributo es una operación de actualización y suprimir una instancia de un elemento o un atributo es una operación de supresión. Desde el punto de vista de XML, mientras exista el texto de elemento o el valor de atributo del nodo de elemento (element_node) raíz, seguirá existiendo el documento XML y, por consiguiente, una operación de actualización. Las operaciones de SQL en las tablas de la colección afectan a los documentos que se van a generar de las tablas.

Requisitos: Cuando actualice datos en una colección XML, siga las normas 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 de SQL, incluya la condición de unión en el elemento <SQL_stmt>.
  - Para la correlación de RDB_node, incluya la condición de unión en el elemento superior <condition> 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 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 de 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.

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. Si no suprime la fila que contiene los valores del element_node superior, la supresión de los valores del elementos se considerará una actualización del documento XML.

Por ejemplo, en la siguiente sentencia DELETE, se suprime un elemento <shipment> especificando un valor exclusivo de uno de los 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. La realización de operaciones de SQL en las tablas de la colección afectará a los documentos generados.

Procedimiento:

Para suprimir el documento, suprima una fila de la tabla que compone el nodo de elemento (element_node) superior 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. La supresión del documento mediante este método sólo funciona si las restricciones de clave primaria y de clave foránea están completamente especificadas en el SQL y si la relación de las tablas visualizada en la DAD coincide exactamente con dichas restricciones.

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, que es la tabla de nivel superior especificada en la DAD. 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.

Búsqueda de colecciones XML

Esta sección describe la búsqueda de una colección XML en cuanto ala generación de documentos XML utilizando los criterios de búsqueda y la búsqueda de los datos XML descompuestos.

Composición de documentos XML utilizando criterios de búsqueda

Esta tarea equivale a componer utilizando una condición.

Procedimiento:

Puede especificar los criterios de búsqueda siguientes:

- Especifique la condición en el text_node y attribute_node del archivo DAD
- Especifique el parámetro *override* al utilizar 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 utilizando datos de formulario 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 del 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 el price_value de algún lugar, como por ejemplo, */
         price_value = 1000,00         /* de los datos*/

         /* 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)  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, a continuación, efectuar una búsqueda de texto estructural en las columnas de texto. Aplique los resultados de la búsqueda estructural para recuperar o generar el documento XML especificado.

Esquemas de correlación para colecciones XML

Si está utilizando una colección XML, debe seleccionar un *esquema de correlación*, que especifica 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 para bases de datos relacionales, es conveniente comprender cómo se corresponden las dos estructuras. La Figura 11 en la página 106 muestra cómo la estructura jerárquica se puede correlacionar con columnas de una tabla relacional.

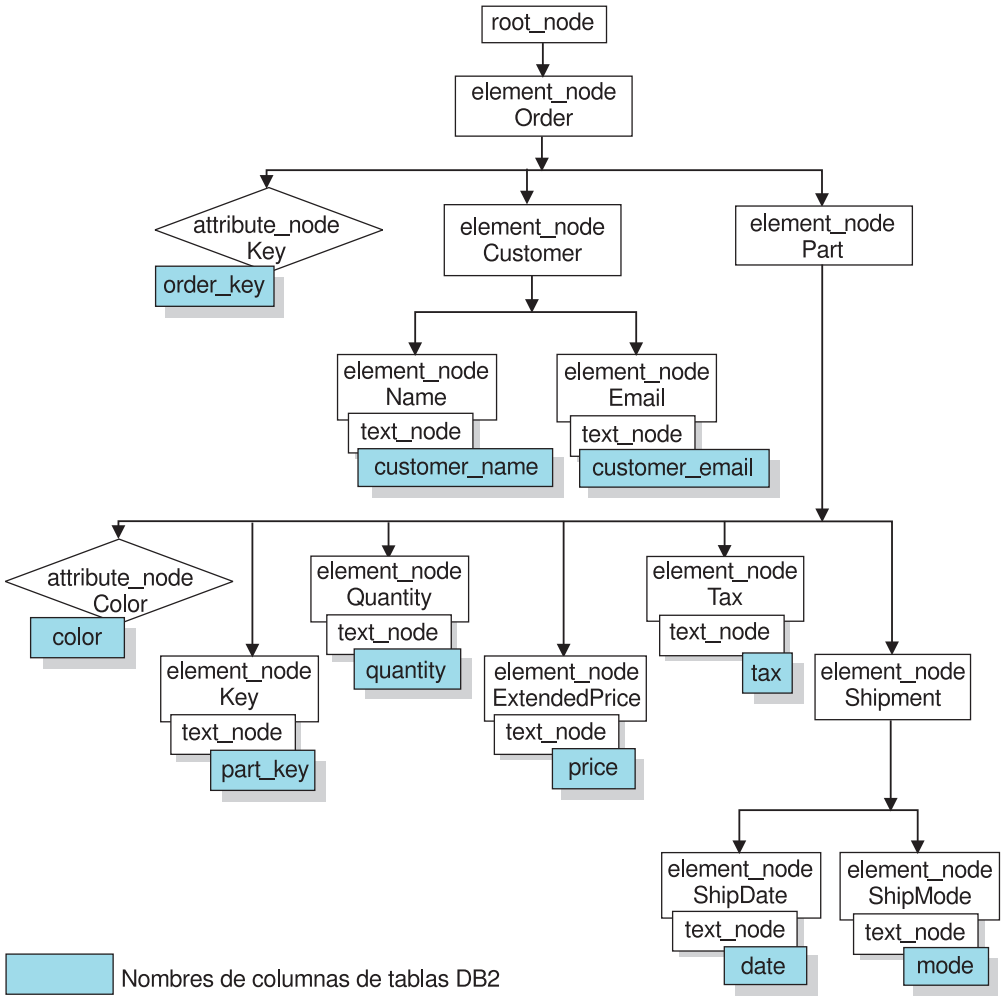


Figura 11. Documento XML estructurado correlacionado con columnas de tablas relacionales

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

Tipos de esquemas de correlación:

Utilice <Xcollection> para especificar el esquema de correlación del archivo DAD. XML Extender proporciona dos tipos de esquemas de correlación: *correlación de SQL* y *correlación de base de datos relacional (RDB_node)*.

Correlación de SQL

Este método permite una correlación directa entre datos relacionales y documentos XML mediante una sola sentencia de SQL. La correlación de SQL sólo se utiliza para la composición. El contenido del elemento <SQL_stmt> debe ser una sentencia de SQL válida. El elemento <SQL_stmt> especifica columnas en la cláusula SELECT que están correlacionados con elementos o atributos XML más adelante en el archivo DAD. Cuando se definen para componer documentos XML, los nombres de columna de la cláusula SELECT de la sentencia de SQL se utilizan para asociar el valor de un *nodo_atributo* o un contenido de un *nodo_texto* con

columnas que tienen el mismo *atributo_nombre*.La cláusula FROM define las tablas que contienen los datos; la cláusula WHERE especifica la condición de *unión* y *búsqueda*.

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

Correlación de RDB_node

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

Este método utiliza el *RDB_node* proporcionado por XML Extender, que contiene una o más definiciones para las tablas, columnas opcionales y condiciones opcionales. Los elementos <table> y <column> en DAD definen 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, debe crear un archivo DAD con un elemento <Xcollection>. La Figura 12 en la página 108 muestra un fragmento de un archivo DAD de ejemplo con una correlación de SQL para una colección XML, que compone un conjunto de documentos XML a partir de datos de tres tablas relacionales.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM
"instalación_dxx/samples/db2xml/dtd/dad.dtd">
<DAD>
  <dtdid>instalación_dxx/samples/dad/getstart.dtd</dtdid>
   <validation>YES</validation>
  <Xcollection>
    <SQL_stmt>
       SELECT o.order_key, customer, p.part_key, quantity, price, tax, date,
              ship_id, mode, comment
                FROM order_tab o, part_tab p,
               table(select substr(char(timestamp
(generate_unique())),16)
                as ship_id, date, mode, from ship_tab)
        WHERE p.price > 2500,00 and s.date > "1996-06-01" AND
              p.order_key = o.order_key and s.part_key = p.part_key
    </SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
    <doctype>!DOCTYPE DAD SYSTEM
    "instalación_dxx/samples/db2xml/dtd/getstart.dtd
     "</doctype>
      <root_node>
    <element_node name="Order">
    <attribute_node name="Key">
                <column name="order_key"/>
    </attribute_node>
    <element_node name="Customer">
     <text_node>
      <column name="customer"/>
     </text_node>
    <element_node>

  ...

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

Figura 12. Esquema de correlación de SQL

XML Extender proporciona varios procedimientos almacenados que gestionan datos en una colección XML. Estos procedimientos almacenados permiten utilizar ambos tipos de correlación.

Conceptos relacionados:

- "Archivos DAD para colecciones XML" en la página 171
- "Requisitos para utilizar la correlación de SQL" en la página 109
- "Requisitos para la correlación de RDB_Node" en la página 110

Tareas relacionadas:

- "Composición de documentos XML utilizando la correlación de SQL" en la página 62
- "Composición de colecciones XML utilizando la correlación de RDB_node" en la página 66
- "Descomposición de una colección XML utilizando la correlación de RDB_node" en la página 68

Requisitos para utilizar la correlación de SQL

Requisitos cuando se utiliza la correlación de SQL

En este esquema de correlación, debe especificar el elemento <SQL_stmt> dentro del elemento <Xcollection> de la DAD. <SQL_stmt> debe contener una sola sentencia de SQL que pueda unir varias tablas relacionales con la consulta *predicado*. 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 alias para una de las columnas.
  - Agrupe las columnas de la misma tabla conjuntamente y ordene las tablas de acuerdo con el nivel de estructura de árbol cuando se correlacionan con la estructura jerárquica del documento XML. La primera columna de cada agrupación de columna es un ID de objeto. En la cláusula SELECT, las columnas de las tablas del nivel superior deben 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, las columnas `order_key` y `customer` de la tabla ORDER_TAB tienen el nivel relacional más alto porque están más altas en el árbol jerárquico del documento XML. Las columnas `ship_id`, `date` y `mode` de la tabla SHIP_TAB tienen 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 una para dicha tabla utilizando una expresión de tabla y la función 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. `ship_id` se genera como una clave primaria porque la tabla SHIP_TAB no tiene una clave primaria. `ship_id` aparece listada como la primera columna para el 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 generate_unique() para generar una sola clave para las tablas que no tengan una sola clave 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, part key from ship_tab) as s
    ```

    En este ejemplo, se genera una clave adecuada de columna individual con la función generate_unique() y se le asigna un nombre de alias `ship_id`.

- Utilice un nombre de alias cuando necesite diferenciar una columna. Por ejemplo, podría utilizar `o` para las columnas de la tabla ORDER_TAB, `p` para las columnas de la tabla PART_TAB y `s` para las columnas de la tabla SHIP_TAB.

- **Cláusula WHERE**
  - Especifique una relación de 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 no haya nada después de los nombres de columna como ASC o DESC.
  - Asegúrese de que los nombres de columna coincidan con la cláusula SELECT.
  - Liste todos los ID del objeto en el mismo orden relativo en el que aparecen en la cláusula SELECT.
  - Se puede generar un identificador utilizando una expresión de tabla y la función generate_unique() o una función definida por el usuario.
  - Mantenga el orden descendente de la jerarquía de las entidades. La primera columna especificada en la cláusula ORDER BY debe ser la primera columna listada para cada entidad. La conservación del orden asegura que no haya duplicados incorrectos en los documentos XML que se crearán.
  - No califique las columnas en la cláusula ORDER BY con un nombre de esquema o de tabla.

El elemento <SQL_stmt> es potente porque puede especificar cualquier predicado en la cláusula WHERE, siempre y cuando la expresión del predicado utilice las columnas de las tablas.

Información relacionada:

• Apéndice A, “Ejemplos”, en la página 305

Requisitos para la correlación de RDB_Node

Cuando utilice RDB_Node como método de correlación, no utilice el elemento <SQL_stmt> en el elemento <Xcollection> del archivo DAD. En su lugar, utilice el elemento RDB_node como hijo del element_node superior y para cada attribute_node y text_node.

- **RDB_node para el element_node superior**

  El element_node superior del archivo DAD representa el elemento raíz del documento XML. Especifique un RDB_node para el element_node superior del modo siguiente:

  - Especifique todas las tablas que están asociadas a la colección XML. Por ejemplo, la correlación siguiente especifica tres tablas en el <RDB_node> 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.

– Los elementos de condición pueden hacer referencia a un nombre de columna un número ilimitado de veces.
– Si está habilitando una colección, deberá especificar una clave primaria para cada tabla. La clave primaria consta de una o varias columnas; en este segundo caso se denomina clave compuesta. Especifique la clave primaria añadiendo una *clave de atributo* al elemento de tabla de RDB_node. Cuando suministre una clave compuesta, el atributo de *clave* se especificará 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 se pasa por alto si se utiliza la misma DAD para la composición.

– Utilice el atributo orderBy para recomponer documentos XML que contienen elementos o atributos de aparición múltiple y devolverlos a su estructura original. Este atributo permite especificar el nombre de una columna que se utilizará para preservar el orden del documento. El atributo orderBy forma parte del elemento de tabla en el archivo DAD y es un atributo opcional. Cuando se descomponen documentos XML en una colección XML, puede perder el orden de los elementos y valores de atributos de múltiple aparición, a no ser que especifique el orden en el archivo DAD. Para preservar dicho orden, debe utilizar el esquema de correlación RDB_node y especificar el atributo orderBy para la tabla que contenga el elemento raíz en su RDB_node.

Escriba el nombre de tabla y el nombre de columna en el código <table>.

- **RDB_node para cada attribute_node y text_node**

  XML Extender tiene que saber en qué lugar de la base de datos debe recuperar los datos. Asimismo, XML Extender tiene que saber en qué lugar de la base de datos debe colocar el contenido de un documento XML. Debe especificar un RDB_node para cada nodo de atributo y nodo de texto. También debe especificar los nombres de tabla y columna; el valor de condición es opcional.

  1. Especifique el nombre de la tabla donde residen los datos de columna. El nombre de tabla debe incluirse en el RDB_node del element_node superior. En este ejemplo, para el text_node del elemento <Price>, la tabla se especifica 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>
```

2. 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.
3. Especifique una condición de consulta si desea que se generen documentos XML utilizando dicha condición. Los documentos XML resultantes sólo contendrán los datos que cumplen la condición. La condición debe ser una cláusula WHERE válida. En el ejemplo anterior, la condición está especificada como `price > 2500,00`, por lo que sólo se incluirán en los documentos XML las filas en las que el precio sea superior a 2500.
4. Si está descomponiendo un documento o habilitando la colección XML especificada por el archivo DAD, debe especificar el tipo de columna para cada nodo de atributo y nodo de texto. Al especificar el tipo de columna para cada nodo de atributo y nodo de texto, se asegura que el tipo de datos sea el correcto para cada columna cuando se crean tablas nuevas durante la habilitación de 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"/>
   ```

   El tipo de columna especificado cuando se descompone un documento se omite para la composición.

- Mantenga el orden descendente de la jerarquía de las entidades. Asegúrese de que los nodos de elementos se anidan correctamente para que XML Extender comprenda la relación entre los elementos al componer o descomponer documentos. Por ejemplo, el siguiente archivo DAD no anida Shipment dentro de Part:

  ```
   <element_node name="Part">
       ...
  <element_node name="ExtendedPrice">
             ...
        </element_node>
             ...
  </element_node> <!-- end of element Part -->

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

     </element_node> <!-- end of element Shipment-->
  ```

  Este archivo DAD genera un documento XML en el que los elementos Part y Shipment son iguales.

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

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

  El siguiente código muestra el elemento Shipment anidado dentro del elemento Par del archivo DAD.

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

    </element_node> <!-- end of element Shipment-->
</element_node> <!-- end of element Part -->
```

Al anidar el elemento Shipment dentro del elemento Part se genera un archivo XML con Shipment como elemento hijo del elemento Part:

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

No existen restricciones de clasificación respecto a los predicados de la condición de nodo raíz.

Con el método de correlación de RDB_node, no es necesario suministrar sentencias de SQL. Sin embargo, establecer condiciones de consulta complejas en el elemento de RDB_node puede ser más difícil.

Para un subárbol de la DAD con los element_node y attribute_node que correlacionan con la misma tabla, se aplica lo siguiente:

- Los nodos de atributo no tienen necesariamente que ser los primeros hijos del ancestro común más bajo de los nodos de elementos que se correlacionan con la misma tabla.
- Los nodos de atributos pueden aparecer en cualquier parte del subárbol, siempre que no estén incluidos en una condición de unión.

Restricciones: El límite del número de tablas permitidas en una correlación DAD de RDB_node es de 30. El número de columnas permitidas por tabla es de 500. El número de veces que puede especificarse cada tabla o columna en los predicados de unión de la sentencia de condición es ilimitado

Hojas de estilo para una colección XML

Al componer documentos, XML Extender también da soporte al procesado de instrucciones 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 version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
```

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

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

Vías de ubicación

Una *vía de ubicación* define la ubicación de un elemento o atributo XML dentro de la estructura del documento XML. XML Extender utiliza la vía de ubicación con el siguiente objetivo:

- Para localizar los elementos y atributos que deben extraerse cuando se utilizan las UDF de extracción, como por ejemplo, dxxRetrieveXML.
- Para especificar la correlación entre un elemento XML o un atributo y una columna DB2® cuando se define el esquema de indexación en la DAD para columnas XML
- Para la búsqueda de texto estructural, utilice Net Search Extender
- Para alterar temporalmente los valores del archivo DAD para la colección XML en un procedimiento almacenado.

La Figura 13 muestra un ejemplo de una vía de ubicación y su relación con la estructura del documento XML.

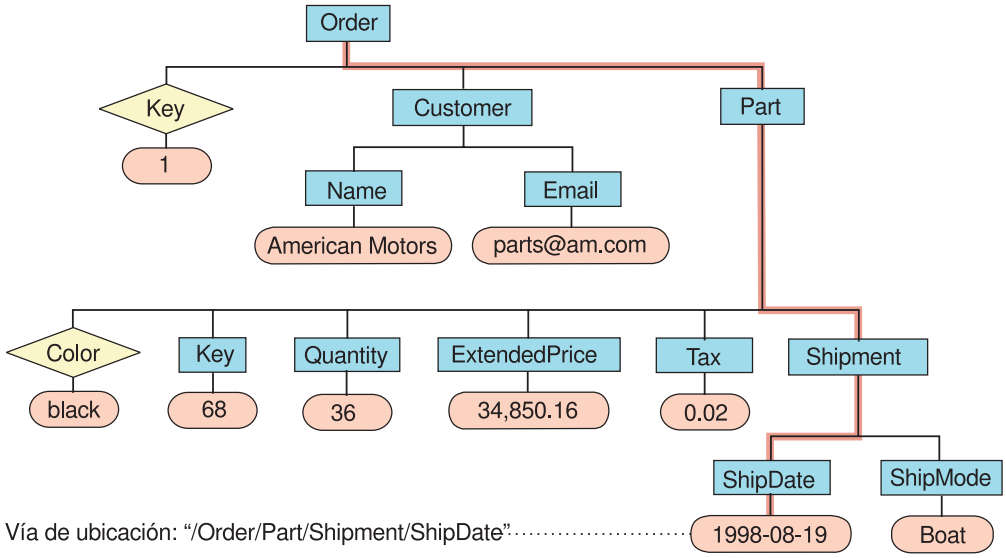


Figura 13. Almacenamiento de documentos como documentos XML estructurados en una columna de una tabla de DB2 UDB

Información relacionada:

- "Sintaxis de la vía de ubicación" en la página 115

Sintaxis de la vía de ubicación

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

1. / Representa el elemento raíz de XML. Es el elemento que contiene los demás elementos del documento.

2. */código1*
Representa el elemento *código1* debajo del elemento raíz.

3. */código1/código2/..../códigon*
Representa un elemento con el nombre *códigon* como hijo de la serie descendente que va desde el elemento raíz, *código1*, *código2*, hasta *códigon*-1.

4. *//códigon*
Representa cualquier elemento bajo el nombre *códigon*, donde las barras inclinadas dobles (//) indican cero o más códigos arbitrarios.

5. */código1//códigon*
Representa cualquier elemento con el nombre *códigon*, un descendiente de un elemento con el nombre *código1* debajo del elemento raíz, donde las barras inclinadas dobles (//) indican cero o más códigos arbitrarios.

6. */código1/código2/@atr1*
Representa el atributo *atr1* de un elemento con el nombre *código2*, que desciende del elemento *código1* debajo del elemento raíz.

7. */código1/código2[@atr1="5"]*
Representa un elemento con el nombre *código2* cuyo atributo *atr1* tiene el valor 5. *código2* es hijo del elemento *código1* debajo del elemento raíz.

8. */código1/código2[@atr1="5"]/.../códigon*
Representa un elemento con el nombre *códigon*, que es hijo de la serie descendente que va desde el elemento raíz, *código1*, *código2*, hasta *códigon*-1, donde el atributo *atr1* de *código2* tiene el valor 5.

Vía de ubicación simple

Vía de ubicación simple es un tipo de vía de ubicación que se utiliza en el archivo DAD de la columna XML. Una vía de ubicación simple se representa como una secuencia de nombres de tipos de elementos que están conectados por una barra inclinada (/). Los valores de cada atributo están entre corchetes, a continuación del tipo de elemento. La Tabla 13 resume la sintaxis de la vía de ubicación simple.

Tabla 13. Sintaxis de la vía de ubicación simple

Sujeto	Vía de ubicación	Descripción
Elemento XML	*/código1/código2/..../códigon-1/códigon*	Contenido de un elemento identificado por el elemento denominado *códigon* y sus padres
Atributo XML	*/código_1/código_2/..../código_n-1/código_n/@atr1*	Atributo denominado *atr1* del elemento identificado por *códigon* y sus padres

Utilización de la vía de ubicación

La sintaxis de la vía de ubicación depende del contexto en el que se acceda a la ubicación de un elemento o un atributo. Debido a que XML Extender utiliza la correlación unilateral entre un elemento o un atributo y una columna DB2, restringe las normas de sintaxis del archivo DAD y de las funciones. La Tabla 14 describe en qué contextos se utilizan las opciones de sintaxis.

Tabla 14. Restricciones de XML Extender para utilizar la vía de ubicación

Uso de la vía de ubicación	Vía de ubicación soportada
Valor del atributo de la vía de acceso en la correlación de DAD de columna XML para las tablas auxiliares	*/tag1/tag2/..../tagn* y */tag1/tag2/@attr1* (vía de ubicación simple descrita en la Tabla 13 en la página 115)
Funciones UDF de extracción	Todas las vías de ubicación[1]
UDF de actualización	Todas las vías de ubicación[1]
UDF de búsqueda de Net Search Extender	*/tag1/tag2/..../tagn* — Excepción: la marca de root (raíz) se especifica sin la barra inclinada. Por ejemplo: `código1/código2/..../códigon`

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

Conceptos relacionados:

- "Vías de ubicación" en la página 114

Habilitación de colecciones XML

Al habilitarse 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. La habilitación de una colección XML es opcional para:

- Descomponer un documento XML y almacenar los datos en nuevas tablas de DB2 UDB
- Componer un documento XML a partir de datos existentes en varias tablas de DB2 UDB

Si utiliza el mismo archivo DAD para la composición y la descomposición, puede habilitar la colección para una y otra operación.

Puede habilitar una colección XML con el Asistente de administración de XML Extender, con el mandato **dxxadm** con la opción enable_collection, o bien con el procedimiento almacenado de XML Extender dxxEnableCollection().

Utilización del Asistente de administración:

Para habilitar una colección XML utilizando el asistente:

1. Configure e inicie el Asistente de administración.
2. Pulse **Trabajar con colecciones XML** desde la ventana Área de ejecución. Se abrirá la ventana Seleccionar tarea.
3. Pulse **Habilitar una colección** y luego **Siguiente**. Se abrirá la ventana Habilitar una colección.
4. Seleccione el nombre de la colección que desea habilitar en el campo **Nombre de colección**.
5. Especifique el nombre de archivo DAD en el campo **Nombre de archivo DAD**.

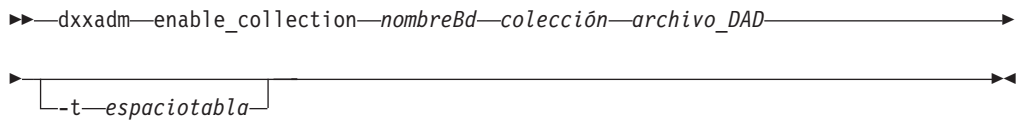
6. Opcional: Escriba el nombre de un espacio de tabla creado anteriormente en el campo **Espacio de tabla**.

   El espacio de tabla contendrá las nuevas tablas de DB2 UDB generadas para su descomposición.
7. Pulse **Finalizar** para habilitar la colección y regresar a la ventana Área de ejecución.
   - Si la colección se habilita satisfactoriamente, se visualizará el mensaje `Colección habilitada satisfactoriamente`.
   - Si la colección no se habilita satisfactoriamente, se visualizará un mensaje de error. Repita los pasos anteriores hasta que la colección se haya habilitado satisfactoriamente.

Habilitación de colecciones utilizando el mandato dxxadm:

Para habilitar una colección XML, entre el mandato **dxxadm** desde una línea de mandatos de DB2 UDB:

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).

espaciotabla
: Un espacio de tabla existente que contiene nuevas tablas de DB2 UDB que se generaron para la descomposición. Si no se especifica este parámetro, se utilizará el espacio de tabla por omisión.

Ejemplo: El siguiente ejemplo habilita una colección llamada sales_ord en la base de datos SALES_DB utilizando la línea de mandatos. El archivo DAD utiliza la correlación de SQL.

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

Una vez habilitada la colección XML, podrá componer o descomponer documentos XML utilizando los procedimientos almacenados de XML Extender.

Conceptos relacionados:

- "Colecciones XML como método de almacenamiento y acceso" en la página 93

Tareas relacionadas:

- "Inhabilitación de colecciones XML" en la página 118

- "Gestión de datos en colecciones XML" en la página 94

Inhabilitación de colecciones XML

Al inhabilitar una colección XML se elimina el registro en la tabla XML_USAGE que identifica las 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 o descartar una colección.

Puede inhabilitar una colección XML con el Asistente de administración de XML Extender, con el mandato **dxxadm** con la opción disable_collection o bien con el procedimiento almacenado de XML Extender dxxDisableCollection().

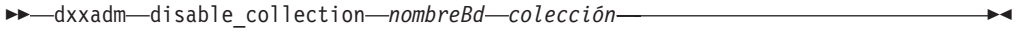
Procedimiento:

Para inhabilitar una colección XML utilizando el asistente de Administración:

1. Inicie el Asistente de administración.
2. Pulse **Trabajar con colecciones XML** en la ventana del Área de tareas para ver las tareas relacionadas con la colección de 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. Escriba el nombre de la colección que desea inhabilitar en el campo **Nombre de colección**.
5. Pulse **Finalizar** para inhabilitar la colección y volver a la ventana Área de ejecución.
   - Si se ha inhabilitado satisfactoriamente la colección, aparecerá el mensaje `Colección inhabilitada satisfactoriamente`.
   - Si la colección no se inhabilita satisfactoriamente, se visualiza una ventana de error. Repita los pasos anteriores hasta que se haya inhabilitado satisfactoriamente la colección.

Para inhabilitar una colección XML desde la línea de mandatos, escriba 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
```

Conceptos relacionados:

- "Colecciones XML como método de almacenamiento y acceso" en la página 93

- "Procedimientos almacenados de administración de XML Extender - Visión general" en la página 202

Tareas relacionadas:

- "Gestión de datos en colecciones XML" en la página 94

Capítulo 5. Esquemas XML

El esquema XML puede utilizarse en lugar de una DTD para definir las especificaciones del contenido de los documentos XML. El esquema XML utiliza el formato o sintaxis XML para definir los elementos y nombres de atributos de un documento XML, y define el tipo de contenido que los elementos y atributos pueden contener.

Ventajas de la utilización de esquemas XML en lugar de DTD

Las DTD son más sencillas de programar y validar que un esquema. No obstante, a continuación se listan las ventajas de utilizar un esquema XML:

- Los esquemas XML son documentos XML válidos que se pueden procesar mediante herramientas tales como XSD Editor en WebSphere® Studio Application Developer, XML Spy o XML Authority.
- Los esquemas XML son más potentes que las DTD. Todo aquello que puede definirse utilizando la DTD puede también definirse utilizando esquemas, aunque no así viceversa.
- Los esquemas XML dan soporte a un conjunto de tipos de datos, similares a los utilizados en la mayoría de lenguajes de programación, y proporcionan la capacidad de crear tipos adicionales. Puede restringir el contenido del documento según el tipo adecuado. Por ejemplo, puede replicar las propiedades de los campos encontrados de DB2®.
- Los esquemas XML dan soporte a expresiones regulares para establecer restricciones sobre los datos de caracteres, lo cual no es posible si se utiliza una DTD.
- Los esquemas XML proporcionan mejor soporte para los espacios de nombres XML, lo que le permite validar documentos que utilizan varios espacios de nombres y volver a utilizar construcciones de esquemas ya definidos en diferentes espacios de nombres.
- Los esquemas XML proporcionan mejor soporte para la modularidad y la reutilización con elementos de inclusión e importación.
- Los esquemas XML utilizan la herencia para definiciones de elementos, atributos y tipos de datos.

Tareas relacionadas:

- “Tipos de datos, elementos y atributos en esquemas” en la página 122

Información relacionada:

- “Ejemplo de un esquema XML” en la página 123

Elemento complexType del esquema XML

El elemento de esquema XML complexType se utiliza para definir un tipo de elemento que consiste de subelementos. Por ejemplo, los siguientes códigos muestran la proyección de una dirección en un documento XML:

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

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

La estructura de este elemento puede definirse en el esquema XML del modo siguiente:

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

En el ejemplo anterior, se asume el prefijo `xsd` está vinculado al espacio de nombres del esquema XML. Las líneas 2 a 12 definen complexType USAddress como una secuencia de cinco elementos y un atributo. El orden de los elementos está determinado por el orden que siguen al aparecer en el código sequence.

Los elementos interiores son del tipo de datos xsd:string o xsd:decimal. Ambos son tipos de datos simples previamente definidos.

Opcionalmente, puede utilizar el código <all> o <choice> en lugar del código <sequence>. Con todos los códigos, deben aparecer todos los subelementos, pero no necesariamente en un orden determinado. Con el código ″choice″, exactamente uno de los subelementos debe aparecer en el documento XML

También se puede utilizar un tipo de datos definidos por el usuario para definir otros elementos.

Tipos de datos, elementos y atributos en esquemas

Tipos de datos simples en esquemas XML

Los esquemas XML proporcionan un conjunto de tipos de datos incorporados simples. Puede hacer que otros tipos de datos procedan de ellos aplicando restricciones.

En el Ejemplo 1, el rango del tipo base xsd:positiveInteger queda limitado de 0 a 100.

Ejemplo 1

```
< xsd:element name="quantity">
   < xsd:simpleType>
    < xsd:restriction base="xsd:positiveInteger">
     < xsd:maxExclusive value="100"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
```

En el Ejemplo 2, el tipo base type xsd:string queda limitado por una expresión regular.

Ejemplo 2

```
<xsd:simpleType name="SKU">
   < xsd:restriction base="xsd:string">
    < xsd:pattern value="\d{3}-[A-Z]{2}"/>
   </xsd:restriction>
  </xsd:simpleType>
```

El Ejemplo 3 muestra un tipo base enumerado en el tipo incorporado serie.

Ejemplo 3

```
<xsd:simpleType name="SchoolClass">
   < xsd:restriction base="xsd:string">
    < xsd:enumeration value="WI"/>
    < xsd:enumeration value="MI"/>
    < xsd:enumeration value="II"/>
    < xsd:enumeration value="DI"/>
    < xsd:enumeration value="AI"/>
   </xsd:restriction>
  </xsd:simpleType>
```

Elementos en esquemas XML

Para declarar un elemento en un esquema XML debe indicar el nombre y el tipo como atributo del elemento. Por ejemplo:

```
<xsd:element name="street" type="xsd:string"/>
```

Además, puede utilizar los atributos minOccurs y maxOccurs para determinar el número máximo y mínimo de veces que debe aparecer el elemento en el documento XML. El valor por omisión de minOccurs y maxOccurs es 1.

Atributos en esquemas XML

Las declaraciones de atributos aparecen al final de una definición de elemento. Por ejemplo:

```
<xsd:complexType name="PurchaseOrderType">
< xsd:sequence>
  < xsd:element name="billTo" type="USAddress"/>
< xsd:sequence>
  < xsd:attribute name="orderDate" type="xsd:date"/>
 </xsd:complexType>
```

Conceptos relacionados:

- "Ventajas de la utilización de esquemas XML en lugar de DTD" en la página 121

Tareas relacionadas:

- "Funciones de validación" en la página 166

Información relacionada:

- "Ejemplo de un esquema XML" en la página 123
- "Elemento complexType del esquema XML" en la página 121

Ejemplo de un esquema XML

Una buena estrategia es desarrollar esquemas XML diseñando en primer lugar la estructura de datos del documento XML mediante una herramienta UML. Después de diseñar la estructura, puede correlacionarla con el documento del esquema. El ejemplo siguiente muestra un esquema XML.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
 3
 4   <xs:element name="personnel">
 5    <xs:complexType>
 6     <xs:sequence>
 7       <xs:element ref="person" minOccurs='1' maxOccurs='unbounded'/>
 8    </xs:sequence>
 9   </xs:complexType>
10  </xs:element>
11
12   <xs:element name="person">
13    <xs:complexType>
14     <xs:sequence>
15       <xs:element ref="name"/>
16       <xs:element ref="email" minOccurs='0' maxOccurs='4'/>
17    </xs:sequence>
18     <xs:attribute name="id"  type="xs:ID" use='required'/>
19   </xs:complexType>
20  </xs:element>
21
22   <xs:element name="name">
23    <xs:complexType>
24     <xs:sequence>
25      <xs:element ref="family"/>
26      <xs:element ref="given"/>
27    </xs:sequence>
28   </xs:complexType>
29  </xs:element>
30
31   <xs:element name="family" type='xs:string'/>
32   <xs:element name="given" type='xs:string'/>
33   <xs:element name="email" type='xs:string'/>
34 </xs:schema>
```

Las primeras dos líneas declaran que este esquema XML es compatible con XML 1.0 y decodificado con Unicode, y especifican el uso del espacio de nombres estándar del esquema XML, que permite acceder a los tipos de datos y estructuras del esquema XML.

Las líneas 4 a 10 definen el personal como un elemento complexType que consta de una secuencia de 1 a n personas. A continuación, se define complexType en las líneas 12 a 20. Consiste del nombre del elemento complexType y del correo electrónico del elemento. El elemento de correo electrónico (email) es opcional (minOccurs = '0') y puede aparecer hasta cuatro veces (maxOccurs = '4'). Cuanto mayor sea el número de apariciones de un elemento, más se tardará en validar el esquema. En cambio, en una DTD sólo puede elegir 0, 1 o un número ilimitado de apariciones de un elemento.

Las líneas 22 a 29 definen el tipo de nombre que se utiliza para el tipo de persona. El tipo de nombre consta de una secuencia de una familia y un elemento determinado.

Las líneas 31 a 33 definen los elementos individuales *family, given* y *e-mail*, que contienen las series de tipo que se han declarado.

Instancia de un documento XML mediante el uso del esquema

El siguiente ejemplo muestra un documento XML que es una instancia del esquema *personalnr.xsd*.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:noNamespaceSchemaLocation='personsnr.xsd'>
4
5   <person id="Big.Boss" >
6      <name><family>Boss</family> <given>Big</given></name>
7     <email>chief@foo.com</email>
8   </person>
9
10   <person id="one.worker">
11   <name><family>Worker</family><given>One</given></name>
12      <email>one@foo.com</email>
13   </person>
14
15    <person id="two.worker">
16   <name><family>Worker</family><given>Two</given></name>
17      <email>two@foo.com</email>
18   </person>
19 </personnel>
```

Instancia de un documento XML mediante el uso de una DTD

Este ejemplo muestra cómo se efectuaría este esquema XML como una DTD.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!ELEMENT email (#PCDATA)>
3   <!ELEMENT family (#PCDATA)>
4   <!ELEMENT given (#PCDATA)>
5   <!ELEMENT name (family, given)>
6   <!ELEMENT person (name, email*)>
7
8   <!ATTLIST person
9   id ID #REQUIRED>
10 <!ELEMENT personnel (person+)>
```

Mediante una DTD se puede establecer el número máximo de apariciones de correo electrónico a 1 o apariciones ilimitadas.

Mediante esta DTD, la instancia de documento XML sería la misma que la que del ejemplo anterior, excepto por la línea 2 que sería:

```
<!DOCTYPE personnel SYSTEM "personsnr.dtd">
```

Conceptos relacionados:

- “Ventajas de la utilización de esquemas XML en lugar de DTD” en la página 121

Tareas relacionadas:

- “Tipos de datos, elementos y atributos en esquemas” en la página 122
- “Funciones de validación” en la página 166

Información relacionada:

- “Elemento complexType del esquema XML” en la página 121

Parte 4. Consulta

Esta parte proporciona información de sintaxis para el mandato de administración de XML Extender, tipos de datos definidos por el usuario (UDT), funciones definidas por el usuario (UDF) y procedimientos almacenados. También se proporciona texto de mensaje para actividades de determinación de problemas.

Capítulo 6. Mandato de administración dxxadm

Visión general de dxxadm

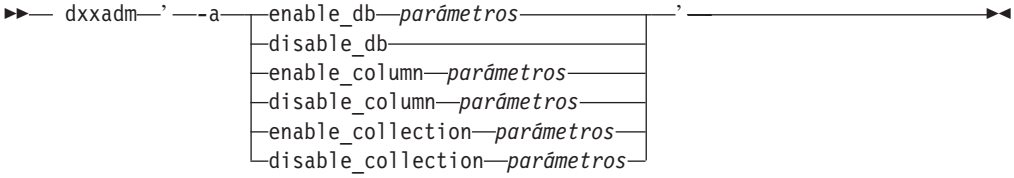
XML Extender proporciona un mandato de administración, **dxxadm**, para realizar las siguientes tareas de administración:

- enable_column
- enable collection
- enable_db
- disable_column
- disable_collection
- disable_db

Conceptos relacionados:

- "Herramientas de administración para XML Extender" en la página 37
- "Visión general de la administración de XML Extender" en la página 39

Sintaxis del mandato de administración dxxadm



Parámetros:

Tabla 15. Parámetros de dxxadm

Parámetro	Descripción
enable_db	Habilita las características de XML Extender para una base de datos.
disable_db	Inhabilita las características de XML Extender para una base de datos.
enable_column	Habilita una columna XML para poder almacenar los documentos XML en la columna.
disable_column	Inhabilita la columna habilitada para XML.
enable_collection	Habilita una colección XML de acuerdo a la DAD especificada.
enable_collection	Inhabilita una colección habilitada para XML.

Opciones del mandato de administración

Las siguientes opciones del mandato **dxxadm** están disponibles para los programadores del sistema:

- enable_column
- enable_collection

- enable_db
- disable_column
- disable_collection
- disable_db

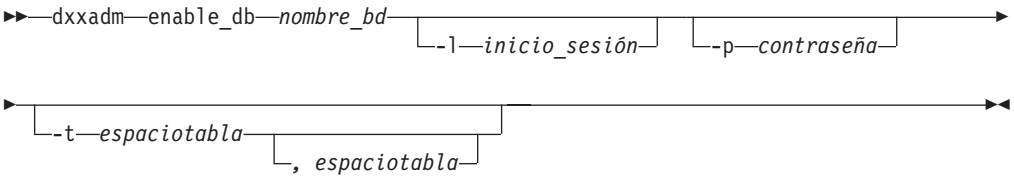
Opción enable_db del mandato dxxadm

Propósito:

Habilita características de XML Extender para una base de datos. Cuando se habilita la base de datos, XML Extender crea los objetos siguientes:

- Los tipos definidos por el usuario (UDT) de XML Extender.
- Las funciones definidas por el usuario (UDF) de XML Extender.
- La tabla de depósito de DTD de XML Extender, DTD_REF, que almacena las DTD e información acerca de cada DTD.
- La tabla de uso de XML Extender, XML_USAGE, que almacena información común para cada columna que está habilitada para XML y para cada colección.

Sintaxis:



Parámetros:

Tabla 16. Parámetros de enable_db

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos en la que residen los datos XML.
-l *inicio_sesión*	ID de usuario opcional, utilizado para conectarse a la base de datos Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Contraseña opcional utilizada para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza la contraseña actual.
-t *espaciotabla*	Nombre opcional del espacio de tabla existente para albergar tablas db2xml.XML_USAGE y db2xml.DTD_REF. También puede especificar un segundo espacio de tabla.

Si está utilizando DB2 UDB Enterprise Server Edition particionado y desea especificar un espacio de tabla mientras se está habilitando la base de datos, necesitará haber especificado un grupo de nodos al crear el espacio de tabla. Por ejemplo:

```
db2 "create database partition group mygroup on node (0,1)"
db2 "create regular tablespace mitb in database partition group mygroup
   managed by system using ('mitb')"
```

En el ejemplo anterior, se especificaría el espacio de tabla `mitb` mientras se habilita la base de datos.

Si no se proporciona ninguna opción de espacio de tabla al habilitar la base de datos, XML Extender comprobará si existen los espacios de tabla DXXDTDRF y DXXXMLUS. Se creará la tabla `db2xml.dtd_ref` en el espacio de tabla DXXDTDRF si el espacio de tabla ya existe, y se creará la tabla `db2xml.xml_usage` en el espacio de tabla DXXXMLUS. Si uno de los espacios de tabla, DXXDTDRF o DXXXMLUS, no existe, se creará la respectiva tabla (`db2xml.dtd_ref` o `db2xml.xml_usage`) en el espacio de tabla más apropiado.

Si sólo se proporciona un espacio de tabla DXXDTDRF al habilitar la base de datos, se crearán ambas tablas en el espacio de tabla especificado. Si se proporcionan dos espacios de tabla cuando se habilita la base de datos, se creará la tabla `db2xml.dtd_ref` en el primer espacio de tabla listado y la tabla `db2xml.xml_usage` en el segundo espacio de tabla listado.

Ejemplo::

El ejemplo siguiente habilita la base de datos SALES_DB.

```
dxxadm enable_db SALES_DB
```

Información relacionada:

- "Visión general de dxxadm" en la página 129

Opción disable_db del mandato dxxadm

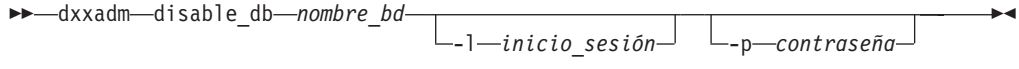
Propósito:

Inhabilita las características de XML Extender para una base de datos. Esta acción se denomina "inhabilitar una base de datos". Cuando la base de datos está inhabilitada, ya no puede ser utilizada por XML Extender. Cuando XML Extender inhabilita la base de datos, elimina los objetos siguientes:

- Los tipos definidos por el usuario (UDT) de XML Extender.
- Las funciones definidas por el usuario (UDF) de XML Extender.
- La tabla de depósito de DTD de XML Extender, DTD_REF, que almacena las DTD e información acerca de cada DTD.
- La tabla de uso de XML Extender, XML_USAGE, que almacena información común para cada columna que está habilitada para XML y para cada colección.

Importante: Debe inhabilitar todas las columnas XML antes de intentar inhabilitar una base de datos. XML Extender no puede inhabilitar una base de datos que contiene columnas o colecciones que están habilitadas para XML. También debe descartar todas las tablas que tienen columnas definidas con tipos definidos por el usuario de XML Extender como, por ejemplo, XMLCLOB.

Sintaxis:



Parámetros:

Tabla 17. Parámetros de disable_db

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos en la que residen los datos XML.
-l *inicio_sesión*	Es el ID de usuario que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplo::
El ejemplo siguiente inhabilita la base de datos SALES_DB.

```
dxxadm disable_db SALES_DB
```

Conceptos relacionados:

- "Procedimientos almacenados de administración de XML Extender - Visión general" en la página 202
- Capítulo 13, "Tablas de soporte de administración de XML Extender", en la página 279

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix

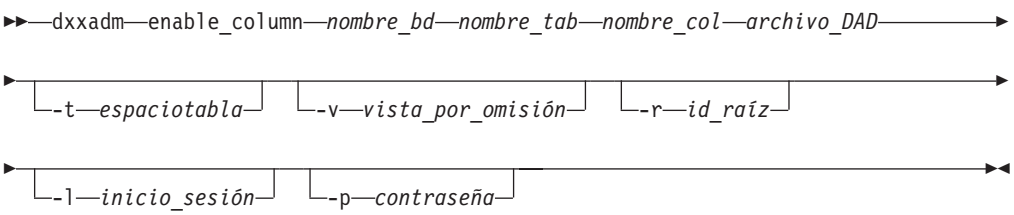
Opción enable_column del mandato dxxadm

Propósito:

Conecta a una base de datos y habilita una columna XML para que pueda contener los UDT de XML Extender. Cuando habilita una columna, XML Extender realiza las siguientes tareas:

- Determina si la tabla XML tiene una clave primaria; si no la tiene, XML Extender modifica la tabla XML y añade una columna denominada 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 especificado por XML Extender.
- Opcionalmente, crea una vista por omisión para la tabla XML y las tablas auxiliares, utilizando opcionalmente un nombre especificado por el usuario.

Sintaxis:



Parámetros:

Tabla 18. Parámetros de enable_column

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos en la que 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.
-t *espaciotabla*	Es el espacio de tabla que contiene las tablas auxiliares asociadas a la columna XML. Si no se especifica este parámetro, se utiliza el espacio de tabla por omisión.
-v *vista_por_omisión*	Es el nombre de la vista por omisión que une 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 *inicio_sesión*	Es el ID de usuario que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza la contraseña actual.

Si está utilizando DB2 UDB Enterprise Server Edition particionado y desea especificar un espacio de tabla mientras se está habilitando una columna, necesitará haber especificado un grupo de nodos al crear el espacio de tabla. Por ejemplo:

```
db2 "create database partition group mygroup on node (0,1)"
db2 "create regular tablespace mitb in database partition group mygroup
   managed by system using ('mitb')"
```

En el ejemplo anterior, se especificaría el espacio de tabla `mitb` al habilitar la columna:

```
dxxadm enable_column mydatabase mytable mycolumn "dad/mydad.dad" -t mitb
```

Ejemplo::

El ejemplo siguiente habilita una columna XML.

```
dxxadm enable_column SALES_DB SALES_TAB ORDER getstart.dad
   -v sales_order_view -r INVOICE_NUMBER
```

Ejemplos relacionados:

- "dxx_xml -- s-getstart_enableCol_NT-cmd.htm"
- "dxx_xml -- s-getstart_enableCol-cmd.htm"

Opción disable_column del mandato dxxadm

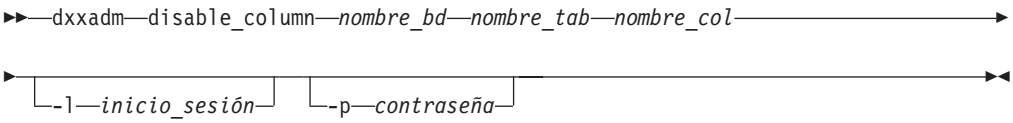
Propósito:

Conecta a una base de datos e inhabilita la columna habilitada para XML. Cuando se inhabilita la columna, 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 activadores 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 la columna XML, XML Extender conserva ambas tablas auxiliares que creó y también la entrada de la tabla XML_USAGE correspondiente a la columna XML.

Sintaxis:



Parámetros:

Tabla 19. Parámetros de disable_column

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos en la que 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 *inicio_sesión*	Es el ID de usuario que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña que se utiliza para conectarse a la base de datos. 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
```

Información relacionada:

- "Opción enable_collection del mandato dxxadm" en la página 135

Ejemplos relacionados:

- "dxx_xml -- s-getstart_clean_NT-cmd.htm"
- "dxx_xml -- s-getstart_clean-cmd.htm"

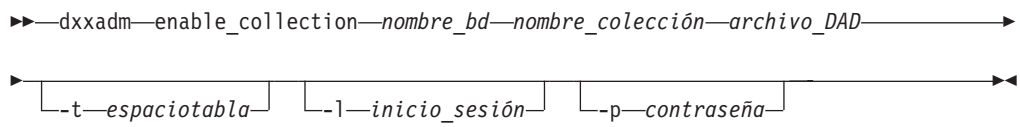
Opción enable_collection del mandato dxxadm

Propósito:

Conecta a una base de datos y habilita una colección XML de acuerdo con la DAD especificada. Al ejecutar XML Extender en un entorno particionado Enterprise Server Edition, compruebe que todas las tablas especificadas en el archivo DAD incluyen al menos una columna que puede ser una clave de particionamiento. Cuando se habilita una colección, XML Extender realiza las siguientes tareas:

- Crea una entrada en la tabla XML_USAGE para la colección XML.
- Para la correlación de RDB_node, crea tablas de colección especificadas en la DAD, si las tablas no existen en la base de datos.

Sintaxis:



Parámetros:

Tabla 20. Parámetros de enable_collection

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos en la que 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 *espaciotabla*	Es el nombre del espacio de tabla que está asociado a la colección. Si no se especifica este parámetro, se utiliza el espacio de tabla por omisión.
-l *inicio_sesión*	Es el ID de usuario que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza la contraseña actual.

Si está utilizando DB2 UDB Enterprise Server Edition particionado y desea especificar un espacio de tabla mientras se habilita una colección, necesitará haber especificado un grupo de nodos al crear el espacio de tabla. Por ejemplo:

```
db2 "create database partition group mygroup on node (0,1)"
db2 "create regular tablespace mitb in database partition group mygroup
   managed by system using ('mitb')"
```

En el ejemplo anterior, se especificaría la tabla `mitb` al inhabilitar la colección.

Ejemplos:

El ejemplo siguiente habilita una colección XML.

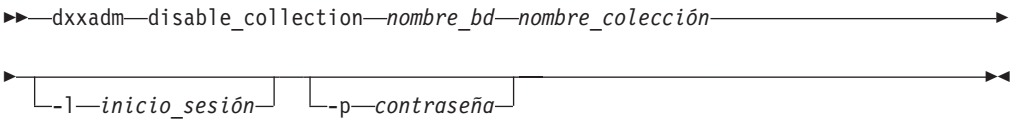
```
dxxadm enable_collection SALES_DB sales_ord
getstart_xcollection.dad -t orderspace
```

Opción disable_collection del mandato dxxadm

Propósito:

Conecta a una base de datos e inhabilita la colección habilitada para XML. El nombre de la colección ya no puede utilizarse en los procedimientos almacenados de composición (dxxRetrieveXML) y descomposición (dxxInsertXML) de documentos. Cuando se inhabilita una colección XML, se suprime la entrada de la tabla XML_USAGE correspondiente a la colección. Si se inhabilita la colección, no se eliminan las tablas de colección que se crean cuando se utiliza la opción enable_collection.

Sintaxis:



Parámetros:

Tabla 21. Parámetros de disable_collection

Parámetro	Descripción
nombre_bd	Es el nombre de la base de datos en la que residen los datos.
nombre_colección	Es el nombre de la colección XML.
-l *inicio_sesión*	Es el ID de usuario que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza el ID de usuario actual.
-p *contraseña*	Es la contraseña que se utiliza para conectarse a la base de datos. Si no se especifica este parámetro, se utiliza la contraseña actual.

Ejemplo::
El ejemplo siguiente inhabilita una colección XML.

```
dxxadm disable_collection SALES_DB sales_ord
```

Capítulo 7. Tipos definidos por el usuario de XML Extender

Los tipos definidos por el usuario (UDT) son tipos de datos creados por una aplicación o herramienta de DB2®. XML Extender crea los siguientes tipos definidos por el usuario para su uso con columnas XML:

- XMLVARCHAR
- XMLCLOB
- XMLFILE

Los tipos de datos se utilizan para definir la columna en la tabla de aplicación que se empleará para almacenar el documento XML. También es posible almacenar documentos XML como archivos en el sistema de archivos especificando un nombre de archivo.

Todos los tipos definidos por el usuario de XML Extender tienen el calificador **DB2XML**, que es el *nombre de esquema* de los tipos definidos por el usuario de DB2 UDB XML Extender. Por ejemplo:

```
db2xml.XMLVarchar
```

XML Extender crea los UDT para almacenar y recuperar documentos XML. La Tabla 22 describe los UDT.

Tabla 22. Los UDT de 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, XML Extender almacena el documento XML en un archivo de servidor externo. Text Extender no se puede habilitar con XMLFILE. Corresponde al usuario asegurar la integridad entre el contenido del archivo, DB2 y la tabla auxiliar creada para el indexado.

long_varchar y *long_clob* son específicos del sistema operativo.

Para XML Extender en DB2 UDB en los sistemas operativos Linux, Unix y Windows®, *varchar_len* = 3K y *clob_len* = 2G.

Para cambiar el tamaño de un UDT XMLVARCHAR o XMLCLOB, cree el UDT antes de habilitar la base de datos para XML Extender.

Procedimiento:

Para cambiar el tamaño de un UDT XMLVARCHAR o XMLCLOB UDT de una base de datos habilitada:

1. Realice una copia de seguridad de todos los datos en la base de datos habilitada para XML Extender.
2. Descarte todas las tablas de colección XML o tablas auxiliares de columna XML.
3. Inhabilite la base de datos mediante el mandato `dxxadm disable_db`.
4. Cree el tipo definido por el usuario XMLVARCHAR o XMLCLOB.
5. Habilite la base de datos con el mandato `dxxadm enable_db`.
6. Vuelva a crear y a cargar las tablas.

Estos UDT sólo se utilizan para especificar los tipos de columnas de aplicación; no se aplican a las tablas auxiliares que XML Extender crea.

Conceptos relacionados:

- "Columnas XML como método de almacenamiento y acceso" en la página 76
- "Colecciones XML como método de almacenamiento y acceso" en la página 93
- "Preparación para administrar XML Extender" en la página 37
- "Esquemas de correlación para colecciones XML" en la página 105

Capítulo 8. Funciones definidas por el usuario de XML Extender

Una función definida por el usuario (UDF) es una función que se define ante el sistema de gestión de bases de datos y a la que se puede hacer referencia en sentencias de SQL. Esta sección describe las funciones definidas por el usuario utilizadas por DB2 UDB XML Extender.

Tipos de funciones definidas por el usuario de XML Extender

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) XML se utilizan con las columnas XML, pero no con las colecciones XML.

Todas las UDF tienen el nombre de esquema de DB2XML.

Los tipos de funciones de XML Extender se describen en la lista siguiente:

funciones de almacenamiento
Las funciones de almacenamiento insertan documentos XML intactos en columnas habilitadas para XML como tipos de datos XML.

funciones de recuperación
Las funciones de recuperación recuperan documentos XML a partir de columnas XML de una base de datos DB2®.

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. XML Extender proporciona un conjunto de funciones de extracción para diversos tipos de datos de SQL.

función de actualización
La función de actualización modifica el contenido de todo un documento XML o de un elemento especificado o valores de atributos y devuelve una copia de un documento XML con un valor actualizado, que está especificado por la vía de ubicación.

Las funciones definidas por el usuario XML le permiten realizar búsquedas en tipos de datos de SQL en general. Además, puede utilizar DB2 UDB Net Search Extender con XML Extender para realizar *búsquedas de texto completo* y estructural en el texto de documentos XML. Esta capacidad de búsqueda se puede utilizar, por ejemplo, para mejorar las posibilidades de utilización de un sitio Web que publique grandes volúmenes de texto legible tales como artículos de prensa o aplicaciones *EDI (Electronic Data Interchange - Intercambio electrónico de datos)*, que contienen elementos o atributos de búsqueda frecuente.

Restricción: Al utilizar los marcadores de parámetros en las UDF, una restricción de base de datos de Java™ (JDBC) obliga a que el marcador de parámetro para la UDF se convierta al tipo de datos de la columna en la que se insertarán los datos devueltos.

Funciones de almacenamiento

Visión general de las funciones de almacenamiento en XML Extender

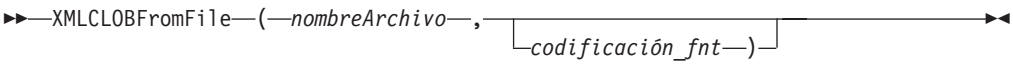
Utilice las funciones de almacenamiento para insertar documentos XML en una base de datos XML. Puede utilizar las funciones por omisión de conversión de datos de un UDT directamente en las sentencias INSERT o SELECT. Además, XML Extender proporciona las UDF para obtener documentos de fuentes distintas del tipo de datos base del UDT y convertirlas al UDT especificado.

Función 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.
codificación_fnt	VARCHAR(100)	La codificación del archivo fuente.

Resultados:
XMLCLOB como LOCATOR
Ejemplo:
El siguiente ejemplo lee un documento XML desde un archivo en un servidor y lo inserta en una columna XML como un tipo XMLCLOB. La codificación del archivo del servidor se especifica explícitamente como iso-8859-1.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLCLOBFromFile('instalación_dxx/samples/db2xml
           /xml/getstart.xml
           ', 'iso-8859-1'))
```

donde *instalación_dxx* es el directorio donde XML Extender se encuentra instalado.

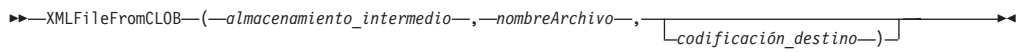
La columna ORDER de la tabla SALES_TAB se ha definido con el tipo XMLCLOB.

Función XMLFileFromCLOB()

Propósito:

Lee un documento XML como un localizador de CLOB, lo graba 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 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.
codificación_destino	VARCHAR(100)	La codificación del archivo de salida.

Resultados:

XMLFILE

Ejemplo:

El ejemplo siguiente lee un documento XML como localizador de CLOB (variable del lenguaje principal con un valor que representa un valor LOB individual del servidor de bases de datos), lo graba 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. La función codificará el archivo de salida en ibm-808.

```
EXEC SQL BEGIN DECLARE SECTION;
             SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLFileFromCLOB(:xml_buf, 'instalación_dxx/samples/db2xml
           /xml/getstart.xml', 'ibm-808'))
```

donde *instalación_dxx* es el directorio donde XML Extender se encuentra instalado.

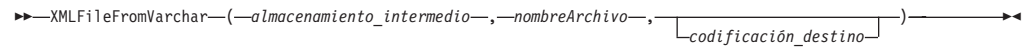
La columna ORDER de la tabla SALES_TAB se ha definido con el tipo XMLFILE. Si tiene un documento XML en el almacenamiento intermedio, puede almacenarlo en un archivo de servidor.

Función XMLFileFromVarchar()

Propósito:

Lee un documento XML de la memoria como VARCHAR, lo graba 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 XMLFileFromVarchar

Parámetro	Tipo de datos	Descripción
almacenamiento_intermedio	VARCHAR(3K)	Es el almacenamiento intermedio donde reside el documento XML.
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.
codificación_destino	VARCHAR(100)	La codificación del archivo de salida.

Resultados:

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. La función codificará el archivo de salida en iso-8859-1.

```
EXEC SQL BEGIN DECLARE SECTION;
      struct { short len; char data[3000]; } xml_buff;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
        XMLFileFromVarchar(:xml_buf, 'instalación_dxx/samples/db2xml
           /xml/getstart.xml', 'iso-8859-1'))
```

donde *instalación_dxx* es el directorio donde XML Extender se encuentra instalado.

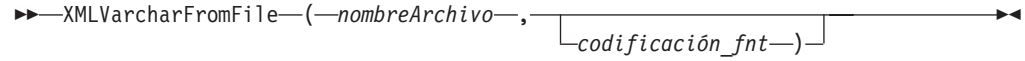
La columna ORDER de la tabla SALES_TAB se ha definido con el tipo XMLFILE.

Función XMLVarcharFromFile()

Propósito:

Lee un documento XML de un archivo de servidor y devuelve el documento como un tipo XMLVARCHAR.

Sintaxis:



Parámetros:

Tabla 26. Parámetro de XMLVarcharFromFile

Parámetro	Tipo de datos	Descripción
nombreArchivo	VARCHAR(512)	Es el nombre completamente calificado del archivo de servidor.
codificación_fnt	VARCHAR(100)	La codificación del archivo fuente.

Resultados:

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. La codificación del archivo del servidor se especifica explícitamente como ibm-808.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLVarcharFromFile('instalación_dxx/samples/db2xml
           /xml/getstart.xml', 'ibm-808'))
```

donde *instalación_dxx* es el directorio donde XML Extender se encuentra instalado.

Este ejemplo inserta un registro en la tabla SALES_TAB. La función XMLVarcharFromFile() importa el documento XML desde un archivo especificado explícitamente como codificado en ibm-808 en DB2 UDB y lo almacena como un XMLVARCHAR.

Funciones de recuperación

Funciones de recuperación en XML Extender

XML Extender proporciona una función sobrecargada Content(), que se utiliza para la recuperación. Esta función sobrecargada es un conjunto de funciones de recuperación que tienen el mismo nombre, pero que se comportan de forma diferente según la ubicación de los datos que se recuperan. También puede utilizar las funciones de conversión de datos por omisión para convertir un UDT de XML al tipo de datos base.

Las funciones Content() proporcionan los tipos siguientes de recuperación:

- **Recuperación desde el almacenamiento externo en el servidor a una variable del lenguaje principal en el cliente.**

  Puede utilizar Content() para recuperar un documento XML que está almacenado como un archivo de servidor externo y colocarlo en una memoria intermedia. Para ello, puede utilizar la función Content(): recuperación de XMLFILE y almacenamiento en CLOB.
- **Recuperación desde el almacenamiento interno en un archivo de servidor externo**

  También puede utilizar Content() para recuperar el contenido de un documento XML almacenado en el interior de DB2 UDB y almacenarlo en un archivo de servidor en el sistema de archivos de DB2 UDB. Las siguientes funciones de Content() se utilizan para almacenar información en archivos de servidor externos:

- Content(): recuperación de XMLVARCHAR y almacenamiento en un archivo de servidor externo
- Content(): recuperación de XMLCLOB y almacenamiento en un archivo de servidor externo

Las siguientes funciones definidas por el usuario tienen un nuevo parámetro que especifica la codificación del archivo fuente o de salida. El valor de este parámetro es cualquier nombre de página de códigos reconocido por el estándar de International Components for Unicode.

```
db2xml.XMLVarcharFromFile(filename varchar(512), src_encoding  varchar(100))
returns XMLVarchar

db2xml.XMLCLOBFromFile(filename varchar(512), src_encoding varchar(100))
returns XMLCLOB AS LOCATOR

db2xml.XMLFileFromVarchar(doc varchar(3000), targetfilename varchar(512),
targetencoding varchar(100))
returns XMLFile

db2xml.XMLFileFromCLOB(doc CLOB(2G) as LOCATOR, targetfilename varchar(512),
 targetencoding varchar(100))
returns XMLFile

db2xml.Content(doc XMLVarchar, targetfilename varchar(512),
 targetencoding varchar(100))
returns varchar(512)

db2xml.Content(doc XMLCLOB as LOCATOR, targetfilename varchar(512),
 targetencoding varchar(100))
returns varchar(512)
```

Ejemplos:

Para importar el contenido de un archivo `/home/collins/xml/entail.xml` en un almacenamiento intermedio varchar y para especificar que el archivo fuente está codificado en iso-8859-1:

```
db2xml.XMLVarcharFromFile('/home/collins/xml/entail.xml', 'iso-8859-1')
```

El archivo se importa en un tipo varchar y se convierte de la página de códigos iso-8859-1 a la página de códigos de la base de datos.

Para exportar un almacenamiento intermedio de tipo varchar en un archivo /home/raskolnikov/xml/confession.xml y para especificar que el archivo de salida debe codificarse en ibm-808:

```
db2xml.Content('<sequence><thought>I did it!</thought></sequence>',
'/home/raskolnikov/xml/confession.xml', 'ibm-808')
```

El contenido del almacenamiento intermedio se exporta a un archivo y se convierte desde la página de códigos de la base de datos a ibm-808. La declaración de codificación del archivo XML se actualiza entonces de forma correspondiente.

Los ejemplos en la siguiente sección dan por supuesto que el usuario está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir “DB2” antes de cada mandato.

Content(): recuperación de XMLFILE y almacenamiento en CLOB

Propósito:

Recupera datos de un archivo de servidor y los almacena en un localizador de CLOB.

Sintaxis:

Parámetros:

Tabla 27. Parámetro de XMLFILE a CLOB

Parámetro	Tipo de datos	Descripción
objxml	XMLFILE	Es el documento XML.

Resultados:
CLOB (*long_clob*) como LOCATOR

clob_len para DB2 UDB es 2G.

Ejemplo:

En el ejemplo siguiente se recuperan datos de un archivo de servidor y se almacenan en un localizador de CLOB.

```
char    subsystem[20];
long   retcode = 0, reason = 0;
extern "OS" { int DSNALI(char * functn, ...); }

extern "OS" short DSNTIAR(struct sqlca *sqlca,
                          error_struct *error_message,
                          long *data_len);

EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS CLOB_LOCATOR xml_buff;
EXEC SQL END DECLARE SECTION;

/* Vincularse al subsistema */
rc = DSNALI("OPEN        ", subsystem, "PLANNAME",
     &retcode, &reason);
if ( retcode != 0 )
{
   /* imprimir mensaje de error */
      goto exit;
}

EXEC SQL DECLARE c1 CURSOR FOR

     SELECT Content(order) from sales_tab
     WHERE sales_person = 'Sriram Srinivasan'

EXEC SQL OPEN c1;

do {
   EXEC SQL FETCH c1 INTO :xml_buff;
   if (SQLCODE != 0) {
      break;
      }
```

```
   else {
     /* se ocupa del doc XML en almacenamiento intermedio */
     }
   }

EXEC SQL CLOSE c1;

/* Desvincularse del subsistema */
DSNALI("CLOSE       ", "SYNC", &retcode, &reason);
if ( retcode != 0 ) {
   /* imprimir mensaje de error */
}
```

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.

Tareas relacionadas:

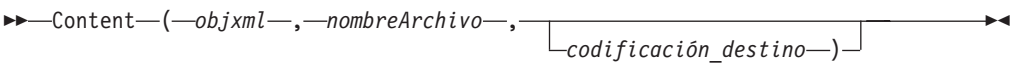
- "Actualización y supresión de datos en colecciones XML" en la página 102

Content(): recuperación de XMLVARCHAR y almacenamiento en un 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 de contenido alterará temporalmente su contenido.

Parámetros:

Tabla 28. 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.
codificación_destino	VARCHAR(100)	La codificación del archivo de salida.

Resultados:

VARCHAR(512)

Ejemplo:

El siguiente ejemplo recupera el contenido XML almacenado como un tipo XMLVARCHAR y lo almacena en un archivo externo que se encuentra en el servidor. La UDF codifica el documento en 'ibm-808'.

```
CREATE table app1 (id int NOT NULL, order DB2XML.XMLVarchar);
INSERT into app1 values (1, '<?xml version="1.0"?>
   <!DOCTYPE SYSTEM
"instalación_dxx/samples/db2xml/dtd/getstart.dtd">

    <Order key="1">
        <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
      </Part>
    </Order>');

SELECT DB2XML.Content(order,
'instalación_dxx/samples/dad/getstart_column.dad'
, 'ibm-808')
  from app1 where ID=1;
```

Tareas relacionadas:

- "Método para recuperar un documento XML" en la página 81

Información relacionada:

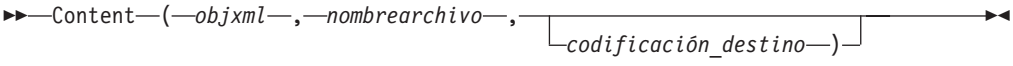
- "Funciones de recuperación en XML Extender" en la página 143

Content(): recuperación de XMLCLOB y almacenamiento en un 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 de contenido alterará temporalmente su contenido.

Parámetros:

Tabla 29. Parámetros de XMLCLOB en el archivo del 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.
codificación_destino	VARCHAR(100)	La codificación del archivo de salida.

Resultados:

VARCHAR(512)

Ejemplo:

El siguiente ejemplo recupera el contenido XML almacenado como un tipo XMLCLOB y lo almacena en un archivo externo que se encuentra en el servidor. La UDF codifica el documento en 'ibm-808'.

```
CREATE table app1 (id int NOT NULL, order DB2XML.XMLCLOB not logged);

INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM
"instalación_dxx/samples/db2xml/dtd/getstart.dtd"
->
    <Order key="1">
      <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
      </Part>
    </Order>');

SELECT DB2XML.Content(order,
'instalación_dxx/samples/db2xml/xml/getstart.xml', 'ibm-808')
 from app1 where ID=1;
```

Funciones de extracción

Extracción de funciones en XML Extender

Las funciones de extracción extraen el contenido del elemento o el valor de atributo de un documento XML y devuelven los tipos de datos SQL solicitados. XML Extender proporciona un conjunto 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 de XML Extender, que puede ser uno de los UDT de XML. El segundo parámetro es la vía de ubicación, que especifica el elemento o el atributo de XML. Cada función de extracción devuelve el valor o el contenido especificado por la vía de ubicación.

Dado que algunos valores de elemento o atributo aparecen varias veces, las funciones de extracción devuelven un valor escalar o un valor de tabla; al primero se le denomina función escalar y al segundo se le denomina función de tabla.

Los ejemplos asumen que se está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" antes de cada mandato.

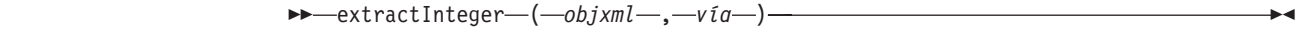
extractInteger() y extractIntegers()

Propósito:

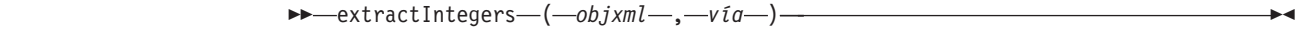
Extrae el contenido de un elemento o el valor de un atributo en un documento XML y devuelve datos de tipo INTEGER.

Sintaxis:

Función escalar:



Función de tabla:



Parámetros:

Tabla 30. Parámetros de la función 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 devuelto:
INTEGER
Códigos de retorno:
returnedInteger
Ejemplos:
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 INTEGER (ENTERO). Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" antes de cada mandato.

```
CREATE TABLE t1(key INT);
INSERT INTO t1 values (
    DB2XML.extractInteger(DB2XML.XMLFile('/samples/db2xml
```

```
    /xml/getstart.xml
    '),
    '/Order/Part[@color="black "]/key'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En el ejemplo siguiente, cada clave de pedido para los pedidos de ventas se extrae como INTEGER. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
SELECT *
  FROM TABLE(
  DB2XML.extractIntegers(DB2XML.XMLFile('/samples/db2xml/xml/getstart.xml'),
      '/Order/Part/key'))   AS X;
```

Conceptos relacionados:

- Apéndice D, "Nombres de UDT y UDF para XML Extender", en la página 325
- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148

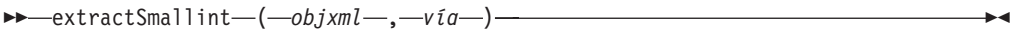
extractSmallint() y extractSmallints()

Propósito:

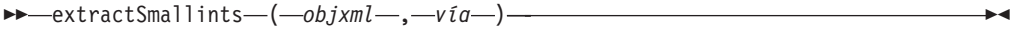
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo SMALLINT.

Sintaxis:

Función escalar:



Función de tabla:



Parámetros:

Tabla 31. Parámetros de la función 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 devuelto:
SMALLINT
Códigos de retorno:
returnedSmallint

Ejemplos:

Ejemplo de función escalar:

En el ejemplo siguiente, el valor de clave en todos los pedidos de ventas se extrae como SMALLINT. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
CREATE TABLE t1(key INT);
INSERT INTO t1 values (
    DB2XML.extractSmallint(db2xml.xmlfile('instalación_dxx
     /samples/db2xml/xml/getstart.xml'),
    '/Order/Part[@color="black "]/key'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En el ejemplo siguiente, el valor de clave en todos los pedidos de ventas se extrae como SMALLINT. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
SELECT *
  FROM TABLE(
    DB2XML.extractSmallints(DB2XML.XMLFile('instalación_dxx
      /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/key'))   AS X;
```

Conceptos relacionados:

- "Utilización de índices para datos de columna XML" en la página 77
- Apéndice D, "Nombres de UDT y UDF para XML Extender", en la página 325
- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148
- "Códigos de retorno de los procedimientos almacenados de XML Extender" en la página 283

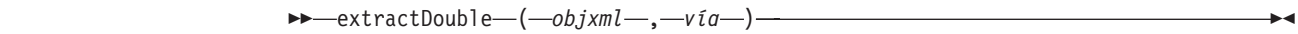
extractDouble() y extractDoubles()

Propósito:

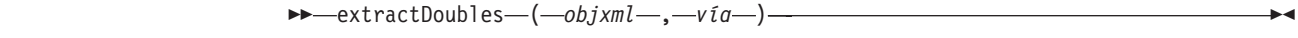
Extrae el contenido de un elemento o valor de un atributo en un documento XML y devuelve datos de tipo DOUBLE.

Sintaxis:

Función escalar:



Función de tabla:



Parámetros:

Tabla 32. Parámetros de la función 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 devuelto:
DOUBLE
Códigos de retorno:
returnedDouble
Ejemplos: Ejemplo de función escalar:
El ejemplo siguiente convierte automáticamente el precio de un pedido de un tipo DOUBLE a DECIMAL. Los ejemplos dan por supuesto que se utiliza el shell de mandatos de DB2, en el que no es es necesario escribir "DB2" al principio de cada mandato.

```
CREATE TABLE t1(price DECIMAL(9,2));
INSERT INTO t1 values (
    DB2XML.extractDouble(DB2XML.xmlfile('instalación_dxx
    /samples/db2xml/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. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir DB2 UDB antes de cada mandato.

```
SELECT CAST(RETURNEDDOUBLE AS DOUBLE)
  FROM TABLE(
     DB2XML.extractDoubles(DB2XML.XMLFile('instalación_dxx
       /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/ExtendedPrice'))   AS X;
```

Conceptos relacionados:

- Apéndice D, "Nombres de UDT y UDF para XML Extender", en la página 325

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148

extractReal() y extractReals()

Propósito:

Extrae el contenido de un elemento o el valor de un atributo en un documento XML y devuelve datos de tipo REAL.

Sintaxis:

Función escalar:

Función de tabla:

Parámetros:

Tabla 33. Parámetros de lafunción 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 devuelto:
REAL
Códigos de retorno:
returnedReal
Ejemplos:
Ejemplo de función escalar:

En el ejemplo siguiente, el valor de ExtendedPrice se extrae como REAL. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
CREATE TABLE t1(price DECIMAL(9,2));
INSERT INTO t1 values (
    DB2XML.extractReal(DB2XML.xmlfile('instalación_dxx
    /samples/db2xml/xml/getstart.xml'),
    '/Order/Part[@color="black "]/ExtendedPrice'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En el ejemplo siguiente, el valor de ExtendedPrice se extrae como REAL. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
SELECT CAST(RETURNEDREAL AS REAL)
  FROM TABLE(
     DB2XML.extractReals(DB2XML.XMLFile('instalación_dxx
      /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/ExtendedPrice'))   AS X;
```

Conceptos relacionados:

- Apéndice D, "Nombres de UDT y UDF para XML Extender", en la página 325
- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148
- "Códigos de retorno de las UDF de XML Extender" en la página 282

extractChar() y extractChars()

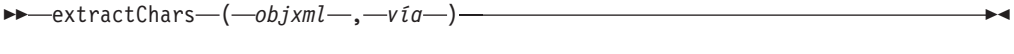
Propósito:

Extrae el contenido del elemento o el valor de atributo en un documento XML y devuelve datos de tipo CHAR.

Sintaxis:

Función escalar:

Función de tabla:



Parámetros:

Tabla 34. Parámetros de la función 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 devuelto:
CHAR
Códigos de retorno:
returnedChar
Ejemplos:
Ejemplo de función escalar:

En el ejemplo siguiente, el valor de Name se extrae como CHAR. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
CREATE TABLE t1(name char(30));
INSERT INTO t1 values (
    DB2XML.extractChar(DB2XML.xmlfile('instalación_dxx
    /samples/db2xml/xml/getstart.xml'),
    '/Order/Customer/Name'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En el ejemplo siguiente, el valor de Color se extrae como CHAR. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
SELECT *
  FROM TABLE(
     DB2XML.extractChars(DB2XML.XMLFile('instalación_dxx
       /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/@color'))   AS X;
```

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148
- "Cómo leer los diagramas de sintaxis" en la página ix

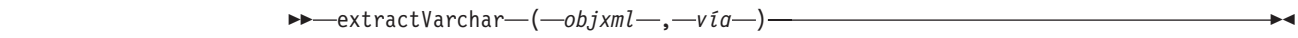
extractVarchar() y extractVarchars()

Propósito:

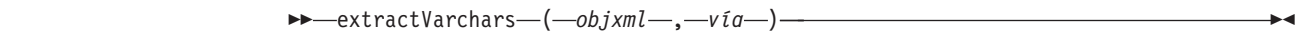
Extrae el contenido de un elemento o el valor de un atributo en un documento XML y devuelve datos de tipo VARCHAR.

Sintaxis:

Función escalar:



Función de tabla:



Parámetros:

Tabla 35. Parámetros de la función extractVarchar y de la función 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 devuelto:
VARCHAR(4 K)
Códigos de retorno:
returnedVarchar
Ejemplos:
Ejemplo de función escalar:

Suponga una base de datos que contiene más de 1000 documentos XML, que 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
```

Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato. 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.

Ejemplo de función de tabla:

Suponga una base de datos que contiene más de 1000 documentos XML, que 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
```

Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato. 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.

Ejemplo de función escalar:

En el ejemplo siguiente, el valor de Name se extrae como VARCHAR. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
CREATE TABLE t1(name varchar(30));
INSERT INTO t1 values (
    DB2XML.extractVarchar(DB2XML.xmlfile('instalación_dxx
     /samples/db2xml/xml/getstart.xml'),
    '/Order/Customer/Name'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En el ejemplo siguiente, el valor de Color se extrae como VARCHAR. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
SELECT*
  FROM TABLE(
     DB2XML.extractVarchars(DB2XML.XMLFile('instalación_dxx
      /samples/xml/getstart.xml'),
      '/Order/Part/@color'))   AS X;
```

Conceptos relacionados:

- Apéndice D, "Nombres de UDT y UDF para XML Extender", en la página 325
- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148
- "Códigos de retorno de las UDF de XML Extender" en la página 282

extractCLOB() y extractCLOBs()

Propósito:

Extrae un fragmento de documentos XML, con markup para el elemento y el atributo y el contenido de los elementos y atributos, incluidos los subelementos. Esta función es diferente de las demás funciones de extracción, que sólo devuelven el contenido de elementos y atributos. Las funciones extractClob(s) se utilizan para extraer fragmentos de documentos, mientras que extractVarchar(s) y extractChar(s) se utilizan para extraer valores simples.

Sintaxis:

Función escalar:

Función de tabla:

Parámetros:

Tabla 36. Parámetros de la función 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 devuelto:
CLOB(10K)
Códigos de retorno:
returnedCLOB
Ejemplos:
Ejemplo de función escalar:

En este ejemplo, el contenido y los identificadores de todos los elementos de nombre se extraen de un pedido de compra. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
CREATE TABLE t1(name DB2XML.xmlclob);
INSERT INTO t1 values (
    DB2XML.extractClob(DB2XML.xmlfile('instalación_dxx
     /samples/db2xml/xml/getstart.xml'),
    '/Order/Customer/Name'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En este ejemplo, todos los atributos de color se extraen de un pedido de compra. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
SELECT *
  FROM TABLE(
     DB2XML.extractCLOBs(DB2XML.XMLFile('instalación_dxx
     /samples/db2xml/xml/getstart.xml'),
      '/Order/Part/@color'))   AS X;
```

Conceptos relacionados:

- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148

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. La fecha debe estar en el siguiente formato: AAAA-MM-DD.

Sintaxis:

Función escalar:

Función de tabla:

Parámetros:

Tabla 37. Parámetros de la función 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 devuelto:
DATE
Códigos de retorno:
returnedDate
Ejemplos:
Ejemplo de función escalar:

En el ejemplo siguiente, el valor de ShipDate se extrae como DATE. Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

```
CREATE TABLE t1(shipdate DATE);
INSERT INTO t1 values (
    DB2XML.extractDate(DB2XML.xmlfile('instalación_dxx
      /samples/db2xml/xml/getstart.xml'),
    '/Order/Part[@color="red   "]/Shipment/ShipDate'));
SELECT * from t1;
```

Ejemplo de función de tabla:

En el ejemplo siguiente, el valor de ShipDate se extrae como DATE.

```
SELECT *
  FROM TABLE(
     DB2XML.extractDates(DB2XML.XMLFile('instalación_dxx
      /samples/db2xml/xml/getstart.xml'),
      '/Order/Part[@color="black "]/Shipment/ShipDate'))   AS X;
```

Conceptos relacionados:

- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148
- "Códigos de retorno de las UDF de XML Extender" en la página 282

extractTime() y extractTimes()

Propósito:

Extrae el contenido de un elemento o el valor de un atributo en un documento XML y devuelve datos de tipo TIME.

Sintaxis:

Función escalar:

Función de tabla:

Parámetros:

Tabla 38. Parámetros de la función 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 devuelto:
TIME
Códigos de retorno:
returnedTime
Ejemplos:
Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

Ejemplo de función escalar:

```
CREATE TABLE t1(testtime  TIME);
INSERT INTO t1 values (
    DB2XML.extractTime(DB2XML.XMLCLOB(
         '<stuff><data>11.12.13</data></stuff>'), '//data'));
SELECT * from t1;
```

Ejemplo de función de tabla:

```
select *
from table(
    DB2XML.extractTimes(DB2XML.XMLCLOB(
        '<stuff><data>01.02.03</data><data>11.12.13</data></stuff>'),
    '//data')) as x;
```

Conceptos relacionados:

- Apéndice D, "Nombres de UDT y UDF para XML Extender", en la página 325
- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148

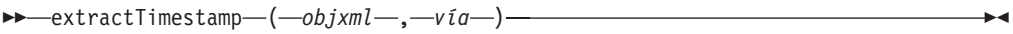
extractTimestamp() y extractTimestamps()

Propósito:

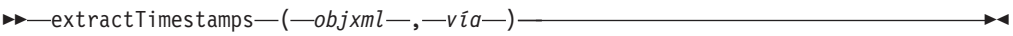
Extrae el contenido de un elemento o el valor de un atributo en un documento XML y devuelve datos de tipo TIMESTAMP.

Sintaxis:

Función escalar:



Función de tabla:



Parámetros:

Tabla 39. Parámetros de la función extractTimestamp y de la función 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 devuelto:
TIMESTAMP
Códigos de retorno:
returnedTimestamp
Ejemplos:
Los ejemplos asumen que está utilizando el shell de mandatos de DB2 UDB, en el que no es necesario escribir "DB2" al principio de cada mandato.

Ejemplo de función escalar:

```
CREATE TABLE t1(testtimestamp  TIMESTAMP);
INSERT INTO t1 values (
    DB2XML.extractTimestamp(DB2XML.XMLCLOB(
        '<stuff><data>2003-11-11-11.12.13.888888</data></stuff>'),
        '//data'));
SELECT * from t1;
```

Ejemplo de función de tabla:

```
select * from
table(DB2XML.extractTimestamps(DB2XML.XMLClob(
    '<stuff><data>2003-11-11-11.12.13.888888
    </data><data>2003-12-22-11.12.13.888888</data></stuff>'),
    '//data')) as x;
```

Si es necesario, XML Extender normalizará automáticamente las indicaciones de la hora extraídas de documentos XML para que cumplan el formato de indicaciones de la hora de DB2. Las indicaciones de la hora se normalizan al formato `aaaa-mm-dd-hh.mm.ss.nnnnnn` o al formato `aaaa-mm-dd-hh mm.ss.nnnnnn`. Por ejemplo:

```
2003-1-11-11.12.13
```

se normalizará a:

```
2003-01-11-11.12.13.000000
```

Conceptos relacionados:

- Apéndice D, "Nombres de UDT y UDF para XML Extender", en la página 325
- "Tipos de funciones definidas por el usuario de XML Extender" en la página 139

Información relacionada:

- "Extracción de funciones en XML Extender" en la página 148
- "Códigos de retorno de las UDF de XML Extender" en la página 282

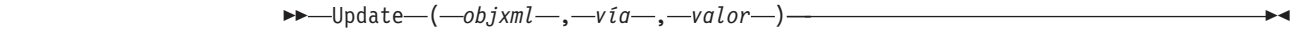
Funciones de actualización en XML Extender

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 de conversión de datos por omisión para convertir un tipo base de SQL al UDT de XML.

Propósito

Utiliza 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 40. 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. **Restricción:** La función de actualización no tiene una opción para inhabilitar el escape de la salida; la salida de un elemento extractClob (que es un fragmento con códigos) no se puede insertar mediante esta función. Utilice únicamente valores de texto.

Restricción: Tenga presente que la UDF de actualización da 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"]'
```

Tipo devuelto

Tipo de datos	Tipo devuelto
XMLVARCHAR	XMLVARCHAR
XMLCLOB como LOCATOR	XMLCLOB

Ejemplo

El ejemplo siguiente actualiza el pedido de compra gestionado por el vendedor Sriram Srinivasan.

```
UPDATE sales_tab
   set order = db2xml.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, la 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. Las secciones siguientes describen cómo se procesa el documento y ofrecen ejemplos de qué aspecto tienen los documentos antes y después de las actualizaciones.

Cómo procesa la función Update() el documento XML

Cuando la función Update() sustituye los documentos XML, debe reconstruir el documento basándose en la salida del analizador XML. La Tabla 41 en la página 163 describe cómo se manejan las partes del documento con ejemplos.

Tabla 41. 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' >	La declaración XML se conserva: • Se conserva la información sobre la versión. • La declaración de codificación se conserva y aparece cuando se especifica 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 definir los valores.
Declaración DOCTYPE		Se conserva la declaración del tipo de documento: • Se da soporte al nombre del elemento raíz. • Los ExternalID públicos y del sistema se preservan y aparecen cuando se especifican en el documento original. • El subconjunto de DTD interna *no* se preserva. 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 definir los valores URI públicos y privadas. • El analizador XML4c actual no informa de una declaración XML que no contenga un ExternalID y de subconjunto de DTD interno. Después de la actualización, faltaría en este caso la 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] >	
Instrucciones de proceso	<?xml-stylesheet title="compact" href="datatypes1.xsl" type="text/xsl"?>	Las instrucciones de proceso se conservan.

Tabla 41. Normas de la función Update (continuación)

Tipo de elemento o nodo	Ejemplo de código de documento XML	Estado después de la actualización
Comentarios	`<!-- comment -->`	Los comentarios dentro del elemento raíz se preservan. Se eliminan los comentarios fuera del elemento raíz.

Elementos	`<books>`
	content
	`</books>`
		Se conservan los elementos.

Atributos	`id='1' date="01/02/2003"`	Se conservan los atributos de los elementos. • Después de la actualización, se utilizan comillas dobles para definir los valores. • Se pierden los datos dentro de los atributos. • Se sustituyen las entidades.
Nodos de texto	`Esta sección trata acerca de mi perro &mydog;.`	Se conservan los nodos de texto (contenido del elemento). • Se pierden los 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 con el valor proporcionado en el parámetro *value*.

Puede especificar un predicado en el parámetro *vía* para proporcionar vías de ubicación diferentes con el fin de evitar actualizaciones involuntarias. La UDF Update() da soporte a vías de ubicación que tengan predicados con atributos, pero no elementos.

Ejemplos

En los siguientes ejemplos se muestran casos de un documento XML antes y después de una actualización.

Tabla 42. Documentos XML antes y después de una actualización

Ejemplo 1:

Antes:

Tabla 42. Documentos XML antes y después de una actualización (continuación)

<?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 comentario 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="07/01/2003"> <!-- first section --> <section>This is a section in Chapter One.</section> </chapter> <chapter id="2" date="01/02/2003"> <section>This is a section in Chapter Two.</section> <footnote>A footnote in Chapter Two is here.</footnote> </chapter> <price date="12/22/2003" time="11.12.13" timestamp="2003-12-22-11.12.13.888888"> 60.02</price> </book>	• 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 un subconjunto de DTD interna. No está soportado.

Tabla 42. Documentos XML antes y después de una actualización (continuación)

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 de DTD interna
	• 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 subconjunto de DTD interna
	• Se resuelve y se sustituye la entidad en el nodo de texto

Funciones de validación

DB2 XML Extender ofrece dos funciones definidas por el usuario (UDF) que validan documentos XML contra un esquema XML o una DTD.

Un elemento de un documento XML es válido de acuerdo con un esquema determinado si se cumplen las normas de tipo de elemento asociadas. Si todos los elementos son válidos, todo el documento será válido. Con una DTD, sin embargo, no existe ningún modo de requerir un elemento raíz específico. Las funciones de validación devuelven 1 si el documento es válido o devuelven 0 y graban un mensaje de error en el archivo de rastreo si el documento no es válido. Las funciones son las siguientes:

db2xml.svalidate:
Valida una instancia de documento XML contra es esquema especificado.

db2xml.dvalidate:
Valida una instancia de documento XML contra la DTD especificada.

Función SVALIDATE()

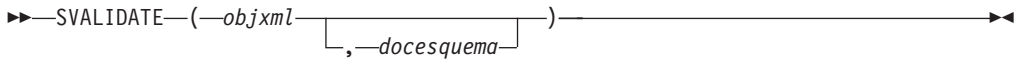
Esta función valida un documento XML contra un esquema especificado (o aquel mencionado en el documento XML) y devuelve 1 si el documento es válido o 0 si no lo es. Esta función asume que el documento XML y un esquema existen en el sistema de archivos o como un CLOB en DB2.

Antes de ejecutar la función SVALIDATE, asegúrese de que XML Extender está habilitado en la base de datos ejecutando el siguiente mandato:

```
dxxadm enable_db nombredb
```

Si el documento XML no se puede validar, se grabará un mensaje de error en el archivo de rastreo de XML Extender. Habilite el rastreo antes de ejecutar el mandato SVALIDATE.

Sintaxis



Parámetros

Tabla 43. Parámetros de SVALIDATE

Parámetro	Tipo de datos	Descripción
objxml	VARCHAR(256)	Es la vía de archivo del documento XML que se debe verificar.
	CLOB(2G)	Columna XML que contiene el documento a verificar.
docesquema	VARCHAR(256)	Es la vía de archivo del documento del esquema.
	CLOB(2G)	Columna XML que contiene el esquema.

Ejemplos

Ejemplo 1: Este ejemplo valida `equiplog2001.xml` contra el esquema especificado en el documento.

```
db2 values db2xml.svalidate("/home/jean/xml/equiplog2001.xml")
```

Ejemplo 2: Este ejemplo valida un documento XML utilizando el esquema especificado y tanto el documento como el esquema se almacenan en tablas de DB2 UDB.

```
db2 select db2xml.svalidate(doc,schema) from equiplogs where id=1
```

Función DVALIDATE()

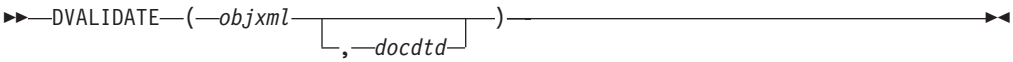
Esta función valida un documento XML contra una DTD especificada (o una mencionada en el documento XML) y devuelve 1 si el documento es válido o 0 si no lo es. Esta función asume que existe un documento XML y una DTD en el sistema de archivos o como un CLOB en DB2.

Antes de ejecutar la función DVALIDATE, asegúrese de que XML Extender se encuentra habilitado con la base de datos ejecutando el siguiente mandato:

```
dxxadm
enable_db minombrebd
```

Si el documento XML no se puede validar, se grabará un mensaje de error en el archivo de rastreo de XML Extender. Habilite el rastreo antes de ejecutar el mandato DVALIDATE.

Sintaxis



Parámetros

Tabla 44. Parámetros de DVALIDATE

Parámetro	Tipo de datos	Descripción
objxml	VARCHAR(256)	Es la vía de archivo del documento XML que se debe verificar.
	CLOB(2G)	Columna XML que contiene el documento a verificar.
docdtd	VARCHAR(256)	Es la vía de archivo del documento de la DTD.
	CLOB(2G)	Columna XML que contiene la DTD, que procede de la tabla DTD_REF o de una tabla normal.

Ejemplos

Ejemplo 1: Este ejemplo valida `equiplog2001.xml` contra la DTD especificada en el documento.

```
db2 values db2xml.dvalidate(/home/jean/xml/equiplog2001.xml)
```

Ejemplo 2: Este ejemplo valida un documento XML utilizando la DTD especificada, y tanto el documento como la DTD se encuentran en el sistema de archivos.

```
db2 values db2xml.dvalidate (c:/xml/equiplog.xml,c:/xml/dtds/equip.dtd)
```

Ejemplo 3: Este ejemplo valida un documento XML utilizando la DTD especificada, y tanto el documento como la DTD se almacenan en tablas de DB2 UDB.

```
db2 values db2xml.dvalidate (doc,dtdid) from equiplogs, db2xml.dtd_ref \
    where dtdid="equip.dtd"
```

Información relacionada:

- "Inicio del rastreo de XML Extender" en la página 281

Capítulo 9. Archivos de definición de acceso a documento (DAD)

Creación de un archivo DAD para columnas XML

Esta tarea forma parte de la tarea más amplia de definición y habilitación de una columna XML.

Vea el sitio Web de XML Extender en la dirección www.ibm.com/software/data/db2/extenders/xmlext/downloads.html para obtener la información más reciente sobre los archivos DAD.

Para acceder a los datos XML y habilitar columnas para datos XML en una tabla XML, es necesario definir un archivo DAD (definición de acceso a documento). Este archivo define los atributos y elementos clave de los datos que deben buscarse dentro de la columna. Para las columnas XML, el archivo DAD especifica principalmente cómo se deben indexar los documentos guardados dentro de dicho archivo. El archivo DAD también especifica una DTD o esquema que se utiliza para validar documentos insertados en la columna XML. Los archivos DAD se almacenan como un tipo de datos CLOB y su límite de tamaño es de 100 KB.

Requisitos previos:

Antes de crear el archivo DAD, es necesario:

- Decidir los elementos o atributos que el usuario espera utilizar a menudo en la búsqueda. Los elementos o atributos que especifique se extraerán en las tablas auxiliares para las búsquedas rápidas por parte de XML Extender.
- Definir la vía de ubicación para representar cada elemento o atributo indexado en una tabla auxiliar. También debe especificar el tipo de datos al que desea que se convierta el elemento o el atributo.

Procedimiento:

Para crear un archivo DAD:

1. Cree un documento nuevo en un editor de texto y escriba la siguiente sintaxis:

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

   "vía/dtd/dad.dtd " es el nombre de vía y archivo de la DTD del archivo DAD. Se proporciona una DTD en `instalación_dxx\samples\db2xml\dtd`.
2. Inserte los códigos de DAD después de las líneas a partir del paso 1.

   ```
   <DAD>
   </DAD>
   ```

   Este elemento contendrá los demás elementos.
3. Especifique la validación para el documento y la columna:
   - Si desea validar todo el documento XML con una DTD o esquema antes de insertarlo en la base de datos:
     - Inserte el código correspondiente para especificar cómo desea validar el documento:

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

– Inserte el siguiente código para validar el documento utilizando un esquema:

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

– Valide la columna insertando los siguientes códigos:

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

- Si no desea validar el documento, utilice el código siguiente:

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

4. Inserte los códigos <Xcolumn> </Xcolumn> para especificar que está utilizando columnas XML como método de acceso y almacenamiento de los datos XML.
5. Especifique las tablas auxiliares. Para cada tabla auxiliar que desee crear:
   a. Especifique un código <table></table>. Por ejemplo:

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

   b. Dentro de los códigos de la tabla, inserte un código <column> para cada columna que desee que la tabla auxiliar contenga. Cada columna tiene cuatro atributos: name, type, path y multi_occurrence.

      Ejemplo:

```
<table name="person_names">>
<column name ="fname"
          type="varchar(50)"
     path="/person/firstName"
            multi_occurrence="NO"/>
<column name ="lname"
          type="varchar(50)"
     path="/person/lastName"
            multi_occurrence="NO"/>
</table>
```

      Donde:

      name Especifica el nombre de la columna que se crea en la tabla auxiliar.

      type Indica el tipo de datos de SQL en la tabla auxiliar para cada atributo o elemento indexado.

      vía Especifica la vía de ubicación del documento XML para cada elemento o atributo que debe indexarse

      multi_occurrence
      Indica si el elemento o el atributo al que se hace referencia mediante el atributo de vía puede aparecer más de una vez en el documento XML. Los posibles valores de **multi_occurrence** son **YES** o **NO**. Si el valor es **NO**, podrá especificar varias columnas por tabla. Si el valor es **YES**, sólo se podrá especificar una columna en la tabla auxiliar.
6. Guarde el archivo con una extensión DAD.

El siguiente ejemplo muestra un archivo DAD completo:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM
"c:\instalación_dxx\samples\db2xml\dtd\dad.dtd">
<DAD>
<dtid>C:\SG246130\code\person.dtd</dtdid>
<validation>YES</validation>
<Xcolumn>
  <table name="person_names">
```

```
   <column name="fname"
          type="varchar(50)"
     path="/person/firstName"
            multi_occurrence="NO"/>
   <column name="lname"
          type="varchar(50)"
     path="/person/lastName"
            multi_occurrence="NO"/>
</table>
<table name="person_phone_number">
   <column name="pnumber"
     type="varchar(20)"
     path="/person/phone/number"
            multi_occurrence="YES"/>
</table>
<table name="person_phone_number">
   <column name="pnumber"
     type="varchar(20)"
     path="/person/phone/number"
            multi_occurrence="YES"/>
</table>
<table name="person_phone_type">
   <column name="ptype"
     type="varchar(20)"
     path="/person/phone/type"
            multi_occurrence="YES"/>
</table>
<Xcolumn>
</DAD>
```

Ahora que ha creado un archivo DAD, el siguiente después de definir y habilitar una columna XML es crear la tabla en la que se almacenarán los documentos XML.

Conceptos relacionados:

- "Colecciones XML como método de almacenamiento y acceso" en la página 93
- "Archivos DAD para colecciones XML" en la página 171
- "Comprobador del archivo DAD" en la página 186

Tareas relacionadas:

- "Utilización del comprobador de DAD" en la página 187

Archivos DAD para colecciones XML

Para las colecciones XML, el archivo DAD correlaciona la estructura del documento XML con las tablas de DB2 ® desde las cuales compone el documento. También puede descomponer documentos en las tablas de DB2 utilizando el archivo DAD.

Por ejemplo, si tiene un elemento llamado <Tax> en el documento XML, deberá correlacionar <Tax> con una columna llamada TAX. El archivo DAD se utiliza para definir la relación entre los datos XML y los datos relacionales.

Debe especificar el archivo DAD al habilitar una colección o al utilizar el archivo DAD en procedimientos almacenados para colecciones XML. La DAD es un documento con formato XML y reside en el sistema cliente. Si decide validar los documentos XML utilizando una DTD, el archivo DAD se puede asociar a esa DTD. Cuando se utiliza como el parámetro de entrada de los procedimientos almacenados de XML Extender, el archivo DAD tiene un tipo de datos de CLOB. Este archivo puede tener hasta 100 KB.

Para especificar el método de acceso y almacenamiento de la colección XML, utilice el código <Xcollection> 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 conjunto de tablas relacionales que contiene datos XML. Los programas de aplicación pueden habilitar una colección XML de tablas de usuario cualesquiera. Estas tablas de usuario pueden ser tablas de datos comerciales existentes o tablas que XML Extender haya creado recientemente.

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

root_node (nodo raíz)
Especifica el elemento raíz del documento.

element_node (nodo de elemento)
Identifica un elemento, que puede ser el elemento raíz o un elemento hijo.

text_node (nodo de texto)
Representa el texto CDATA de un elemento.

attribute_node (nodo de atributo)
Representa un atributo de un elemento.

La Figura 14 en la página 173 muestra un fragmento de la correlación que se utiliza 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:\dxx\samples\db2xml\dtd\dad.dtd">
<DAD>
  ...
<Xcollection>
<SQL_stmt>
       ...
</SQL_stmt> <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM
                  ""c:\dxx\samples\db2xml\dtd\getstart.dtd""</doctype>
<root_node>
   <element_node name="Order">         --> 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 el
                                      elemento y el atributo
                                      se correlacionan
             </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 14. Definiciones de nodo para el documento XML que está correlacionado con la tabla de la colección XML

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

XML Extender también da soporte a las instrucciones de proceso para hojas de estilo, utilizando 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>
```

Utilice cualquier editor de texto para crear y actualizar un archivo DAD.

Conceptos relacionados:

- "Esquemas de correlación para colecciones XML" en la página 105

Composición SQL

Puede componer documentos XML utilizando columnas con el mismo nombre. Las columnas seleccionadas con el mismo nombre, aunque sean de tablas diferentes, deben estar identificadas mediante un alias exclusivo de forma que cada variable en la cláusula Select de la sentencia de SQL sea diferente. El siguiente ejemplo muestra cómo dar a columnas con el mismo nombre aliases exclusivos.

```
<SQL_stmt>select o.order_key as oorder_key,      key customer_name, customer_email,
                 p.part_key p.order_key as porder_key,
                 color, qty, price, tax, ship_id, date, mode
            from order_tab o.part_tab p
            order by order_key, part_key</SQL_stmt>
```

También puede componer documentos XML utilizando columnas con valores aleatorios generados. Si una sentencia de SQL en un archivo DAD tiene un valor aleatorio, deberá a la función de valor aleatorio un alias para que lo utilice en la cláusula ORDER BY. Este requisito es necesario ya que el valor no está asociado con ninguna columna en una tabla determinada. Consulte el alias de generate_unique al final de la cláusula ORDER BY en el siguiente ejemplo.

```
<SQL_stmt>select o.order_key, customer_name,customer_email,
                 p.part_key,color,qty,price,tax,ship_id,
                 date, mode
          from order_tab o,part_tab p,
               table(select substr(char(timestamp(generate_unique())),16)
as ship_id, date, mode,
                           part_key
                      from ship_tab) s
          where o.order_key=1 and p.price>2000 and
                o.order_key=o.order_key and s.part_key
          order by order_key, part_key,ship_id</SQL_stmt>
```

Composición de nodo RDB

Las restricciones siguientes se aplican a la composición de nodo RDB:

- La condición asociada con cualquier archivo DAD de nodo RBD no root_node debe compararse contra un literal.
- Cada igualdad en la condición asociada con un RDB_node superior especifica la relación de unión entre las columnas de dos tablas y se aplica independientemente de las otras igualdades. Es decir, todos los predicados conectados mediante AND no se aplican simultáneamente para una sola condición de unión; simulan de este modo una unión exterior cuando se compone el documento. La relación padre-hijo entre cada par de tablas está determinada por su anidamiento relativo en el archivo DAD. Por ejemplo:

```
<condition>order_tab.order_key=part_tab.order_key AND
part_tab.part_key=ship_tab.part_key</condition>
```

Composición partiendo de filas con valores nulos

Puede utilizar columnas que tengan valores nulos para componer documentos XML.

El siguiente ejemplo ilustra cómo generar un documento XML partiendo de una tabla *MyTable* que contiene una fila con un valor nulo en la columna *Col 1*. LA DAD utilizada en el ejemplo es *nullcol.dad*.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO validation>NO>
<Xcollection>
```

```
<SQL_stmt>SELECT 1 as X, Col1 FROM MyTable order by X, Col1<\SQL_stmt>
<prolog>?xml version="1.0"?prolog>?xml version="1.0"?>
<doctype>!DOCTYPE Order SYSTEM "e:\t3xml\x.dtd">
<root_node>
<element_node name="MyColumn">
<element_node name="Column1" multi_occurrence="YES">
  <text_node>
  <column name="Col1"/>
  </text_node>
 </element_node>
      </element_node>
</roott_node>
</Xcollection>
</DAD>

MyTable
__________
| Col 1 |
|_______|
| 1 |
|_______|
| 3 |
|_______|
| - |
|_______|
```

Ejecute tests2x mydb nullcol.dad result_tab o utiliza dxxGenXML para producir el siguiente documento: tenga en cuenta que el tercer elemento Column1 representa un valor nulo.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "e:\t3xml\x.dtd">
<MyColumn>
 <Column1>1</Column1>
 <Column1>3</Column1>
 <Column1></Column1>
</MyColumn>
```

- La condición asociada con cualquier archivo DAD de nodo RBD no root_node debe compararse contra un literal.
- La condición asociada con cualquier nodo RDB de nivel bajo en la DAD debe compararse con un literal.
- La condición asociada con un root_node describe la relación entre las tablas que toman parte en la composición del nodo RDB. Por ejemplo, una relación de clave foránea primaria.
- Cada igualdad en la condición asociada con un RDB_node superior especifica la relación de unión entre las columnas de dos tablas y se aplica independientemente de las otras igualdades. Es decir, todos los predicados conectados mediante AND no se aplican simultáneamente para una única condición de unión, simulando de este modo una unión exterior cuando se compone el documento. La relación padre-hijo entre cada par de tablas está determinada por su anidamiento relativo en el archivo DAD. Por ejemplo:

```
<condition>order_tab.order_key=part_tab.order_key AND
part_tab.part_key=ship_tab.part_key</condition>
```

DTD para el archivo DAD

Esta sección describe las DTD (declaraciones de tipo de documento) del archivo DAD (definición de acceso a documento). El propio archivo DAD es un documento XML con estructura de árbol y necesita una DTD. El nombre del archivo DTD es `dad.dtd`. El ejemplo siguiente muestra la DTD para el archivo DAD.

```
<?xml encoding="US-ASCII"?>

  <!ELEMENT DAD ((schemabindings | dtdid)?, validation,
  (Xcolumn | Xcollection))>
  <!ELEMENT dtdid (#PCDATA)>
  <!ELEMENT schemabindings (nonamespacelocation)>
  <!ELEMENT nonamespacelocation (empty)>
  <!ATTLIST nonamespacelocation location CDATA #REQUIRED>
  <!ELEMENT validation (#PCDATA)>
  <!ELEMENT Xcolumn (table+)>
<!ELEMENT table (column+)>
 <!ATTLIST table name CDATA #REQUIRED
                              key CDATA #IMPLIED
                              orderBy CDATA #IMPLIED>
  <!ELEMENT column EMPTY>
<!ATTLIST column
                     name CDATA #REQUIRED
                     type  CDATA #IMPLIED
                     path  CDATA #IMPLIED
                       multi_occurrence CDATA #IMPLIED>
  <!ELEMENT Xcollection (SQL_stmt?, prolog, doctype, root_node)>
  <!ELEMENT SQL_stmt (#PCDATA)>
  <!ELEMENT prolog (#PCDATA)>
  <!ELEMENT doctype (#PCDATA | RDB_node)*>
  <!ELEMENT root_node (element_node)>
  <!ELEMENT element_node (RDB_node*,
                                       attribute_node*,
                                       text_node?,
                                       element_node*,
                                       namespace_node*,
                                       process_instruction_node*,
                                       comment_node*)>
  <!ATTLIST element_node
                     name CDATA #REQUIRED
                     ID CDATA #IMPLIED
                     multi_occurrence CDATA "NO"
                     BASE_URI CDATA #IMPLIED>
  <!ELEMENT attribute_node (column | RDB_node)>
  <!ATTLIST attribute_node
                     name CDATA #REQUIRED>
  <!ELEMENT text_node (column | RDB_node)>
  <!ELEMENT RDB_node (table+, column?, condition?)>
  <!ELEMENT condition (#PCDATA)>
  <!ELEMENT comment_node (#PCDATA)>
  <!ELEMENT process_instruction_node (#PCDATA)>
```

El archivo DAD tiene cuatro elementos principales:

- DTDID
- validation
- Xcolumn
- Xcollection

Xcolumn y Xcollection tienen elementos y atributos hijos que ayudan a correlacionar los datos XML con tablas relacionales de DB2. La lista siguiente describe los elementos principales y los elementos y atributos hijos. Los ejemplos de sintaxis proceden del ejemplo anterior.

Elemento DTDID

Las DTD que se proporcionan XML Extender se almacenan en la tabla DTD_REF. Cada DTD está identificada mediante un ID exclusivo que se facilita en el código `DTDID` del archivo DAD. El código `DTDID` apunta a la DTD que valida los documentos XML o dirige la correlación entre las tablas de la colección XML y los documentos XML. Para las colecciones

XML, este elemento sólo se requiere en la validación de documentos XML de entrada y salida. Para las columnas XML, este elemento sólo es necesario en la validación de documentos XML de entrada. El elemento `DTDID` debe ser igual que el ID del sistema (SYSTEM ID) especificado en el elemento `doctype` de los documentos XML.

Sintaxis: `<!ELEMENT dtdid (#PCDATA)>`

Elemento validation
Indica si el documento XML se valida con 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 atributo obligatorio y dos atributos implicados:

Atributo name
Especifica el nombre de la tabla auxiliar.

Atributo key
La clave única primaria de la tabla.

Atributo orderBy
Son los nombres de las columnas que determinan el orden secuencial de valores de atributo o texto de elementos que aparecen varias veces al generar documentos XML.

El elemento table tiene un elemento hijo:

Elemento column
Correlaciona un atributo de un nodo CDATA del documento XML de entrada con una columna de la tabla.

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 alias de la vía de ubicación que identifica un elemento o atributo.

Atributo type
Define el tipo de datos de la columna. Puede ser cualquier tipo de datos SQL.

Atributo path
Muestra la vía de ubicación de un elemento o atributo XML y debe ser la vía de ubicación simple como se especifica en la Tabla 3.1.a.

Atributo multi_occurrence
Indica si el elemento o atributo puede aparecer más de una vez en un documento XML. 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 siguientes elementos hijo:

SQL_stmt
Especifica la sentencia de SQL que XML Extender utiliza para definir la colección. En concreto, la sentencia selecciona datos XML de las tablas de la 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 para componer documentos, y sólo está permitido un único valor de `SQL_stmt`.

Sintaxis: `<!ELEMENT SQL_stmt  #PCDATA >`

prolog (prólogo)
Es el texto del prólogo XML. Se proporciona el mismo prólogo para 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 utiliza para especificar el tipo de documento (DOCTYPE) del documento resultante. Para definir un valor explícito. Este valor se proporciona para todos los documentos de la colección completa.

`doctype` tiene un elemento hijo:

root_node (nodo raíz)
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 (nodo raíz)` del documento XML.

Sintaxis: `<!ELEMENT root_node(element_node)>`

RDB_node
Define la tabla de DB2 UDB donde almacenar el contenido del elemento XML o el valor de un atributo XML o bien desde el cual se va a recuperar. rdb_node es un elemento hijo de element_node, text_node y attribute_node, y tiene los siguientes elementos hijo:

table Especifica la tabla en la que se almacena el contenido del elemento o atributo.

column
Especifica la columna en la que se almacena el contenido del elemento o el atributo.

condition
Especifica una condición para la columna. Es opcional.

element_node (nodo de elemento)
Representa un elemento XML. Debe definirse en la DAD especificada para la colección. Para la correlación de `RDB_node`, el `element_node` raíz debe tener un `RDB_node` a fin de especificar todas las tablas que contienen datos XML para él mismo y para todos sus nodos hijos. Puede haber 0, 1 o más `nodos de atributo` y `nodos de elemento` hijos, así como 0 ó 1 `nodos de texto`. Para aquellos elementos que no sean el elemento raíz no es necesario ningún nodo RDB.

Sintaxis:

Un `element_node (nodo de elemento)` se define mediante los siguientes elementos hijo:

RDB_node
(Opcional) Especifica tablas, columnas y condiciones para datos XML. El RDB_node de un elemento sólo necesita definirse para la correlación de RDB_node. En este caso, se deben especificar una o más tablas. La columna no es necesaria ya que el contenido del elemento está especificado por su text_node (nodo de texto). La condición es opcional, dependiendo de la DTD y de la condición de consulta.

nodos hijos
Opcional: Un `element_node` puede también tener los siguientes nodos hijo:

element_node (nodo de elemento)
Representa los elementos hijo del elemento XML actual.

attribute_node (nodo de atributo)
Representa los atributos del elemento XML actual.

text_node (nodo de texto)
Representa el texto CDATA del elemento XML actual.

attribute_node (nodo de atributo)
Representa un atributo XML. Es el nodo que define la correlación entre un atributo XML y los datos de columna en una tabla relacional.

Sintaxis:

El attribute_node (nodo de atributo) 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 siguientes elementos hijo:

column
Se utiliza para la correlación de SQL. La columna se debe especificar en la cláusula SELECT de SQL_stmt.

RDB_node
Se utiliza para la correlación de RDB_node. 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 (nodo de texto)
Representa el contenido de texto de un elemento XML. Es el nodo que define la correlación entre el contenido del elemento XML y los datos de columna en una tabla relacional.

Sintaxis: Debe estar definido por un elemento hijo column o bien RDB_node:

column
Es necesario para la correlación de SQL. En este caso, la columna se debe especificar en la cláusula SELECT de SQL_stmt.

RDB_node
Es necesario para la correlación de RDB_node. El nodo define la correlación entre el contenido de texto y los datos de columna de la tabla relacional. Se deben especificar table (tabla) y column (columna). La condición es opcional.

Conceptos relacionados:

• "Archivos DAD para colecciones XML" en la página 171

Tareas relacionadas:

• "Alteración temporal dinámica de valores en el archivo DAD" en la página 180

Alteración temporal dinámica de valores en el archivo DAD

Procedimiento:

En las consultas dinámicas, puede utilizar dos parámetros opcionales para alterar temporalmente condiciones definidas en el archivo DAD: *override* y *overrideType*. Basándose en el valor de *overrideType*, la aplicación puede alterar temporalmente los valores de código <SQL_stmt> para la correlación de SQL o las condiciones de RDB_nodes para la correlación de RDB_node de la DAD.

Estos parámetros tienen los valores y reglas siguientes:

overrideType
Es un parámetro de entrada obligatorio que identifica el tipo de parámetro *override*. El parámetro *overrideType* tiene los valores siguientes:

NO_OVERRIDE
Especifica que no se altere temporalmente una condición definida en el archivo DAD.

SQL_OVERRIDE
Especifica que se altere temporalmente una condición en el archivo DAD con una sentencia de SQL.

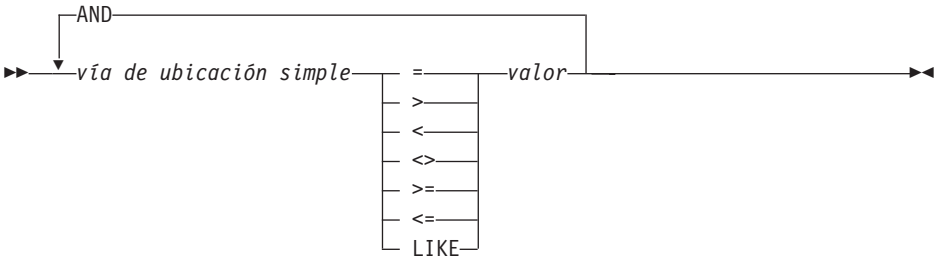
XML_OVERRIDE
Especifica que se altere temporalmente, mediante una condición basada en XPath, una condición definida en el archivo DAD.

override
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 el parámetro *overrideType*:

- Si especifica NO_OVERRIDE, el valor de entrada es una serie NULL.
- Si especifica SQL_OVERRIDE, el valor de entrada es una sentencia de SQL válida.

  Si utiliza SQL_OVERRIDE como sentencia de SQL, debe utilizar el esquema de correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de SQL especificada mediante el elemento <SQL_stmt> en el archivo DAD.
- Si especifica XML_OVERRIDE, el valor de entrada es una serie que contiene una o más expresiones.

  Si utiliza XML_OVERRIDE y una expresión, deberá utilizar el esquema de correlación de RDB_node en el archivo DAD. La expresión XML de entrada altera temporalmente la condición RDB_node especificada en el archivo DAD. La expresión utiliza la sintaxis siguiente:



Esta sintaxis contiene los componentes siguientes:

vía de ubicación simple
Especifica una vía de ubicación simple, utilizando la sintaxis definida por XPath.

operadores
Los operadores SQL que figuran en el diagrama de sintaxis pueden tener un espacio para separar el operador de las demás partes de la expresión.

Los espacios antes y después de los operadores son opcionales. Los espacios son obligatorios antes y después del operador LIKE.

valor
Valor numérico o serie entre comillas simples.

AND
And funciona como un operador lógico en la misma vía de ubicación. Si

se especifica una vía de ubicación simple más de una vez en la serie de alteración, todos los predicados de dicha vía de ubicación simple se aplicarán simultáneamente.

Si se especifica XML_OVERRIDE, la condición de RDB_node en text_node o attribute_node que coincida con la vía de ubicación simple queda alterada temporalmente por la expresión especificada.

XML_OVERRIDE no es del todo compatible con XPath. La vía de ubicación simple sólo se utiliza para identificar el elemento o el atributo que está correlacionado con una columna.

Los ejemplos siguientes utilizan SQL_OVERRIDE y XML_OVERRIDE para mostrar la alteración temporal dinámica.

Ejemplo 1: Procedimiento almacenado que hace uso de SQL_OVERRIDE. En este ejemplo, el elemento <xcollection> del archivo DAD debe tener un elemento <SQL_stmt>. El parámetro *override* prevalece sobre el valor de <SQL_stmt>, cambiando el precio para que sea mayor que 50,00 y la fecha para que sea mayor que 1998-12-01.

```
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[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   collection_ind;

         short   ovtype_ind;
         short   ov_ind;
         short   maxrow_ind;
         short   numrow_ind;
         short   returnCode_ind;
         short   returnMsg_ind;

         EXEC SQL END DECLARE SECTION;

     /* crear tabla */
         EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

              /* inicializar variable del 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)  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 db2xml.dxxRetrieve(:collection:collection_ind;
          :result_tab:rtab_ind,
               :overrideType:ovtype_ind,:override:ov_ind,
          :max_row:maxrow_ind,:num_row:numrow_ind,
          :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Ejemplo 2: Procedimiento almacenado que hace uso de XML_OVERRIDE. En este ejemplo, el elemento <xcollection> del archivo DAD tiene un RDB_node para el element_node (nodo de elemento) raíz. El valor de *override* se basa en el contenido de XML. XML Extender convierte la vía de ubicación simple en la columna de DB2 UDB correlacionada.

```
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[256];   /* alteración temporal; condición XPATH */
         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   collection_ind;
         short   ovtype_ind;
short   ov_ind;
short   maxrow_ind;
short   numrow_ind;
short   returnCode_ind;
short   returnMsg_ind;

EXEC SQL END DECLARE SECTION;

/* crear tabla */
EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

             /* inicializar variable del 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);
```

Varias alteraciones temporales

XML Extender da soporte a varias alteraciones temporales en la misma vía. Se aceptarán todas las alteraciones temporales especificadas para el nodo RDB.

Es posible especificar varias alteraciones temporales de XML en la misma vía de ubicación para ajustar las condiciones establecidas de la búsqueda. En el siguiente ejemplo, se compone un documento XML a partir de las dos tablas utilizando el archivo test.dad.

Tabla 45. Tabla de departamento

Número de departamento	Nombre del departamento
10	Diseño técnico
20	Operaciones
30	Marketing

Tabla 46. Tabla de empleados

Número de empleado	Número de departamento	Salario
123	10	98.000,00 dólares
456	10	$87.000,00 dólares
111	20	65.000,00 dólares
222	20	71.000,00 dólares
333	20	66.000,00 dólares
500	30	55.000,00 dólares

El archivo DAD `test.dad` descrito a continuación contiene una condición que compara la variable *deptno* con el valor 10. Para ampliar la búsqueda a valores mayores de 10 y menores 30, es necesario alterar temporalmente la condición. Es necesario establecer el parámetro de alteración temporal al llamar a dXXGenXML como se describe a continuación:

```
/ABC.com/Department>10 AND /ABC.com/Department<30

<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "C:\dxx_xml\test\dtd\dad.dtd">
<DAD>
 <dtdid>E:\dtd\lineItem.dtd</dtdid>
<validation>NO</validation>    <Xcollection>
<porlog>?xmol version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd"</doctype>
   <root_node>
<element_node name="ABC.com">
<TDB_node>
<table name="dept" key="deptno"/>
<table name="empl" key="emplno"/>
<condition>dept deptno=empl.deptno</condition>
```

```
          </RDB_node>

<element_node name="Department" multi_occurrence="YES">
     <text_node>
         <RDB_node>
<table name="dept"/>
<column name="deptno">
<condition>deptno=10</condition><RDB_node></RDB_node><text_node></text_node>
<element_node name="Employees" multi_occurrence="YES">

     <text_node>

         <RDB_node>

<table name="dept"><column name="deptnot"><condition>deptno=10</condition>
</table></RDB_node></text_node>
<element_node name="Employees" multi_occurence="YES">

<element_node name="EmployeeNo">

     <text_node>

         <RDB_node>

<table name="empl"><column name="emplno"><condition>emplno<500</condition>
</table></RDB_node></text_node></element_node>
<element_node name="Salary">

     <text_node>

         <RDB_node>

<table name="empl"><column name="salary"><condition>salary>5000.00</condition>
</table></RDB_node></text_node></element_node></element_node></element_node>
```

Para componer un documento XML sin realizar una alteración temporal, entre tests2x mydb test.dad result_tab o invoque dxxGenXML sin establecer una alteración temporal. De esta forma se generará un documento similar al siguiente:

```
<?xml version="1.0">
<!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd">
<ABC.com>
<Department>10
<Employees>
<EmployeeNo>123</EmployeeNO>
<Salary>98,000.00</Salary>
</Employees>
<Employees>
<EmployeeNo>456</EmployeeNo>
<Salary>87,000.00</Salary>
</Employees>
</Department>
</ABC.COM>
```

Para alterar temporalmente el archivo DAD puede invocar dxxGenXML como se menciona anteriormente o bien ejecute el programa test2x con las condiciones especificadas:

tests2x mydb test.dad result_tab -o 2 ″/ABC.com/Department>10 AND /ABC.com/Department<30″

```
<?xml version="1.0">
<!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd">
<ABC.com>
<Department>20
```

```
<Employees>
<EmployeeNo>111</EmployeeNo>
<Salary>65,000.00</Salary>
</Employees>
<EmployeeNo>222</EmployeeNo>
<Salary>71,000.00</Salary>
</Employees>
<Employees>
<EmployeeNo>333</EmployeeNo>
<Salary>66,000.00</Salary>
</Employees>
</Department>
</ABC.com>
```

Conceptos relacionados:

- "Archivos DAD para colecciones XML" en la página 171
- "Comprobador del archivo DAD" en la página 186

Tareas relacionadas:

- "Creación de un archivo DAD para columnas XML" en la página 169
- "Utilización del comprobador de DAD" en la página 187

Información relacionada:

- "DTD para el archivo DAD" en la página 175

Comprobador del archivo DAD

El comprobador de DAD puede utilizarse para verificar la validez de los archivos DAD que utilizan el método de almacenamiento de la colección XML. En cada archivo DAD se especifica un esquema de correlación que define la relación entre las tablas y la estructura del documento XML.

Al igual que se utilizan las DTD (descripciones de tipo de documento) para validar la sintaxis de los documentos XML, el comprobador DAD se utiliza para asegurarse de que un archivo DAD es semánticamente correcto. Esta validación puede tener lugar sin conectarse a una base de datos. El uso del comprobador de DAD puede ayudar a minimizar el número de errores que se producen cuando se somete el archivo al XML Extender para procesarlo. El comprobador de DAD es una aplicación de Java™ que se invoca desde la línea de mandatos. Cuando se invoca, produce un conjunto de dos archivos de salida que contienen errores, avisos e indicadores de procesos satisfactorios. Los dos archivos son equivalentes; uno es un archivo de texto plano que se utiliza para comprobar errores o avisos; el otro es un archivo XML, `errorsOutput.xml`, que comunica los resultados del comprobador DAD a otras aplicaciones. El nombre del archivo de texto de salida está definido por el usuario. Si no se especifica ningún nombre, se utiliza la salida estándar.

Conceptos relacionados:

- "Archivos DAD para colecciones XML" en la página 171

Tareas relacionadas:

- "Alteración temporal dinámica de valores en el archivo DAD" en la página 180
- "Creación de un archivo DAD para columnas XML" en la página 169
- "Utilización del comprobador de DAD" en la página 187

Utilización del comprobador de DAD

Requisitos previos:

Debe tener JRE o SDK Versión 1.3.1 o posterior instalado en el sistema.

Procedimiento:

Para utilizar el comprobador de DAD:

1. Baje el archivo DADChecker.zip y extraiga todos los archivos en el directorio que desee.
2. Desde una línea de mandatos, vaya al subdirectorio `/bin` en el directorio donde ha instalado el comprobador de DAD.
3. Establezca la vía de acceso de clase ejecutando el archivo setCP.bat, que se encuentra en el directorio `/bin`.
4. Ejecute el siguiente mandato:

   ```
   java dadchecker.Check_dad_xml [-dad | -xml] [-all][-tag nombrecódigo]
   [-out archivoSalida]
   archivoParaComprobar
   ```

   Donde:

   -dad
   indica que el archivo que se debe comprobar es un archivo DAD. Esta es la opción por omisión.

   -xml
   indica que el archivo que se debe comprobar es un documento XML en vez de un archivo DAD. Para documentos XML grandes, es posible que la máquina virtual de Java se quede sin memoria, lo cual produciría una excepción tipo java.lang.OutOfMemoryError. En estos casos, se puede utilizar la opción -Xmx para asignar más memoria a Java Virtual Machine. Consulte la documentación del SDK para obtener más detalles.

   -all
   indica que la salida mostrará todas las apariciones de códigos que tengan errores.

   -tag
   indica que sólo se deben mostrar los códigos duplicados cuyos valores de atributo de nombre sean *nombre_código*. Para los documentos XML, sólo se visualizan los códigos duplicados cuyo nombre sea el nombre de código.

   -out
   archivo_salida especifica el nombre del archivo de texto de salida. En caso de que se omita, se utiliza la salida estándar. También se creará un segundo archivo de salida, `errorsOutput.xml` en el mismo directorio que el archivo DAD. Este archivo siempre se genera y contiene en el formato XML la misma información que el texto de salida, excepto los avisos y errores del analizador.

Para visualizar las opciones de la línea de mandatos, escriba `java dadchecker.Check_dad_xml help`.

Para visualizar información de la versión, escriba `java dadchecker.Check_dad_xml version`.

Archivos de ejemplo para el comprobador DAD:

Los siguientes archivos de ejemplo se pueden encontrar en el directorio `samples`:

bad_dad.dad
archivo DAD de ejemplo que muestra todos los posibles errores de semántica.

bad_dad.chk
archivo de texto de salida generado por el comprobador de DAD para bad_dad.dad.

bad_dad.chk
archivo de texto de salida generado por el comprobador de DAD para bad_dad.dad.

errorsOutput.xml
archivo XML de salida generado por el comprobador de DAD para bad_dad.dad.

dup.xsl
hoja de estilo XSL utilizada para transformar el archivo errorsOutput.xml en un archivo HTML que sólo muestra los códigos duplicados.

dups.html
archivo HTML generado que sólo muestra los códigos duplicados contenidos en bad_dad.dad.

Errores y avisos del archivo de texto de salida:

Los errores y avisos se indican mediante la aparición de códigos. Se considera que dos códigos son apariciones del mismo código si:

- Sus atributos de nombre tienen el mismo valor.
- Tienen el mismo número de ascendientes.
- Los atributos de nombre de los códigos ascendientes correspondientes tienen el mismo valor.

Las apariciones del mismo código podrían tener potencialmente códigos hijos diferentes.

Las apariciones de códigos que no siguen las normas semánticas de DAD se indican en el archivo de texto del siguiente modo:

- Todos los códigos ascendientes y los atributos se visualizan secuencialmente.
- Se visualiza el código que es erróneo, precedido por un número que indica la profundidad en la estructura de árbol XML. El nombre de código va seguido de una lista de números de línea donde todas las apariciones del código aparecen en el archivo DAD. Puede visualizar cada aparición de error por separado utilizando la opción de la línea de mandatos *-all*.
- Se visualizan los códigos hijos directos de la primera aparición de código. Para aquellos códigos hijos que especifican una correlación de datos, también se visualizan los códigos de correlación de datos. Puede utilizar la opción de la línea de mandatos *-all* para visualizar cada aparición de error por separado.

Ejemplo de un informe de error del comprobador de DAD:

En este ejemplo, el código de element_node cuyo atributo de nombre tiene el valor "Password" es erróneo. Existen dos apariciones de este código en el archivo DAD, uno en la línea 49 y uno en la línea 75. El código erróneo se puede identificar en la

lista de códigos ancestros e hijos localizando el indicador de profundidad del código (en este ejemplo, 4). La lista de códigos ancestros e hijos ayuda a establecer el contexto en el que se ha producido el error.

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4       <element_node name="Password">  line(s): 49 75
        <element_node name="Pswd1">
     <element_node name="Pswd2">
```

Si se hubiera utilizado la opción all, el archivo de texto de salida tendría un aspecto como el siguiente:

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4         <element_node name="Password">  line: 49
        <element_node name="Pswd1">
        <element_node name="Pswd2">


<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4      <element_node name="Password">  line: 75
        <element_node name="Pswd1">
    <element_node name="Pswd3">
```

En este ejemplo, dos apariciones tienen idénticos ascendientes y valores de atributo, pero diferentes elementos hijo.

Verificaciones realizadas por el comprobador de DAD

Al invocar el comprobador de DAD, se recibe el siguiente mensaje:

```
Comprobando documento DAD: vía_archivo
```

donde *vía_archivo* es la vía que conduce al archivo DAD que se está validando.

El comprobador de DAD realiza las siguientes comprobaciones de validación:

1. Comprobación de la correcta formación y validación de DTD.
2. Detección de <attribute_node> duplicado y <element_node> suelto (correlación de RDB_node).
3. Detección de atributo de tipo que falta.
4. Detección de declaración de tabla que falta.
5. Detección de <text_node> o <attribute_node> que falta.
6. Comprobación del orden de correlación de <attribute_node> y <element_node>.
7. Comprobación de la coherencia de correlación en los datos para buscar los códigos con valores de atributo de nombre idénticos.
8. Comprobación del valor de atributo multi_ocurrence para buscar el <element_node> principal con los hijos correlacionados (correlación de RDB_node).
9. Comprobación del posible conflicto de denominación de atributos y elementos (documentos XML).

Estas comprobaciones de validación se describen en las siguientes secciones.

Correcta formación y validación de DTD

Los archivos DAD deben validarse con la DTD de DAD, que se encuentra en "c:*instalación_dxx*\samples\db2xml\dtd\dad.dtd". Si el archivo DAD no está formado correctamente o si no puede hallarse la DTD, se produce un error muy grave que ocasiona la interrupción del comprobador de DAD y que se indica en el archivo de texto de salida. Por ejemplo:

```
org.xml.sax.SAXException: Detención tras error muy grave,
línea 1, col 22. La declaración XML debe finalizar por "?>".
```

Los errores y avisos de validación también deben notificarse en el archivo de texto de salida, pero no hacen que el comprobador de DAD termine. El ejemplo siguiente es un fragmento de un archivo de texto de salida que muestra dos posibles errores de validación que se pueden encontrar al analizar el archivo DAD:

```
 ** El documento no es válido respecto a la DTD, col 15.
Debe declararse el tipo de elemento "XCollection"

** El documento no es válido respecto a la DTD, línea 578, col 21.
El contenido del tipo de elemento "text_node" debe coincidir
con "(column|RDB_node)".
```

Detección de <attribute_node> duplicado y <element_node> suelto (correlación de RDB_node)

Esta comprobación sólo es importante para los archivos DAD que utilizan la correlación de RDB_node.

Se considera que dos elementos son duplicados si dos o más códigos <attribute_node> o <element_node> tienen el mismo valor en el atributo de nombre y tienen el mismo ascendiente.

Se considera que dos o más códigos tienen los mismos ascendientes si los atributos de nombre de los códigos ascendientes correspondientes tienen el mismo valor.

Un <element_node> suelto es un nodo de elemento (element_node) que se utiliza para correlacionar un código que no tiene códigos hijos en la estructura de árbol del documento XML. Por consiguiente, los códigos sueltos <element_node> deben tener un código de nodo de texto como uno de los hijos directos. Ningún otro código <element_ node> puede tener códigos de nodo de texto como hijos directos.

Este conflicto puede surgir entre dos códigos de hoja <element_node>, entre dos o más códigos <attribute_node> o entre lo códigos de hoja <element_node> y <attribute_node>.

Ejemplos:

Ejemplo 1:

Conflicto de códigos <element_node> sueltos:

```
<element_node name = "A1">
    <element_node name = "B">
       <element_node name = "C">
       <text_node
          ....
<element_node name = "A2">
    <element_node name = "B">
```

```
       <element_node name = "C">
          <text_node>
       ....
</element_node>
```

En este ejemplo, <element_node name = ″C″> está duplicado porque está correlacionado a través de dos vías distintas: \A1\B\C y \A2\B\C. Tenga en cuenta que <element_node name=″B″> no se considera que esté duplicado, porque es un <element_node> no suelto.

Ejemplo 2:

Este ejemplo muestra un conflicto de <attribute node>.

```
<element_node name = "A1">
<attribute_node name = "B">
                  ....
<element_node name = "A2">
<attribute_node name = "B">
 /element_node>               ....
<
```

En este ejemplo, <attribute_node name = ″B″> está duplicado, porque se correlaciona a través de dos vías diferentes: \A1\B y \A2\B.

Ejemplo 3:

Este ejemplo muestra un conflicto entre <element_node> y <attribute_node>.

```
<element_node name = "A">
    <element_node name = "B">
          <text_node>
            ....
       </element_node>
       </element_node>
            ....
<attribute_node name = "B">
            ....
<attribute_node name = "A">
            ....
```

En este ejemplo, <element_node name = ″B″> está en conflicto con <attribute_node name = ″B″>. Tenga en cuenta que <element_node name = ″A″> y <attribute_node name = ″A″> no están en conflicto, porque <element_node name = ″A″> no es un código suelto <element_node>.

Si se producen conflictos, debe revisarse la DTD del documento XML para eliminar los conflictos. El documento XML y el archivo DAD también deben revisarse para reflejar los cambios en la DTD.

Ejemplo 4:

```
Se han encontrado 7 conflictos de denominación duplicados
Un total de 16 códigos son erróneos (apariciones acumuladas de estos códigos: 20)

Los siguientes códigos están duplicados:

<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4       <element_node name="Country">  line(s): 127 135
          <text_node>
              <RDB_node>
```

```
             <table name="advertiser">
               <column type="VARCHAR(63)" name="country">

------------
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
     <element_node name="Campaign" multi_occurrence="YES">
      <element_node name="Target" multi_occurrence="YES">
       <element_node name="Location" multi_occurrence="YES">
7         <element_node name="Country">  line(s): 460
            <text_node>
                    <RDB_node>
               <table name="target_location">
               <column type="VARCHAR(63)" name="country">
-------------------------------------------------------------
```

Los códigos erróneos están agrupados según el conflicto de denominación. Los grupos están separados por líneas y los códigos por líneas cortas. También puede visualizar todas las apariciones de error utilizando la opción de la línea de mandatos *all*.

Si no hay ningún duplicado en el archivo DAD, se graba el siguiente mensaje en el archivo de texto de salida:

```
No se ha encontrado ningún código duplicado.
```

Detección de atributo de tipo que falta

Al utilizar un archivo DAD para habilitar una colección o para la descomposición, debe especificarse el atributo de tipo para cada código <column>. Por ejemplo:

```
<column name="email" type="varchar(20)">
```

El mandato enable_collection utiliza las especificaciones del tipo de columna para crear las tablas de la colección si las tablas no existen. Si las tablas existen, el tipo especificado en la DAD debe coincidir con el tipo de columna real de la base de datos.

Ejemplo:

El siguiente ejemplo es un fragmento de un archivo de texto de salida que muestra los códigos <column> que no tienen el atributo de tipo.

```
Si se va a utilizar esta DAD para la descomposición o para habilitar una
colección, faltan los atributos de tipo para el código o los códigos <column>
siguientes:
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
      <element_node name="Address">
            <text_node>
              <RDB_node>
7           <column name="address">  line: 86
```

Si no falta ningún atributo de tipo, se graba el siguiente mensaje en el archivo de texto de salida:

```
No falta ningún atributo de tipo para los códigos <column>.
```

Detección de declaración de tabla que falta

El primer código <RDB_node> del archivo DAD debe incluir la declaración de tabla, incluidos todos los códigos <table> que declaran las tablas relacionales que

se utilizan para la correlación de datos. Este código debe incluirse en el primer código <element_node>. Todos los códigos <RDB_node> posteriores deben incluirse en un código <text_node>.

También se añade un error al archivo de salida si el primer código<RDB_node> encontrado contiene un código <column>. Este error indica que falta la declaración de tabla o bien que la declaración de tabla contiene de forma incorrecta un código <column>.

Detección de <text_node> o <attribute_node> que falta

Cada código <RDB_node> distinto del primero, que se utiliza para la declaración de tabla, debe incluirse en un código <attribute_node> o un código <text_node>.

Ejemplos:

Ejemplo 1:

```
 <element_node name ="amount">
            <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
           <column name="amount" type="decimal(8,2)"/>
          </RDB_node>
       </element_node>
```

Ejemplo 2:

El siguiente ejemplo es un fragmento de un archivo de texto de salida que muestra un código <text_node> o <attribute_node> que falta:

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
      <element_node name="PostalCode">
5         <RDB_node>  line: 107
             <table name="advertiser">
        <column type="VARCHAR(10)" name="postal_code">
```

Comprobación del orden de correlación de <attribute_node> y <element_node>

Esta comprobación es necesaria para FixPak 3 y las versiones anteriores. Los códigos <attribute_node> se han de correlacionar con una tabla antes de que los códigos <element_node> se correlacionen con la tabla.

Ejemplo:

El siguiente ejemplo muestra códigos que deben correlacionarse con una tabla.

```
<element_node name="payment-request"
multi_occurrence="YES">
  <element_node name="payment-request-id">
            <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
            <column name="statement_id" type="varchar(30)"/>
                    ....
<element_node name="bank-customer-info">
    <element_node name="account">
          <attribute_node name="type">
            <text_node>
```

```
  <RDB_node>
<table name="fakebank.payments"/>
     <column name="payor_account" type="char(6)"/
```

En este ejemplo, <attribute_node name=″type″> se correlaciona con la misma tabla (fakebank.payments) como <element_node name = ″payment-request-id″>. La correlación del <attribute_node> debe preceder a la correlación de <element_node>.

Comprobación de la coherencia de correlación en los datos para buscar los códigos con valores de atributo de nombre idénticos

Dentro del archivo DAD, todos los códigos <element_node> y todos los códigos <attribute_node> que están correlacionados e identificados por valores de atributo de nombre diferentes sólo deben correlacionarse una vez. Si dos o más apariciones de un código <element_node> o un código <attribute_node> están correlacionados con columnas diferentes, a los atributos de nombre se deben asignar valores diferentes.

Ejemplo:

Ejemplo 1: En este ejemplo, el código <element_node name=″type″> tiene una correlación distinta de la primera aparición. Los códigos duplicados <attribute_node> y sueltos duplicados <element_node> no se visualizan como resultado de esta comprobación.

```
<element_node name="bank-customer-info">
    <element_node name="account">
      <element_node name="type">
            <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
              <column name="payor_account" type="char(20)
        </RDB_node>
         </text_node>
       </element_node>
      <element_node>
<element_node>
<element_node name="bank-customer-info">
    <element_node name="account">
      <element_node name="type">
            <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
              <column name="payto_account" type="char(20)"/>
        </RDB_node>
         </text_node>
       </element_node>
       </element_node>
<element_node>
```

Puede arreglar este error creando un elemento nuevo para utilizar con la segunda correlación. También es necesario cambiar la DTD, el documento XML y el archivo DAD.

Ejemplo 2: Este ejemplo es un fragmento de un archivo de texto de salida que indica que los códigos <element_node> tienen los mismos nombres y ancestros, pero no las mismas correlaciones.

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4      <element_node name="PostalCode">  line(s): 127
```

```
            <text_node>
              <RDB_node>
               <table name="advertiser">
         <column type="VARCHAR(10)" name="postal_code">
----------------
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4         <element_node name="PostalCode">  line(s): 135 143
            <text_node>
              <RDB_node>
              <table name="advertiser">
               <column type="VARCHAR(10)" name="postal_code2">
```

En este ejemplo, una aparición del <element_node name=″PostalCode″> en la línea 127 está correlacionada con la columna 'postal_code' y dos apariciones del mismo código, en las líneas 135 y 143 están correlacionadas con la columna 'postal_code2'.

Comprobación del valor de atributo multi_occurrence para buscar el <element_node> principal con los hijos correlacionados

Esta comprobación sólo es importante en los archivos DAD que utilizan la correlación de RDB-node.

El valor por omisión del atributo multi_occurrence es NO. Debe asignarse al multi_occurrence el valor YES para cada código <element_node> que tenga, como hijos directos, un código <attribute_node> o dos o más códigos <element_node> que cumplan uno o dos de los siguientes criterios:

- El <element_node> está correlacionado (tiene un <text_node> como su hijo directo).
- El <element_node> tiene al menos un <attribute_node> como hijo directo.

Ejemplo:

Ejemplo 1: En el siguiente ejemplo, payment-request-id y amount están correlacionados con una tabla de DB2 UDB. El remitente tiene un <attribute_node> como hijo directo. Payment-request-id, amount y sender son todos hijos directos depayment-request:

```
<element_node name="payment-request" multi_occurrence="YES">
  <element_node name="payment-request-id">
            <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
            <column name="statement_id" type="varchar(30)"/>
        </RDB_node>
         </text_node>
  </element_node>
 <element_node name ="amount">
            <text_node>
              <RDB_node>
            <table name="fakebank.payments"/>
           <column name="amount" type="decimal(8,2)"/>
        </RDB_node>
         </text_node>
  </element_node>
  <element_node name ="sender">
     <attribute_node name ="ID">
              <RDB_node>
            <table name="fakebank.payments"/>
            <column name="sender_ID" type="decimal(8,2)"/>
```

```
        </RDB_node>
      </attribute_node>
  </element_node>
  </element_node>
```

El comprobador de DAD indica todos los códigos <element_node> cuyos atributos multi_occurrence tienen el valor NO.

Ejemplo 2: El siguiente ejemplo es un fragmento de un archivo de texto de salida que sugiere códigos <element_node> cuyos atributos multi_occurrence deben tener el valor YES.

```
<DAD>
 <Xcollection>
   <root_node>
     <element_node name="Advertiser" multi_occurrence="YES">
4       <element_node name="Password">  line(s): 49 75
        <element_node name="Pswd1">
        <element_node name="Pswd2">
```

Conflicto de denominación de atributos y elementos

En los documentos XML, los elementos con el mismo nombre pueden aparecer en diferentes contextos, como, por ejemplo, tener diferentes elementos ascendientes. Los atributos y los elementos pueden tener nombres idénticos.

El comprobador de DAD se puede utilizar para comprobar en los documentos XML. Si es necesario correlacionar más de uno de los elementos o atributos en conflicto, los cambios de denominación deben efectuarse en el documento y la DTD.

Es mejor comprobar el documento XML antes de crear el archivo DAD. El comprobador de DAD no valida el documento XML respecto a la DTD.

Ejemplo:

El ejemplo siguiente es un fragmento de un documento XML donde se producen conflictos de denominación:

```
<A1>
   <B>
      <C>
          ....
<A2>
   <B>
      <C>
          ....
<D C="attValue">
.....

Si se correlacionaran el elemento <C> y el atributo C, el archivo DAD
resultante tendría los siguientes conflictos de duplicados:
<element_node name = "A1">
    <element_node name = "B">
       <element_node name = "C">
            <text_node>
            .....
<element_node name = "A2">
    <element_node name = "B">
       <element_node name = "C">
            <text_node>
            ....
```

```
    <element_node name = "D">
       <attribute_node name = "C">
        ....
</element_node>
```

Los dos códigos <element_node name = ″C″> y el código <attribute_node name = ″C″> están duplicados en la DAD.

Capítulo 10. Procedimientos almacenados de XML Extender

Procedimientos almacenados de XML Extender - Visión general

XML Extender proporciona procedimientos almacenados para la administración y gestión de columnas y colecciones XML. Estos procedimientos almacenados se pueden invocar desde el cliente DB2. La interfaz cliente puede estar incorporada en SQL, ODBC o JDBC. Consulte la sección sobre procedimientos almacenados en la publicación *DB2 Administration Guide* para obtener detalles acerca de cómo llamar a procedimientos almacenados.

Los procedimientos almacenados utilizan el esquema DB2XML, que es el nombre de esquema de XML Extender.

XML Extender proporciona tres tipos de procedimientos almacenados:

Procedimientos almacenados de administración
ayudan a los usuarios a realizar tareas administrativas

Procedimientos almacenados de composición
general documentos XML utilizando los datos de tablas de bases de datos existentes

Procedimientos almacenados de descomposición
fragmentan o desglosan los documentos XML de entrada y almacenan datos en tablas de bases de datos nuevas o existentes

Asegúrese de incluir los archivos de cabecera externos de XML Extender en el programa que invoca procedimientos almacenados. Los archivos de cabecera se encuentran en el directorio *"$instalación_dxx$*\dxx\samples\db2xml\include" , donde *$instalación_dxx$* es el directorio donde DB2 XML Extender se encuentra instalado.. Los archivos de cabecera son:

dxx.h Tipos de datos y constantes definidos de XML Extender

dxxrc.h Código de retorno de 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 esté especificada en el archivo makefile mediante la opción de compilación.

Invocación de procedimientos almacenados de XML Extender

Puede utilizar XML Extender en distintos sistemas operativos desde una única aplicación cliente, escribiendo los nombres de los procedimientos almacenados tanto en mayúsculas como en minúsculas. Para llamar a los procedimientos almacenados de esta manera, utilice las versiones `result_colname` y `valid_colname` de los procedimientos almacenados de composición. La utilización de este método le proporciona las ventajas siguientes:

- Puede utilizar estos procedimientos almacenados en todos los entornos de DB2 Universal Database porque puede incluir muchas columnas en la tabla de

resultados. Las versiones de los procedimientos almacenados que no dan soporte a `result_colname` y `valid_colname` requieren exactamente una columna en la tabla de resultados.

- Puede utilizar una tabla temporal declarada en la tabla de resultados. La tabla temporal se identifica mediante un esquema que se establece como ″session″. Las tablas temporales declaradas le permiten dar soporte a entornos de cliente de múltiples usuarios.

Utilice las mayúsculas cuando llame a los procedimientos almacenados de DB2 XML Extender para acceder a los procedimientos de forma coherente en distintas plataformas.

Requisitos previos: Vincule la base de datos con el procedimiento almacenado XML Extender y los archivos de vinculación para la CLI de DB2 UDB. Puede utilizar un archivo de mandatos de ejemplo, `getstart_prep.cmd`, para vincular los archivos. Este archivo de mandatos está en el directorio ″c:*instalación_dxx*\samples\db2xml\cmd″. Para vincular:

1. Conéctese a la base de datos. Por ejemplo:
   ```
   db2 "connect to SALES_DB"
   ```
2. Pase al directorio ″c:*instalación_dxx*\samples\db2xml\bnd″ y vincule XML Extender con la base de datos.
   ```
   db2 "bind @dxxbind.lst"
   ```
3. Pase al directorio ″c:*instalación_dxx*\samples\db2xml\bnd″ y vincule la CLI con la base de datos.
   ```
   db2 "bind @db2cli.lst"
   ```
4. Finalice la conexión.
   ```
   db2 "terminate"
   ```

Procedimiento:

Invoque XML Extender utilizando la siguiente sintaxis:

```
CALL DB2XML.function_entry_point
```

Donde:

punto_entrada_función
Especifica el nombre de la función.

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.

Encontrará ejemplos para llamar a los procedimientos almacenados en los directorios `instalación_dxx/samples/db2xml/c` e `instalación_dxx/samples/db2xml/cli`. En el directorio `instalación_dxx/samples/db2xml/c`, se proporcionan archivos de código de SQX para llamar a procedimientos almacenados de colección XML con SQL incorporado. En el directorio `instalación_dxx/samples/db2xml/cli`, los archivos de ejemplo muestran cómo llamar a los procedimientos almacenados con la CLI (Call Level Interface).

Aumento del límite CLOB para procedimientos almacenados

El límite por omisión de los parámetros CLOB cuando se pasa a un procedimiento almacenado es de 1 MB. Puede aumentar el límite.

Procedimiento:

Para aumentar el límite de CLOB:

1. Elimine cada procedimiento almacenado. Por ejemplo:

```
db2   "drop procedure DB2XML.dxxShredXML restrict"
```

2. Cree un nuevo procedimiento con el límite de CLOB aumentado. Por ejemplo, para aumentar el límite CLOB a 10 megabytes:

```
db2   "create procedure DB2XML!dxxShredXML(in     dadBuf      clob(100K),
                                           in     XMLObj      clob(10M),
                                           out    returnCode  integer,
                                           out    returnMsg   varchar(1024)
                                           )
         external name 'DB2XML.dxxShredXML_STP'
         language C
         parameter style SQL
         not deterministic
         fenced
         null call"
```

Tareas relacionadas:

- "Invocación de procedimientos almacenados de XML Extender" en la página 199
- "Procedimientos almacenados que devuelven CLOB" en la página 201

Procedimientos almacenados que devuelven CLOB

Si tiene archivos CLOB que sean mayores de 1 MB, puede redefinir el parámetro del procedimiento almacenado creando y ejecutando un archivo que contenga los siguientes mandatos:

```
drop procedure db2xml.dxxGenXMLClob;

create procedure db2xml.dxxGenXMLClob(
          in     dadBuf        clob(100K),
          in     overrideType  integer,
          in     override      varchar(32672),
          out    resultDoc     clob(1M),
          out    valid         integer,
          out    numDocs       integer,
          out    returnCode  integer,
          out    returnMsg   varchar(1024)
          )
          external name 'db2xml!dxxGenXMLClob'
          specific DB2XML.DXXGENXMLCLOB
       language C
          parameter style DB2DARI
       not deterministic
       fenced
       null call;
drop procedure db2xml.dxxRetrieveXMLClob;

create procedure db2xml.dxxRetrieveXMLClob(
          in   collectionName    varchar(128),
          in   overrideType      integer,
          in   override          varchar(32672),
          out  resultDoc         clob(1M),
```

```
   out  valid             integer,
   out  numDocs          integer,
   out  returnCode       integer,
   out  returnMsg        varchar(1024)
   )
   external name 'db2xml!dxxRetrieveXMLClob'
   specific DB2XML.DXXRETRIEVEXMLCLOB
language C
   parameter style DB2DARI
not deterministic
fenced
```

para especificar la longitud CLOB: Abra el archivo en un editor y modifique el parámetro *resultDoc* como se muestra en el siguiente ejemplo:

```
out resultDoc clob(clob_tamaño),
```

Si se genera más de un documento, el procedimiento almacenado devuelve el mismo documento.

Recomendación de tamaño: El límite de tamaño del parámetro *resultDoc* depende de la configuración del sistema. Tenga en cuenta que la cantidad especificada en este parámetro es la cantidad asignada por JDBC, independientemente del tamaño del documento. Este tamaño debe acomodar los archivos XML más grandes, pero no debe exceder 1,5 gigabytes.

Para ejecutar el mandato desde la línea de mandatos de DB2 y el directorio donde se encuentra el archivo, entre:

```
db2 -tf crtgenxc.db2
```

Tareas relacionadas:

• "Invocación de procedimientos almacenados de XML Extender" en la página 199

Procedimientos almacenados de administración de XML Extender

Procedimientos almacenados de administración de XML Extender - Visión general

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 de XML Extender y el mandato de administración **dxxadm**.

- dxxEnableDB()
- dxxDisableDB()
- dxxEnableColumn()
- dxxDisableColumn()
- dxxEnableCollection()
- dxxDisableCollection()

Procedimiento almacenado dxxEnableDB()

Propósito:

Habilita la base de datos. Cuando se habilita la base de datos, XML Extender crea los objetos siguientes:

- Los tipos definidos por el usuario (UDT) de XML Extender.
- Las funciones definidas por el usuario (UDF) de XML Extender.
- La tabla de depósito de DTD de XML Extender, DTD_REF, que almacena las DTD e información acerca de cada DTD.
- La tabla de uso de XML Extender, XML_USAGE, que almacena información común para cada columna que está habilitada para XML y para cada colección.

Sintaxis:

```
DB2XML.dxxEnableDB(char(dbName) dbName,       /* entrada */
            long           returnCode,   /* salida */
            varchar(1024)  returnMsg)    /* salida */
```

Parámetros:

Tabla 47. Parámetros de dxxEnableDB()

Parámetro	Descripción	Parámetro de E/S
dbName	Es el nombre de la base de datos.	ENTRADA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Conceptos relacionados:

- "Procedimientos almacenados de administración de XML Extender - Visión general" en la página 202
- Capítulo 13, "Tablas de soporte de administración de XML Extender", en la página 279

Tareas relacionadas:

- "Habilitación de bases de datos para XML" en la página 55
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxDisableDB()

Propósito:

Inhabilita la base de datos. Cuando XML Extender inhabilita la base de datos, elimina los objetos siguientes:

- Los tipos definidos por el usuario (UDT) de XML Extender.
- Las funciones definidas por el usuario (UDF) de XML Extender.
- La tabla de depósito de DTD de XML Extender, DTD_REF, que almacena las DTD e información acerca de cada DTD.
- La tabla de uso de XML Extender, XML_USAGE, que almacena información común para cada columna que está habilitada para XML y para cada colección.

Importante: Debe inhabilitar todas las columnas XML antes de intentar inhabilitar una base de datos. XML Extender no puede inhabilitar una base de datos que contiene columnas o colecciones que están habilitadas para XML.

Sintaxis:

```
DB2XML.dxxDisableDB(char(dbName)     dbName,  /* entrada */
                    long          returnCode,   /* salida */
                    varchar(1024) returnMsg)    /* salida */
```

Parámetros:

Tabla 48. Parámetros de dxxDisableDB()

Parámetro	Descripción	Parámetro de E/S
dbName	Es el nombre de la base de datos.	ENTRADA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Conceptos relacionados:

- "Procedimientos almacenados de administración de XML Extender - Visión general" en la página 202
- Capítulo 13, "Tablas de soporte de administración de XML Extender", en la página 279

Tareas relacionadas:

- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxEnableColumn()

Propósito:

Habilita una columna XML. Cuando habilita una columna, XML Extender realiza las siguientes tareas:

- Determina si la tabla XML tiene una clave primaria; si no la tiene, XML Extender modifica la tabla XML y añade una columna denominada 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 especificado por el usuario o el valor DXXROOT_ID especificado por XML Extender.
- Crea una vista por omisión para la tabla XML y sus tablas auxiliares, utilizando opcionalmente un nombre especificado por el usuario.

Sintaxis:

```
DB2XML.dxxEnableColumn(char(dbName) dbName,       /* entrada */
                       char(tbName) tbName,       /* entrada */
                       char(colName) colName,     /* entrada */
                       CLOB(100K)     DAD,       /* entrada */
                       char(tablespace) tablespace,  /* entrada */
```

```
char(defaultView) defaultView,     /* entrada */
char(rootID) rootID,               /* entrada */
long          returnCode,          /* salida */
varchar(1024) returnMsg)              /* salida */
```

Parámetros:

Tabla 49. Parámetros de dxxEnableColumn()

Parámetro	Descripción	Parámetro de E/S
dbName	Es el nombre de la base de datos.	ENTRADA
tbName	Es el nombre de la tabla donde reside la columna XML.	ENTRADA
colName	Es el nombre de la columna XML.	ENTRADA
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
tablespace	Es el espacio de tabla, distinto del espacio de tabla por omisión, que contiene las tablas auxiliares. Si no se especifica este parámetro, se utiliza el espacio de tabla por omisión.	ENTRADA
defaultView	Es el nombre de la vista por omisión que une la tabla de aplicación y las tablas auxiliares.	ENTRADA
rootID	Es el nombre de la clave primaria, contenida en la tabla de la aplicación, que se utilizará como ID raíz para la tabla auxiliar.	ENTRADA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Conceptos relacionados:

- "Columnas XML como método de almacenamiento y acceso" en la página 76
- "Procedimientos almacenados de administración de XML Extender - Visión general" en la página 202

Tareas relacionadas:

- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxDisableColumn()

Propósito:

Inhabilita la columna habilitada para XML. Cuando la columna XML está inhabilitada, ya no puede contener tipos de datos XML.

Sintaxis:

```
DB2XML.dxxDisableColumn(char(dbName) dbName,        /* entrada */
                char(tbName) tbName,                /* entrada */
                char(colName) colName,              /* entrada */
                long           returnCode,          /* salida */
                varchar(1024)  returnMsg)           /* salida */
```

Parámetros:

Tabla 50. Parámetros de dxxDisableColumn()

Parámetro	Descripción	Parámetro de E/S
dbName	Es el nombre de la base de datos.	ENTRADA
tbName	Es el nombre de la tabla donde reside la columna XML.	ENTRADA
colName	Es el nombre de la columna XML.	ENTRADA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Información relacionada:

- Apéndice C, “Límites de XML Extender”, en la página 321

Procedimiento almacenado dxxEnableCollection()

Propósito:

Habilita una colección XML que está asociada a una tabla de aplicación.

Sintaxis:

```
dxxEnableCollection(char(dbName)   dbName,       /* entrada */
                    char(colName) colName,       /* entrada */
                    CLOB(100K)     DAD,          /* entrada */
                    char(tablespace) tablespace,  /* entrada */
                    long           returnCode,  /* salida */
                    varchar(1024)  returnMsg)   /* salida */
```

Parámetros:

Tabla 51. Parámetros de dxxEnableCollection()

Parámetro	Descripción	Parámetro de E/S
dbName	Es el nombre de la base de datos.	ENTRADA
colName	Es el nombre de la colección XML.	ENTRADA
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
tablespace	Es el espacio de tabla, distinto del espacio de tabla por omisión, que contiene las tablas auxiliares. Si no se especifica este parámetro, se utiliza el espacio de tabla por omisión.	ENTRADA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA

Tabla 51. Parámetros de dxxEnableCollection() (continuación)

Parámetro	Descripción	Parámetro de E/S
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Conceptos relacionados:

- "Colecciones XML como método de almacenamiento y acceso" en la página 93
- "Procedimientos almacenados de administración de XML Extender - Visión general" en la página 202

Tareas relacionadas:

- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxDisableCollection()

Propósito:

Inhabilita una colección habilitada para XML, eliminando los marcadores que identifican tablas y columnas como integrantes de una colección.

Sintaxis:

```
dxxDisableCollection(char(dbName) dbName,       /* entrada */
                     char(colName) colName,     /* entrada */
                     long          returnCode,  /* salida */
                     varchar(1024)  returnMsg)   /* salida */
```

Parámetros:

Tabla 52. Parámetros de dxxDisableCollection()

Parámetro	Descripción	Parámetro de E/S
dbName	Es el nombre de la base de datos.	ENTRADA
colName	Es el nombre de la colección XML.	ENTRADA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Información relacionada:

- Apéndice C, "Límites de XML Extender", en la página 321

Procedimientos almacenados de composición de XML Extender

Procedimientos almacenados de composición XML Extender - Visión general

Los procedimientos almacenados de composición dxxGenXML(), dxxRetrieveXML(), dxxGenXMLCLOB() y dxxRetrieveXMLCLOB() se utilizan para generar documentos XML utilizando los datos de tablas de bases de datos existentes. 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.

Se han efectuado las mejoras siguientes en el rendimiento para los procedimientos almacenados de composición.

- En los sistemas operativos UNIX® y Windows®, la longitud del parámetro de alteración temporal se ha aumentado de 1KB a 32KB.

  La alteración temporal de 1 KB ha impuesto una restricción en la longitud de la sentencia de SQL para la composición SQL. La restricción hace recomendable la utilización de vistas de bases de datos para reducir la longitud de la sentencia de SQL necesaria. Sin embargo, a veces las vistas de bases de datos pueden incurrir en una longitud adicional de vía de acceso debido a la materialización de la vista. Con una alteración temporal larga, se reduce la fuerte necesidad de vistas.
- Se ha eliminado el requisito de una tabla de resultados intermedios.
- Utilizando estos procedimientos almacenados:
  - Reduce la longitud de vía de acceso de instrucciones porque ya no hay la necesidad de crear tablas de resultados.
  - Puede simplificar la programación.
- Utilice los procedimientos almacenados que requieren una tabla de resultados intermedios si desea producir más de un documento.
- Se ha mejorado el rendimiento las funciones definidas por el usuario para la columna XML.
- Las funciones definidas por el usuario de DB2® UDB XML Extender mantendrán ahora los documentos XML pequeños (512KB) en memoria mientras los procesan. Esto reduce la actividad de entrada/salida y la contención para el disco que se utiliza para los archivos temporales.
- Se ha modificado la definición de las funciones definidas por el usuario escalares (no de tabla) de DB2 UDB XML Extender para permitir su ejecución en paralelo. Este cambio proporciona mejoras sustanciales del rendimiento en la ejecución de consultas que se refieran más de una vez a las funciones definidas por el usuario. Deberá ejecutar el programa de script de migración para obtener la capacidad de paralelismo de las UDF escalares. Si ya tiene columnas habilitadas mediante las UDF escalares, deberá inhabilitar todas las columnas, ejecutar el script de migración y, luego, volver a habilitar las columnas.

Procedimiento almacenado 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 *override*. Basándose en el valor de *overrideType*, la aplicación puede alterar temporalmente la sentencia de SQL_stmt, para la correlación de SQL, o las condiciones del RDB_node, para la correlación de RDB_node, en el archivo DAD. El parámetro de entrada *overrideType* se utiliza para clarificar el tipo de *override (alteración temporal)*.

Sintaxis:

```
dxxGenXML(CLOB(100K)     DAD,                       /* entrada */
          char(resultTabName) resultTabName,        /* entrada */
          char resultColName, char resultValidCol   /* entrada */

          char(30)       valid_column,                 /* entrada */
          integer        overrideType                 /* entrada */
          varchar(1024)  override,                    /* entrada */
          integer        maxRows,                     /* entrada */
          integer        numRows,                     /* salida */
          long           returnCode,                  /* salida */
          varchar(1024)  returnMsg)                   /* salida */
```

Donde valor_varchar es 32672 para Windows y UNIX, y 16366 para iSeries y z/OS.

Parámetros:

Tabla 53. Parámetros de dxxGenXML()

Parámetro	Descripción	Parámetro de E/S
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
resultTabName	Es el nombre de la tabla resultante, que debe existir antes que la llamada. La tabla contiene una sola columna, de tipo XMLVARCHAR o XMLCLOB.	ENTRADA
result_column	Nombre de la columna de la tabla de resultados en la que se almacenan los documentos XML compuestos.	ENTRADA
valid_column	Nombre de la columna que indica si el documento XML es válido cuando se valida con una definición de tipo de documento (DTD).	ENTRADA

Tabla 53. Parámetros de dxxGenXML() (continuación)

Parámetro	Descripción	Parámetro de E/S
overrideType	Distintivo para indicar el tipo del siguiente parámetro *override*: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal mediante una sentencia de SQL_stmt. • **XML_OVERRIDE**: Alteración temporal mediante una condición basada en XPath.	ENTRADA
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar dicha correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de 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 utiliza este parámetro *overrideType*, deberá utilizar dicha correlación de RDB_node en el archivo DAD.	ENTRADA
resultDoc	Un CLOB que contiene el documento XML compuesto.	SALIDA
valid	Este parámetro se establece del modo siguiente: • Si VALIDATION=YES, valid=1 para una validación satisfactoria o valid=0 para una validación no satisfactoria. • Si VALIDATION=NO, valid=NULL.	SALIDA
maxRows	Es el número máximo de filas de la tabla resultante.	ENTRADA
numRows	Es el número real de filas creadas de la tabla resultante.	SALIDA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos:

En el fragmento de ejemplo siguiente, se crea la tabla resultante XML_ORDER_TAB, que tiene una sola columna de tipo XMLVARCHAR. Puede ver un ejemplo funcional completo en `DXXSAMPLES/QCSRC(GENX)`.

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

        EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
        SQL TYPE is CLOB(100K)  dad;      /* DAD */
        SQL TYPE is CLOB_FILE   dadFile;  /* archivo DAD */
        char    result_tab[32];           /* nombre tabla resultante */
        char    verride[2];               /* alt. temp; se estab. NULL*/
        short   overrideType;             /* definido en dxx.h */
        short   max_row;                  /* número máximo filas */
        short   num_row;                  /* número real filas */
        long    returnCode;               /* código error retorno */
        char    returnMsg[1024];          /* texto mensaje 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 SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

              /* leer datos de un archivo y ponerlos en un CLOB */
              strcpy(dadfile.name,"dxxinstall/dad/litem3.dad");
              dadfile.name_length = strlen("dxxinstall/dad/litem3.dad");
              dadfile.file_options = SQL_FILE_READ;
              EXEC SQL VALUES (:dadfile) INTO :dad;
              strcpy(result_tab,"xml_order_tab");
              override[0] = '\0';
              overrideType = NO_OVERRIDE;
              max_row = 500;
              num_row = 0;
              returnCode = 0;
              msg_txt[0] = '\0';
              collection_ind = 0;
              dad_ind = 0;
              rtab_ind = 0;
              ov_ind = -1;
              ovtype_ind = 0;
              maxrow_ind = 0;
              numrow_ind = -1;
              returnCode_ind = -1;
              returnMsg_ind = -1;

              /* 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);
```

Conceptos relacionados:

- "Procedimientos almacenados de composición XML Extender - Visión general" en la página 208

Tareas relacionadas:

- "Composición de documentos XML utilizando la correlación de SQL" en la página 62

- "Composición de colecciones XML utilizando la correlación de RDB_node" en la página 66
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix

Procedimiento almacenado dxxRetrieveXML()

Propósito:

El procedimiento almacenado dxxRetrieveXML() sirve como medio 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 *override*. Basándose en el valor de *overrideType*, la aplicación puede alterar temporalmente la sentencia de SQL_stmt para la correlación de SQL o las condiciones de RDB_node para la correlación de RDB_node del archivo DAD. El parámetro de entrada *overrideType* se utiliza para clarificar el tipo de *override (alteración temporal)*.

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.

Sintaxis:

```
dxxRetrieveXML(char(collectionName) collectionName,      /* entrada */
               char(resultTabName) resultTabName,        /* entrada */
               char resultColName, char resultValidCol   /* entrada */

               integer        overrideType,              /* entrada */
               varchar_value  override,                  /* entrada */
            integer        maxRows,                     /* entrada */
               integer        numRows,                   /* salida */
               long           returnCode,                /* salida */
               varchar(1024)  returnMsg)                 /* salida */
```

Donde *varchar_value* es 32672 para Windows y UNIX y 16366 para iSeries y z/OS.

Parámetros:

Tabla 54. Parámetros de dxxRetrieveXML()

Parámetro	Descripción	Parámetro de E/S
collectionName	Es el nombre de una colección XML habilitada.	ENTRADA
resultTabName	Es el nombre de la tabla resultante, que debe existir antes que la llamada. La tabla contiene una sola columna, de tipo XMLVARCHAR o XMLCLOB.	ENTRADA

Tabla 54. Parámetros de dxxRetrieveXML() (continuación)

Parámetro	Descripción	Parámetro de E/S
result_column	Nombre de la columna de la tabla de resultados en la que se almacenan los documentos XML compuestos.	ENTRADA
valid_column	Nombre de la columna que indica si el documento XML es válido cuando se valida con una definición de tipo de documento (DTD).	ENTRADA
overrideType	Distintivo para indicar el tipo del siguiente parámetro *override*: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal mediante una sentencia de SQL_stmt. • **XML_OVERRIDE**: Alteración temporal mediante una condición basada en XPath.	ENTRADA
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar dicha correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de 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 utiliza este parámetro *overrideType*, deberá utilizar dicha correlación de RDB_node en el archivo DAD.	ENTRADA
maxRows	Es el número máximo de filas de la tabla resultante.	ENTRADA
numRows	Es el número real de filas creadas de la tabla resultante.	SALIDA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos:

En el siguiente fragmento de ejemplo se invoca dxxRetrieveXML(). En este ejemplo, se crea la tabla resultante XML_ORDER_TAB, que contiene una sola columna de tipo XMLVARCHAR. Se encuentra un ejemplo funcional completo en instalación_dxx\samples\db2xml\qcsrc(rtrx).

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

     EXEC SQL INCLUDE SQLCA;
     EXEC SQL BEGIN DECLARE SECTION;
     char    collection[32];        /* almac. intermedio DAD */
     char    result_tab[32];        /* nombre tabla resultante */
     char    override[2];           /* alteración temporal; se estab. en NULL*/
     short   overrideType;          /* definido en dxx.h */
     short   max_row;               /* número máximo filas */
     short   num_row;               /* número real filas */
     long    returnCode;            /* código error retorno */
     char    returnMsg[1024];       /* texto mensaje 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 del 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);
```

Conceptos relacionados:

- "Procedimientos almacenados de composición XML Extender - Visión general" en la página 208

Tareas relacionadas:

- "Composición de documentos XML utilizando la correlación de SQL" en la página 62
- "Composición de colecciones XML utilizando la correlación de RDB_node" en la página 66
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxGenXMLClob

Propósito:

Como entrada, dxxGenXMLClob emplea un almacenamiento intermedio que contiene la DAD. Crea documentos XML utilizando los datos que están almacenados en las tablas de la colección XML que se especifican mediante `<Xcollection>` en la DAD y devuelve el primero y generalmente único documento XML generado en el CLOB *resultDoc*.

Sintaxis:

```
dxxGenXMLClob(CLOB(100k)       DAD                              /*entrada*/
              integer                           overrideType,    /*entrada*/
              varchar(valor_varchar)    override,                /*entrada*/
              CLOB(1M)                          resultDoc,       /*salida*/
              integer                           valid,           /*salida*/
              integer                           numDocs,         /*salida*/
              long                              returnCode,      /*salida*/
              varchar(1024)                     returnMsg),      /*salida*/
```

Donde *valor_varchar* es 32672 para Windows y UNIX y 16366 para iSeries y z/OS.

Parámetros:

Tabla 55. Parámetros de dxxGenXMLClob

Parámetro	Descripción	Parámetro de E/S
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
overrideType	Un distintivo para indicar el tipo de parámetro *override*: **NO_OVERRIDE** Sin alteración temporal. **SQL_OVERRIDE** Alteración temporal mediante SQL_stmt **XML_OVERRIDE** Alteración temporal mediante una condición basada en XPath.	ENTRADA

Tabla 55. Parámetros de dxxGenXMLClob (continuación)

Parámetro	Descripción	Parámetro de E/S
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. **NO_OVERRIDE** Una serie NULL. **SQL_OVERRIDE** Una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar esta correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de SQL_stmt especificada en el archivo DAD. **XML_OVERRIDE** Serie que contiene una o más expresiones entre comillas dobles separadas por la palabra "and". Si utiliza este parámetro con *overrideType*, deberá utilizar dicha correlación de RDB_node en el archivo DAD	ENTRADA
resultDoc	Un CLOB que contiene el documento XML compuesto.	SALIDA
valid	Este parámetro se establece del modo siguiente: • Si VALIDATION=YES, valid=1 para una validación satisfactoria o valid=0 para una validación no satisfactoria. • Si VALIDATION=NO, valid=NULL.	SALIDA
numDocs	Número de documentos XML que se generarían a partir de los datos de entrada. **Nota:** Actualmente, sólo se devuelve el primer documento.	SALIDA

Tabla 55. Parámetros de dxxGenXMLClob (continuación)

Parámetro	Descripción	Parámetro de E/S
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

El tamaño del parámetro CLOB es 1 MB. Si tiene archivos CLOB que son mayores de 1 MB, XML Extender proporciona un archivo de mandatos para redefinir el parámetro del procedimiento almacenado. Descargue el archivo `crtgenxc.zip` del sitio Web de DB2 UDB XML Extender. Este archivo ZIP contiene los programas siguientes:

crtgenxc.db2
Para su uso con XML Extender V7.2 FixPak 5 y posteriores para UNIX y Windows.

crtgenxc.iseries
Para su uso con XML Extender para iSeries

crtgenxc.zox.jci y crtgenxc.zos.cmd
Para su uso con XML Extender para OS/390 V7, APAR PQ58249 y posteriores.

Para especificar la longitud de CLOB: Abra el archivo en un editor y modifique el parámetro *resultDoc*, mostrado en el siguiente ejemplo.

```
out    resultDoc     clob(tamaño_clob),
```

Recomendación de tamaño: El límite del tamaño del parámetro *resultDoc* depende de la configuración del sistema, pero tenga en cuenta que la cantidad especificada en este parámetro es la cantidad asignada por JDBC, independientemente del tamaño del documento. Este tamaño debe acomodar los archivos XML más grandes, pero no debe exceder 1,5 gigabytes.

Para ejecutar el archivo de mandatos en UNIX o Windows, desde la línea de mandatos de DB2 UDB y desde el directorio donde se encuentra el archivo, entre:

```
db2   -tf  crtgenxc.db2
```

Conceptos relacionados:

- "Procedimientos almacenados de composición XML Extender - Visión general" en la página 208

Tareas relacionadas:

- "Composición de documentos XML utilizando la correlación de SQL" en la página 62
- "Composición de colecciones XML utilizando la correlación de RDB_node" en la página 66
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxRetrieveXMLClob

Propósito:

El procedimiento almacenado dxxRetrieveXMLClob habilita la composición de documentos a partir de datos relacionales.

Los requisitos para utilizar dxxRetrieveXMLClob son los mismos que para dxxGenXMLClob. La única diferencia es que la DAD no es un parámetro de entrada para dxxRetrieveXMLClob, sino que es el nombre de una colección XML habilitada.

Sintaxis:

```
dxxRetrieveXMLClob(char(collectionName)          collectionName  /*entrada*/
              integer                            overrideType,    /*entrada*/
              varchar(varchar_value)   override,       /*entrada*/
              CLOB(1M)                           resultDoc,       /*salida*/
              integer                            valid,           /*salida*/
              integer                            numDocs,         /*salida*/
              long                               returnCode,      /*salida*/
              varchar(1024)                      returnMsg),      /*salida*/
```

Parámetros:

Tabla 56. Parámetros de dxxRetrieveXMLClob

Parámetro	Descripción	Parámetro de E/S
collectionName	Es el nombre de una colección XML habilitada.	ENTRADA
overrideType	Un distintivo para indicar el tipo de parámetro *override*: **NO_OVERRIDE** Sin alteración temporal. **SQL_OVERRIDE** Alteración temporal con SQL_stmt **XML_OVERRIDE** Alteración temporal con una condición basada en XPath.	ENTRADA

Tabla 56. Parámetros de dxxRetrieveXMLClob (continuación)

Parámetro	Descripción	Parámetro de E/S
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. **NO_OVERRIDE** Una serie NULL. **SQL_OVERRIDE** Una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar esta correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de SQL_stmt especificada en el archivo DAD. **XML_OVERRIDE** Serie que contiene una o más expresiones entre comillas dobles separadas por la palabra "and". Si utiliza este parámetro con *overrideType*, deberá utilizar dicha correlación de RDB_node en el archivo DAD	ENTRADA
resultDoc	Es el número máximo de filas de la tabla resultante.	ENTRADA
valid	Este parámetro se establece del modo siguiente: • Si VALIDATION=YES, valid=1 para una validación satisfactoria o valid=0 para una validación no satisfactoria. • Si VALIDATION=NO, valid=NULL.	SALIDA
numDocs	Número de documentos XML que se generarían a partir de los datos de entrada. NOTA: Actualmente, sólo se devuelve el primer documento.	SALIDA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

El tamaño del parámetro CLOB es 1 MB. Si tiene archivos CLOB que son mayores de 1 MB, XML Extender proporciona un archivo de mandatos para redefinir el parámetro del procedimiento almacenado. Descargue el archivo `crtgenxc.zip` del sitio Web de DB2 UDB XML Extender. Este archivo ZIP contiene los programas siguientes:

crtgenxc.db2
Para utilizar en XML Extender V7.2 Fixpak 5 y posterior para UNIX y Windows.

crtgenxc.iseries
Para su uso con XML Extender para iSeries

crtgenxc.zox.jci y crtgenxc.zos.cmd
Para su uso con XML Extender para OS/390 V7, APAR PQ58249 y posteriores.

Para especificar la longitud de CLOB: Abra el archivo en un editor y modifique el parámetro *resultDoc*, mostrado en el siguiente ejemplo.

```
out    resultDoc     clob(tamaño_clob),
```

Recomendación de tamaño: El límite del tamaño del parámetro *resultDoc* depende de la configuración del sistema, pero tenga en cuenta que la cantidad especificada en este parámetro es la cantidad asignada por JDBC, independientemente del tamaño del documento. Este tamaño debe acomodar los archivos XML más grandes, pero no debe exceder 1,5 gigabytes.

Para ejecutar el archivo de mandatos en UNIX o Windows, desde la línea de mandatos de DB2 UDB y desde el directorio donde se encuentra el archivo, entre:

```
db2   -tf  crtgenxc.db2
```

Conceptos relacionados:

- "Procedimientos almacenados de composición XML Extender - Visión general" en la página 208

Tareas relacionadas:

- "Composición de documentos XML utilizando la correlación de SQL" en la página 62
- "Composición de colecciones XML utilizando la correlación de RDB_node" en la página 66
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimientos almacenados de descomposición de XML Extender

Procedimientos almacenados de descomposición de XML Extender - Visión general

Los procedimientos almacenados de descomposición, dxxInsertXML() y dxxShredXML(), se utilizan para desglosar o descomponer 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.

Procedimiento almacenado dxxShredXML()

Propósito:

Descompone documentos XML, basados en una correlación de archivo DAD, almacenando el contenido de los elementos y atributos XML en tablas de DB2 UDB. 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 requiere que las columnas especificadas en la condición de unión, en DAD, correspondan a las relaciones de clave primaria-clave foránea en las tablas existentes. Las columnas de la condición de unión que se han especificado en el RDB_node del nodo de elemento (element_node) raíz deben existir en las tablas.

El fragmento de procedimiento almacenado que se muestra en la presente sección es un ejemplo con fines ilustrativos.

Sintaxis:

```
dxxShredXML(CLOB(100K)    DAD,           /* entrada */
            CLOB(1M)      xmlobj,        /* entrada */
            long          returnCode,    /* salida */
            varchar(1024) returnMsg)     /* salida */
```

Parámetros:

Tabla 57. Parámetros de dxxShredXML()

Parámetro	Descripción	Parámetro de E/S
DAD	Es un CLOB donde reside el archivo DAD.	ENTRADA
xmlobj	Es un objeto de documento XML cuyo tipo es XMLCLOB.	ENTRADA
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos:

En el siguiente fragmento de ejemplo se invoca 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 del lenguaje principal e indicadores */
              strcpy(dadFile.name,"instalación_dxx
              /samples/db2xml/dad/getstart_xcollection.dad
              ");
              dadFile.name_length=strlen("instalación_dxx
              /samples/db2xml/dad/getstart_xcollection.dad
              ");
        dadFile.file_option=SQL_FILE_READ;
              strcpy(xmlFile.name,"instalación_dxx
```

```
        /samples/db2xml/xml/getstart.xml");
xmlFile.name_length=strlen("instalación_dxx
        /samples/db2xml/xml/getstart.xml")");
      xmlFile.file_option=SQL_FILE_READ;
      SQL EXEC VALUES (:dadFile) INTO :dad;
      SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
      returnCode = 0;
      returnMsg[0] = '\0';
      dad_ind = 0;
      xmlDoc_ind = 0;
      returnCode_ind = -1;
      returnMsg_ind = -1;

      /* Llamar al procedimiento almacenado */
      EXEC SQL CALL DB2XML.dxxShredXML(:dad:dad_ind;
                              :xmlDoc:xmlDoc_ind,
                              :returnCode:returnCode_ind,
                              :returnMsg:returnMsg_ind);
```

Conceptos relacionados:

- "Procedimientos almacenados de descomposición de XML Extender - Visión general" en la página 220

Tareas relacionadas:

- "Descomposición de una colección XML utilizando la correlación de RDB_node" en la página 68
- "Descomposición de documentos XML en datos de DB2 UDB" en la página 98
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado 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.

Sintaxis:

```
dxxInsertXML(char(nombreColección) collectionName, /*entrada*/
             CLOB(1M)       xmlobj,               /* entrada */
             long           returnCode,           /* salida */
             varchar(1024)  returnMsg)            /* salida */
```

Parámetros:

Tabla 58. Parámetros de dxxInsertXML()

Parámetro	Descripción	Parámetro de E/S
collectionName	Es el nombre de una colección XML habilitada.	ENTRADA
xmlobj	Es un objeto de documento XML cuyo tipo es CLOB.	ENTRADA

Tabla 58. Parámetros de dxxInsertXML() (continuación)

Parámetro	Descripción	Parámetro de E/S
returnCode	Es el código de retorno que emite el procedimiento almacenado.	SALIDA
returnMsg	Es el texto del mensaje que se devuelve en caso de error.	SALIDA

Ejemplos:

En el ejemplo de fragmento siguiente, la llamada dxxInsertXML() descompone el documento XML de entrada `instalación_dxx/xml/order1.xml` e inserta los datos en las tablas de la colección SALES_ORDER de acuerdo con la correlación especificada en el archivo DAD con el que se ha habilitado. .

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

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
            char                 collection[64];  /* nombre de una col. XML */
            SQL TYPE is CLOB_FILE xmlDoc;         /* documento XML de entrada */
            long                 returnCode;      /* código de error */
            char                 returnMsg[1024]; /* texto del mens. de error */
            short                collection_ind;
            short                xmlDoc_ind;
            short                returnCode_ind;
            short                returnMsg_ind;
EXEC SQL END DECLARE SECTION;

/* inicializar variable del lenguaje principal e indicadores */
         strcpy(collection,"sales_ord")
         strcpy(xmlobj.name,"instalación_dxx/samples
         db2xml/xml/getstart.xml");
         xmlobj.name_length=strlen("instalación_dxx/samples
         db2xml/xml/getstart.xml");
         xmlobj.file_option=SQL_FILE_READ;
         returnCode = 0;
         returnMsg[0] = '\0';
         collection_ind = 0;
         xmlobj_ind = 0;
         returnCode_ind = -1;
         returnMsg_ind = -1;

              /* Llamar al procedimiento almacenado */
EXEC SQL CALL DB2XML.dxxInsertXML(:collection:collection_ind;
                             :xmlobj:xmlobj_ind,
               :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

Conceptos relacionados:

- "Procedimientos almacenados de descomposición de XML Extender - Visión general" en la página 220

Tareas relacionadas:

- "Descomposición de una colección XML utilizando la correlación de RDB_node" en la página 68
- "Descomposición de documentos XML en datos de DB2 UDB" en la página 98
- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- “Cómo leer los diagramas de sintaxis” en la página ix
- Apéndice C, “Límites de XML Extender”, en la página 321

Capítulo 11. Procedimientos almacenados y funciones de XML Extender para MQSeries

Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general

XML Extender proporciona dos métodos para almacenar y acceder a datos XML. Mediante el método de columna XML, puede guardar documentos XML en una tabla de DB2® y al mismo tiempo, consultar, actualizar y recuperar el contenido de los documentos. Las funciones definidas por el usuario MQ XML le permiten consultar documentos XML y luego publicar los resultados en una cola de mensajes. Además, puede utilizar el método de colección XML para guardar el contenido sin códigos de un documento XML en una o varias tablas o bien componer documentos XML a partir de varias tablas. Utilizando los procedimientos almacenados de MQ XML puede recuperar un documento XML de una cola de mensajes, descomponerlo en datos sin códigos y almacenar los datos en tablas de DB2 UDB. También puede componer un documento XML a partir de los datos de DB2 y enviar el documento a una cola de mensajes de MQSeries®.

MQSeries da soporte a tres modelos de mensajería para distribuir datos y documentos XML:

datagramas
: Los mensajes se envían a un solo destino sin que se espere ninguna respuesta.

publicar/suscribir
: Uno o más editores envían un mensaje a un servicio de publicación que distribuye el mensaje a los suscriptores interesados.

petición/respuesta
: Los mensajes se envían a un solo destino y el emisor espera recibir una respuesta.

MQSeries se puede utilizar de diversas maneras. Se intercambian datagramas simples para coordinar varias aplicaciones, para intercambiar información, solicitar servicios y proporcionar una notificación de los sucesos de interés. La publicación/suscripción se suele utilizar para divulgar información en tiempo real de forma puntual. El estilo de petición/respuesta se utiliza generalmente como un formulario simple de llamada de procedimiento remoto pseudosíncrona. Se pueden crear modelos más complejos combinando estos estilos básicos.

Las técnicas de mensajería fundamentales que se describen aquí se utilizan de formas muy diversas. Dado que MQSeries está disponible en una amplia gama de sistemas operativos, ofrece un mecanismo importante para enlazar aplicaciones dispares desde entornos similares o distintos.

Para utilizar las funciones y procedimientos almacenados MQXML, asegúrese de que tiene instalado el software siguiente:

- DB2 Universal Database™ Versión 7.2 o posterior

- DB2 MQSeries Functions Versión 7.2 (disponible como característica de instalación opcional de DB2 Universal Database Versión 7.2. La información de instalación se encuentra disponible en las Notas del release de DB2 Universal Database Versión 7.2.)
- MQSeries Publish/Subscribe o MQSeries Integrator cuando utilice funciones de edición.

Funciones MQSeries de XML Extender

Funciones de XML Extender MQSeries - Visión general

DB2® XML Extender incluye las siguientes funciones para su uso con MQSeries®:

- MQPublishXML
- MQReadXML
- MQReadAllXML
- MQReadXMLCLOB
- MQReadAllXMLCLOB
- MQReceiveXML
- MQReceiveAllXML
- MQRcvAllXMLCLOB
- MQReceiveXMLCLOB
- MQSENDXML
- MQSENDXMLFILE
- MQSendXMLFILECLOB

Conceptos relacionados:

- “Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general” en la página 225
- “Procedimientos almacenados de XML Extender MQSeries - Visión general” en la página 248

Información relacionada:

- “Función MQReadXML” en la página 229
- “Función MQReadAllXML” en la página 230
- “Función MQReceiveXML” en la página 235
- “Función MQReadXMLCLOB” en la página 232
- “Función MQReadAllXMLCLOB” en la página 233
- “Función MQRcvXMLCLOB” en la página 242
- “Función MQReceiveXMLCLOB” en la página 241
- “Función MQPublishXML” en la página 227
- “Función MQReceiveAllXML” en la página 237
- “Función MQRcvAllXMLCLOB” en la página 239
- “Función MQSendXMLFILECLOB” en la página 246
- “Función MQSENDXML” en la página 243
- “Función MQSENDXMLFILE” en la página 245

Función MQPublishXML

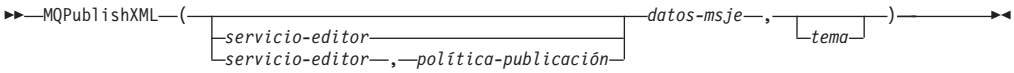
Propósito:

La función MQPublishXML publica los datos XMLVARCHAR y XMLCLOB para MQSeries. Esta función requiere la instalación de MQSeries Publish/Subscribe o bien de MQSeries Integrator. Para obtener más información, visite el siguiente sitio Web:

`http://www.software.ibm.com/MQSeries`

La función MQPublishXML publica los datos XML incluidos en *datos-msje* para el editor de MQSeries especificado por *servicio-editor* utilizando la calidad de la política de publicación *política-publicación*. El tema del mensaje se especifica opcionalmente mediante *tema*. Puede especificar un identificador opcional de correlación de mensajes definido por el usuario mediante *correl-id*. La función devuelve un valor de 1, en caso de ejecutarse satisfactoriamente.

Sintaxis:



Parámetros:

Tabla 59. Parámetros de MQPublishXML

Parámetro	Tipo de datos	Descripción
servicio-editor	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Si se especifica, *servicio-editor* hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Si no se especifica *servicio-editor*, se utilizará DB2.DEFAULT.PUBLISHER. El tamaño máximo de servicio-editor es de 48 bytes.

Tabla 59. Parámetros de MQPublishXML (continuación)

Parámetro	Tipo de datos	Descripción
política-publicación	VARCHAR(48)	Serie que contiene la *política de publicación* AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica, *política-publicación* hace referencia a una política que está definida en el archivo de depósito AMT.XML. La política de publicación también define un grupo de opciones de calidad de publicación que se deben aplicar a las opciones de la operación de mensajería. Estas opciones incluyen la prioridad de los mensajes y la permanencia de éstos; si no se especifica *política-servicio*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de política-servicio es de 48 bytes. Para obtener más información, consulte MQSeries Application Messaging Interface.
datos-msje	XMLVARCHAR o XMLCLOB	Expresión XMLVARCHAR o XMLCLOB que contiene los datos que se deben enviar a través de MQSeries.
tema	VARCHAR(40)	Serie que contiene el tema bajo el cual se debe publicar el mensaje. Si no se especifica ningún tema, no se asociará ninguno al mensaje. El tamaño máximo del tema es de 40 bytes. Se pueden listar varios temas dentro de una serie de temas separando cada tema por el símbolo ″:″.

Códigos de retorno:
Si se ejecutan satisfactoriamente, las funciones de MQPublishXML devuelven un valor de 1. Se devuelve un valor de 0 si la función no se ejecuta satisfactoriamente.
Conceptos relacionados:

- “Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general” en la página 225

Información relacionada:

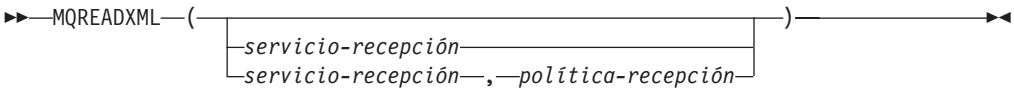
- “Cómo leer los diagramas de sintaxis” en la página ix

Función MQReadXML

Propósito:

La función MQREADXML devuelve datos XMLVARCHAR desde la ubicación MQSeries que está especificada por *servicio-recepción*. Utiliza la calidad de *política de recepción*. La función MQREADXML no elimina mensajes de la cola asociados a *servicio-recepción*

Sintaxis:



Parámetros:

Tabla 60. Parámetros de MQReadXML

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico desde el cual se va a recibir el mensaje. Si se especifica *servicio-recepción*, este parámetro hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Si no se especifica *servicio-recepción*, se utiliza DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes
política-recepción	VARCHAR(48)	Serie que contiene la política de servicios AMI de MQSeries que se utiliza en el manejo de un mensaje. Cuando se especifica *política-recepción*, hace referencia a una política definida en el archivo de depósito AMT.XML. Una política de recepción define un grupo de opciones de calidad de recepción que se aplican a la operación de mensajería. Estas opciones incluyen la prioridad del mensaje y la permanencia del mensaje. Si no se especifica *política-recepción*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-recepción* es de 48 bytes.

Resultados:

Cuando un mensaje de la cola se ha leído satisfactoriamente, MQREADXML devuelve un db2xml.xmlvarchar. Se devuelve un valor NULL si no hay ningún mensaje disponible.

Ejemplos:

Ejemplo 1: Este ejemplo lee el mensaje a la cabeza de la cola que está especificada por el servicio por omisión *DB2.DEFAULT.SERVICE*. Utiliza la política por omisión *DB2.DEFAULT.POLICY* para leer el mensaje.

```
values DB2XML.MQREADXML()
```

Este ejemplo devuelve el contenido del mensaje como un XMLVARCHAR. Si no hay ningún mensaje disponible, se devuelve un valor NULL.

Ejemplo 2: Este ejemplo lee el mensaje situado a la cabeza de la cola especificada por el servicio *MYSERVICE* utilizando la política por omisión *DB2.DEFAULT.POLICY*.

```
values DB2XML.MQREADXML('MYSERVICE')
```

En el ejemplo, el contenido del mensaje se devuelve como XMLVARCHAR. Si no existe ningún mensaje disponible, se devuelve un valor NULL.

Ejemplo 3: Este ejemplo lee el mensaje situado a la cabeza de la cola especificada por el servicio *MYSERVICE* utilizando la política *MYPOLICY*.

```
values DB2XML.MQREADXML('MYSERVICE','MYPOLICY')
```

Se devuelve el contenido del mensaje como XMLVARCHAR, si es satisfactorio. Si no hay ningún mensaje disponible, se devuelve un valor NULL.

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

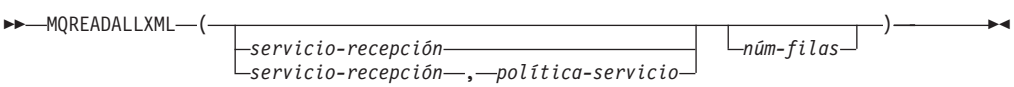
- "Cómo leer los diagramas de sintaxis" en la página ix

Función MQReadAllXML

Propósito:

La función MQReadAllXML devuelve una tabla que contiene los mensajes y metadatos de mensajes de la ubicación de MQSeries especificada por *servicio-recepción* utilizando la calidad de *política-servicio*. Al ejecutar esta operación, no se eliminan los mensajes de la cola asociados a *servicio-recepción*. Si se especifica *núm-filas*, se devolverá un máximo de número de filas de mensajes. Si no se especifica núm-filas, se devolverán todos los mensajes disponibles.

Sintaxis:



Parámetros:

Tabla 61. Parámetros de MQReadAllXML

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico desde el cual se va a leer el mensaje. Si se especifica, *servicio-recepción* debe hacer referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Sin embargo, si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.
política-servicio	VARCHAR(48)	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar este mensaje. Cuando se especifica *política-servicio*, hace referencia a una política definida en el archivo de depósito AMT.XML. El tamaño máximo de servicio-recepción es de 48 bytes. Para obtener más información, consulte el manual MQSeries Application Messaging Interface.
núm-filas	INTEGER	Entero positivo que contiene el número máximo de mensajes que la función va a devolver.

Resultados:
La función MQReadAllXML devuelve una tabla que contiene mensajes y metadatos de mensajes, tal como se describe a continuación.

Tabla 62. Tabla de grupos de resultados

Nombre de columna	Tipo de datos	Descripción
MSG	XMLVARCHAR	Contenido del mensaje MQSeries. La longitud máxima es de 4 Kbytes.
CORRELID	VARCHAR(24)	ID de correlación que se puede utilizar para relacionar los mensajes.
TOPIC	VARCHAR(40)	El tema con el que se ha publicado el mensaje, si está disponible.
QNAME	VARCHAR(48)	Nombre de la cola en la que se ha recibido el mensaje

Tabla 62. Tabla de grupos de resultados (continuación)

Nombre de columna	Tipo de datos	Descripción
MSGID	VARCHAR(24)	Identificador exclusivo asignado por MQSeries para un mensaje.
MSGFORMAT	VARCHAR(8)	Formato del mensaje definido por MQSeries. Las series típicas tienen un formato de MQSTR.

Ejemplos:
Ejemplo 1: Todos los mensajes de la cola que se han especificado mediante el servicio por omisión DB2.DEFAULT.SERVICE se leen utilizando la política por omisión DB2.DEFAULT.POLICY. Los mensajes y todos los metadatos se devuelven en un formato de tabla.

```
select * from table (DB2XML.MQREADALLXML()) t
```

Ejemplo 2: Todos los mensajes que se han especificado mediante el servicio MYSERVICE utilizando la política por omisión DB2.DEFAULT.POLICY. Sólo se devuelven las columnas *msg* y *correlid*. La cola de mensajes tiene un formato de tabla, donde puede seleccionar los campos que desee.

```
select t.MSG, t.CORRELID from table (DB2XML.MQREADALLXML('MYSERVICE')) t
```

Ejemplo 3: La cola que se ha especificado mediante el servicio por omisión DB2.DEFAULT.SERVICE se lee utilizando la política por omisión DB2.DEFAULT.POLICY. Sólo se devuelven los mensajes con un *CORRELID* de '1234'. Se leen y devuelven, como máximo, 10 mensajes. Se devuelven todas las columnas.

```
select * from table (DB2XML.MQREADALLXML()) t where t.CORRELID = '1234'
```

Ejemplo 4: Los mensajes que se han especificado mediante el servicio por omisión DB2.DEFAULT.SERVICE se leen utilizando la política por omisión DB2.DEFAULT.POLICY. Se devuelven todas las columnas.

```
select * from table (DB2XML.MQREADALLXML(10)) t
```

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

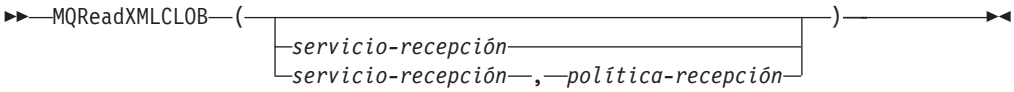
- "Cómo leer los diagramas de sintaxis" en la página ix

Función MQReadXMLCLOB

Propósito:

La función MQREADXMLCLOB devuelve datos XMLCLOB de la ubicación de MQSeries especificada mediante *servicio-recepción* utilizando la calidad de la política de servicios *política-recepción*. Al ejecutar esta operación, no se elimina el mensaje de la cola asociada a *servicio-recepción*. Se devolverá el mensaje situado en la cabeza de la cola. El valor de retorno es un valor XMLCLOB que contiene los mensajes. Si no hay ningún mensaje disponible para devolver, se devolverá un valor NULL.

Sintaxis:



Parámetros:

Tabla 63. Parámetros de MQReadXMLCLOB

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico desde el cual se va a recibir el mensaje. Si se especifica *servicio-recepción*, este parámetro hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes
política-recepción	VARCHAR(48)	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar este mensaje. Cuando se especifica *política-recepción*, hace referencia a una política definida en el archivo de depósito AMT.XML. Una política de servicios define un grupo de opciones de calidad de servicio que se aplican a la operación de mensajería. Estas opciones incluyen la prioridad del mensaje y la permanencia del mensaje. Si no se especifica *política-recepción*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de servicio-recepción es de 48 bytes.

Resultados:
Cuando un mensaje de la cola se ha leído satisfactoriamente, MQREADXMLCLOB devuelve un valor db2xml.xmlclob. Se devuelve un valor NULL si no hay ningún mensaje disponible.

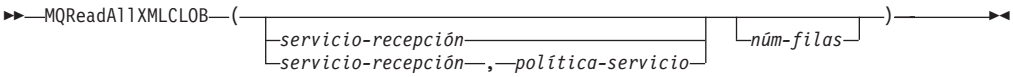
Función MQReadAllXMLCLOB

Propósito:

La función MQReadAllXMLCLOB devuelve una tabla que contiene los mensajes y metadatos de mensajes de la ubicación de MQSeries especificada por *servicio-recepción* utilizando la calidad de la política de servicios servicio-recepción.

Al ejecutar esta operación, no se eliminan los mensajes de la cola asociados a servicio-recepción. Si se especifica *núm-filas*, se devolverá un máximo de número de filas de mensajes. Si no se especifica núm-filas, se devolverán todos los mensajes disponibles.

Sintaxis:



Parámetros:

Tabla 64. Parámetros de MQReadAllXMLCLOB

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico desde el cual se debe leer el mensaje. Si se especifica *servicio-recepción*, este parámetro debe hacer referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Sin embargo, si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.
política-servicio	VARCHAR(48)	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar este mensaje. Cuando se especifica *política-servicio*, hace referencia a una política definida en el archivo de depósito AMT.XML. El tamaño máximo de *política-servicio* es de 48 bytes.
núm-filas	INTEGER	Entero positivo que contiene el número máximo de mensajes que la función va a devolver.

Resultados:

La función MQReadAllXMLCLOB devuelve una tabla que contiene mensajes y metadatos de mensajes, tal como se describe a continuación.

Tabla 65. Tabla de grupos de resultados de MQReadAllXMLCLOB

Nombre de columna	Tipo de datos	Descripción
MSG	XMLCLOB	Contenido del mensaje MQSeries, hasta 1 MB de longitud.

Tabla 65. Tabla de grupos de resultados de MQReadAllXMLCLOB (continuación)

Nombre de columna	Tipo de datos	Descripción
CORRELID	VARCHAR(24)	ID de correlación que se puede utilizar para relacionar los mensajes.
TOPIC	VARCHAR(40)	El tema con el que se ha publicado el mensaje, si está disponible.
QNAME	VARCHAR(48)	Nombre de la cola en la que se ha recibido el mensaje
MSGID	VARCHAR(24)	Identificador exclusivo asignado por MQSeries para este mensaje.
MSGFORMAT	VARCHAR(8)	Formato del mensaje definido por MQSeries. Las series típicas tienen un formato de MQSTR.

Ejemplo 1: Todos los mensajes de la cola que se han especificado mediante el servicio por omisión DB2.DEFAULT.SERVICE se leen utilizando la política por omisión DB2.DEFAULT.POLICY. Los mensajes y todos los metadatos se devuelven en un formato de tabla.

```
select * from table (DB2XML.MQREADALLXMLCLOB()) t
```

Ejemplo 2: Los mensajes de la cabeza de la cola se especifican mediante el servicio MYSERVICE utilizando la política por omisión DB2.DEFAULT.POLICY. Sólo se devuelven las columnas *msg* y *correlid*.

```
select t.MSG, t.CORRELID
from table (DB2XML.MQREADALLXMLCLOB('MYSERVICE')) t
```

Ejemplo 3: La cabeza de la cola que está especificada por el servicio por omisión DB2.DEFAULT.SERVICE se lee utilizando la política por omisión DB2.DEFAULT.POLICY. Sólo se devuelven los mensajes con un *CORRELID* de '1234'. Se devuelven todas las columnas.

```
select *
from table (DB2XML.MQREADALLXMLCLOB()) t where t.CORRELID = '1234'
```

Ejemplo 4: Los 10 primeros mensajes de la cabeza de la cola que están especificados por el servicio por omisión DB2.DEFAULT.SERVICE se leen utilizando la política por omisión DB2.DEFAULT.POLICY. Se devuelven todas las columnas.

```
select * from table (DB2XML.MQREADALLXMLCLOB(10)) t
```

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

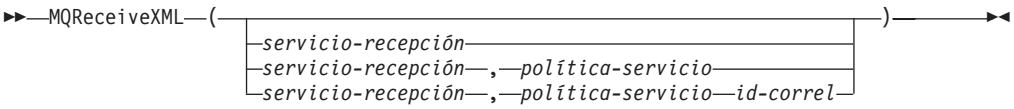
- "Cómo leer los diagramas de sintaxis" en la página ix

Función MQReceiveXML

Propósito:

La función MQReceiveXML elimina un mensaje asociado a *servicio-recepción* de la cola. La función devuelve datos XMLVARCHAR de la ubicación de MQSeries especificada por la función *servicio-recepción* que utiliza la calidad de *servicio-recepción*.

Sintaxis:



Parámetros:

Tabla 66. Parámetros de MQReceiveXML

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico a partir del cual se va a recibir el mensaje. Si se especifica *servicio-recepción*, este parámetro hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.
política-servicio	VARCHAR(48)	Serie que contiene la política de servicios de AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica *política-servicio*, este parámetro debe hacer referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *política-servicio*, se utilizará el valor por omisión DDB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional que se debe asociar a este mensaje. El *id-correl* se suele especificar en los casos de petición/respuesta para asociar peticiones a respuestas. Si no aparece visualizado, no se especificará ningún ID de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

Resultados:

Las funciones de MQReceiveXML devuelven un db2xml.XMLVARCHAR si los mensajes se reciben de la cola satisfactoriamente. El tamaño máximo del mensaje es de 4000 bytes. Se devuelve un valor NULL si no hay ningún mensaje disponible. Si se especifica *id-correl*, se devolverá el primer mensaje con un identificador de correlación que coincida. Si no se especifica *id-correl*, se devolverá el mensaje situado en la cabeza de la cola. El mensaje se elimina de la cola.

Ejemplos:

Ejemplo 1: Este ejemplo recibe el mensaje que se encuentra a la cabeza de la cola y se especifica mediante el servicio por omisión DB2.DEFAULT.SERVICE utilizando la política por omisión DB2.DEFAULT.POLICY.

```
values db2xml.MQRECEIVEXML()
```

Si este ejemplo es satisfactorio, devuelve el contenido de un mensaje como XMLVARCHAR. Si no hay ningún mensaje disponible, se devuelve un valor NULL.

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

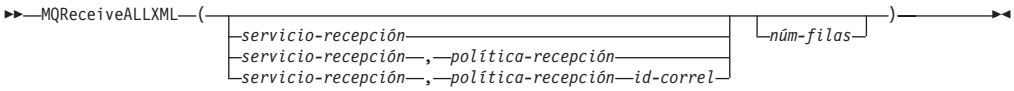
- "Cómo leer los diagramas de sintaxis" en la página ix

Función MQReceiveAllXML

Propósito:

La función MQReceiveAllXML elimina los mensajes asociados a *servicio-recepción* de la cola. Si se especifica id-correl, sólo se devolverán los mensajes con un identificador de correlación que coincida. Si no se especifica *id-correl*, se devolverá el mensaje situado en la cabeza de la cola. Si se especifica *núm-filas*, se devolverá un máximo de mensajes (*núm-filas*). Si no se especifica, se devolverán todos los mensajes disponibles.

Sintaxis:



Parámetros:

Tabla 67. Parámetros de MQReceiveAllXML

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Si se especifica, *servicio-envío* hace referencia a un punto de servicio definido en el archivo de depósito ATM.XML. Si no se especifica *servicio-envío*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-envío* es de 48 bytes.

Tabla 67. Parámetros de MQReceiveAllXML (continuación)

Parámetro	Tipo de datos	Descripción
política-recepción	VARCHAR(48)	Serie que contiene la política de servicios de AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica, *política-recepción* debe hacer referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *política-recepción*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-recepción* es de 48 bytes.
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional que se debe asociar a este mensaje. *id-correl* se suele especificar en los casos de petición/respuesta para asociar peticiones a respuestas. Si no aparece visualizado, no se especificará ningún id de correlación. El tamaño máximo de *id-correl* es de 24 bytes.
núm-filas	INTEGER	Entero positivo que contiene el número máximo de mensajes devueltos por la función.

Resultados:
Cuando se recibe satisfactoriamente una tabla de mensajes desde la cola, MQRECEIVEXML devuelve un valor db2xml.xmlvarchar. Se devuelve un valor NULL cuando no hay ningún mensaje disponible. Los mensajes se devuelven como tabla de mensajes con metadatos.

Nombre de columna	Tipo de datos	Descripción
MSG	XMLVARCHAR	Contenido del mensaje MQSeries.
CORRELID	VARCHAR(24)	ID de correlación que se puede utilizar para relacionar los mensajes.
TOPIC	VARCHAR(40)	El tema con el que se ha publicado el mensaje, si está disponible.
QNAME	VARCHAR(48)	Nombre de la cola en la que se ha recibido el mensaje.
MSGID	CHAR(24)	Identificador exclusivo asignado por MQSeries para este mensaje.

Nombre de columna	Tipo de datos	Descripción
MSGFORMAT	VARCHAR(8)	Formato del mensaje definido por MQSeries. Las series típicas tienen un formato de MQSTR.

Ejemplos:
Ejemplo 1: Todos los mensajes recibidos de la cola se especifican en el servicio por omisión (DB2.DEFAULT.SERVICE) utilizando la política por omisión (DB2.DEFAULT.POLICY). Los mensajes y todos los metadatos se devuelven como una tabla.

```
select * from table (MQRECEIVEALLXML()) t
```

Ejemplo 2: Todos los mensajes se reciben desde la cabeza de la cola y se especifican en el servicio MYSERVICE utilizando la política por omisión (DB2.DEFAULT.POLICY). Sólo se devuelven las columnas MSG y CORRELID. Los mensajes están en formato de tabla, donde puede seleccionar los campos que desee.

```
select t.MSG, t.CORRELID from table (MQRECEIVEALLXML('MYSERVICE')) t
```

Ejemplo 3: Todos los mensajes recibidos desde la cabeza de la cola se especifican en el servicio MYSERVICE utilizando la política MYPOLICY que corresponde con el id '1234'. Sólo se devuelven las columnas MSG y CORRELID.

```
select t.MSG, t.CORRELID from table
(MQRECEIVEALLXML('MYSERVICE','MYPOLICY','1234')) t
```

Ejemplo 4: Los 10 primeros mensajes se reciben desde la cabeza de la cola y se especifican en el servicio por omisión (DB2.DEFAULT.SERVICE) utilizando la política por omisión (DB2.DEFAULT.POLICY). Se devuelven todas las columnas.

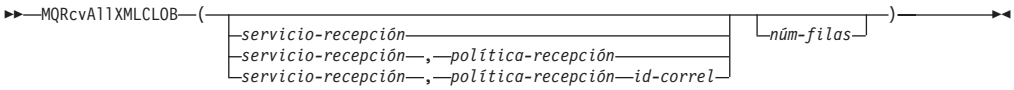
```
 select * from table (MQRECEIVEALLXML(10)) t
```

Función MQRcvAllXMLCLOB

Propósito:

MQRcvAllXMLCLOB elimina los mensajes de la cola asociada a *servicio-recepción*. Si se especifica id-correl, sólo se devolverán aquellos mensajes con un identificador de correlación que coincida. Si no se especifica *id-correl*, se devolverán todos los mensajes. Si se especifica *núm-filas*, se devolverá un máximo de *núm-filas* de mensajes como XMLCLOB. Si no se especifica, se devolverán todos los mensajes disponibles.

Sintaxis:



Parámetros:

Tabla 68. Parámetros de MQRcvAllXMLCLOB

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico desde el cual se va a recibir el mensaje. Si se especifica, *servicio-recepción* hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de servicio-recepción es de 48 bytes.
política-recepción	VARCHAR(48)	Serie que contiene la política de servicios de AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica, *política-recepción* debe hacer referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *política-recepción*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-recepción* es de 48 bytes.
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional que se debe asociar a este mensaje. *id-correl* se suele especificar en los casos de petición/respuesta para asociar peticiones a respuestas. Si no aparece visualizado, no se especificará ningún id de correlación. El tamaño máximo de *id-correl* es de 24 bytes.
núm-filas	INTEGER	Entero positivo que contiene el número máximo de mensajes devueltos por la función.

Resultados:
Cuando se recibe satisfactoriamente un mensaje desde la cola, MQRcvAllXMLCLOB devuelve un valor XMLCLOB. Se devuelve un valor NULL cuando no hay ningún mensaje disponible. Los mensajes se devuelven en una tabla tal y como se describe a continuación.

Tabla 69. Tabla de grupo de resultados de MQRcvAllXML

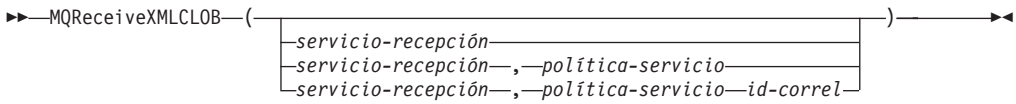
Nombre de columna	Tipo de datos	Descripción
MSG	XMLCLOB	Contenido del mensaje MQSeries.
CORRELID	VARCHAR(24)	ID de correlación que se puede utilizar para relacionar los mensajes.
TOPIC	VARCHAR(40)	El tema con el que se ha publicado el mensaje, si existe alguno.
QNAME	VARCHAR(48)	Nombre de la cola en la que se ha recibido el mensaje.
MSGID	CHAR(24)	Identificador exclusivo asignado por MQSeries para este mensaje.
MSGFORMAT	VARCHAR(8)	Formato del mensaje definido por MQSeries. Las series típicas tienen un formato de MQSTR.

Función MQReceiveXMLCLOB

Propósito:

La función MQReceiveXMLCLOB elimina los mensajes asociados a *servicio-recepción* de la cola. La función devuelve datos XMLVARCHAR de la ubicación de MQSeries especificada por la función *política-servicio* que utiliza la calidad de *servicio-recepción*.

Sintaxis:



Parámetros:

Tabla 70. Parámetros de MQReceiveXMLCLOB

Parámetro	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico desde el cual se debe recibir el mensaje. Cuando se especifica *servicio-recepción*, este parámetro hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Sin embargo, si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.

Tabla 70. Parámetros de MQReceiveXMLCLOB (continuación)

Parámetro	Tipo de datos	Descripción
política-servicio	VARCHAR(48)	Serie que contiene la política de servicios AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica, servicio-recepción debe hacer referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *política-servicio*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional que se debe asociar a este mensaje. El *id-correl* se suele especificar en los casos de petición/respuesta para asociar peticiones a respuestas. Si no aparece visualizado, no se especificará ningún ID de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

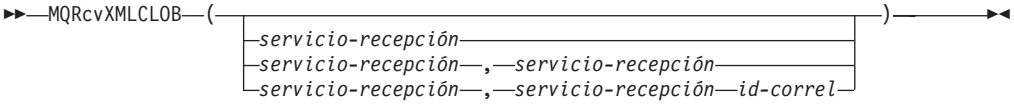
Resultados:
Las funciones de MQReceiveXMLCLOB devuelven un valor db2xml.XMLCLOB, si se reciben mensajes de la cola satisfactoriamente. Se devuelve un valor NULL si no hay ningún mensaje disponible. Si se especifica *id-correl*, se devolverá el primer mensaje con un identificador de correlación que coincida. No obstante, si no se especifica *id-correl*, se devolverá el mensaje que se encuentra a la cabeza de la cola.

Función MQRcvXMLCLOB

Propósito:

MQRcvXMLCLOB elimina los mensajes asociados a *servicio-recepción* de la cola. La función devuelve datos XMLVARCHAR de la ubicación de MQSeries especificada por la función *servicio-recepción* que utiliza la calidad de *servicio-recepción*.

Sintaxis:



Parámetros:

Tabla 71. Parámetros de MQRcvXMLCLOB

Tipo de datos	Tipo de datos	Descripción
servicio-recepción	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico desde el cual se va a recibir el mensaje. Cuando se especifica *servicio-recepción*, este parámetro hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. No obstante, si no se especifica *servicio-recepción*, se utilizará el valor DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.
servicio-recepción	VARCHAR(48)	Serie que contiene la política de servicios AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica, servicio-recepción debe hacer referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *servicio-recepción*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *servicio-recepción* es de 48 bytes.
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional que se debe asociar a este mensaje. El *id-correl* se suele especificar en los casos de petición/respuesta para asociar peticiones a respuestas. Si no aparece visualizado, no se especificará ningún ID de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

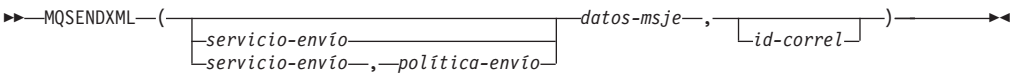
Resultados:
Las funciones de MQRcvMLCLOB devuelven un valor db2xml.XMLCLOB si se reciben satisfactoriamente mensajes de la cola. El tamaño máximo del mensaje es de 1 M. Se devuelve un valor NULL si no hay ningún mensaje disponible. Si se especifica *id-correl*, se devolverá el primer mensaje con un identificador de correlación que coincida. No obstante, si no se especifica *id-correl*, se devolverá el mensaje que se encuentra a la cabeza de la cola.

Función MQSENDXML

Propósito:

La función MQSENDXML envía los datos incluidos en *datos-msje* a la ubicación de MQSeries especificada por servicio-envío utilizando la política-envío. Correl-id puede también especificar un identificador opcional de correlación de mensajes definido por el usuario. La función devuelve un valor de 1, en caso de que se ejecute satisfactoriamente.

Sintaxis:



Parámetros:

Tabla 72. Parámetros de MQSendXML

Parámetro	Tipo de datos	Descripción
datos-msje	XMLVARCHAR o XMLCLOB	Expresión que contiene los datos que se deben enviar a través de MQSeries.
servicio-envío	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *servicio-envío* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *servicio-envío*. El tamaño máximo de *servicio-envío* es de 48 bytes.
política-envío	VARCHAR(48)	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar el mensaje. Cuando se especifica, *política-envío* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *política-envío*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de política-envío es de 48 bytes.
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional asociado a este mensaje. El id-correl se suele especificar en los casos de petición/respuesta para asociar peticiones a respuestas. Si no se especifica, no se visualizará ningún id de correlación. El tamaño máximo de id-correl es de 24 bytes.

Resultados:

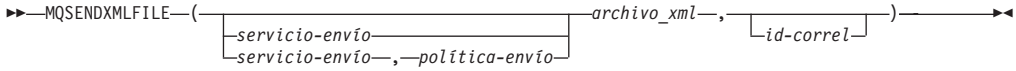
Un mensaje satisfactorio genera un valor de 1. Se enviará un mensaje que contenga los *datos-msje* a la ubicación especificada por servicio-envío utilizando la política definida por *política-envío*.

Función MQSENDXMLFILE

Propósito:

La función MQSENDXMLFILE envía los datos incluidos en *archivo_xml* a la ubicación de MQSeries especificada por servicio-envío utilizando la calidad de la política de servicios. Correl-id puede especificar un identificador opcional de correlación de mensajes definido por el usuario. La función devuelve un valor de '1', en caso de que se ejecute satisfactoriamente.

Sintaxis:



Parámetros:

Tabla 73. Parámetros de MQSENDXMLFILE

Parámetro	Tipo de datos	Descripción
archivo_xml	XMLCLOB	Un nombre de archivo XML con un tamaño máximo de 80 bytes. Este archivo contiene los datos que se deben enviar a través de MQSeries.
servicio-envío	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Si se especifica, *servicio-envío* hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Si no se especifica *servicio-envío*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-envío* es de 48 bytes.
política-envío	VARCHAR(48)	Serie que contiene el servicio AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica, *política-envío* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *política-envío*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-envío* es de 48 bytes

Tabla 73. Parámetros de MQSENDXMLFILE (continuación)

Parámetro	Tipo de datos	Descripción
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional que se debe asociar a este mensaje. *id-correl* se suele especificar en los casos de petición/respuesta especificados para asociar peticiones a respuestas. Si no se especifica, no se listará ningún id de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

Resultados:
Si la función se ejecuta satisfactoriamente, devuelve un valor de '1'. El efecto secundario de ejecutar satisfactoriamente esta función es que se enviará un mensaje que contenga los *datos-msje* a la ubicación especificada por *servicio-envío* utilizando la política definida por *política-envío*.
Ejemplos:
Ejemplo 1: Los documentos XML incluidos en el archivo "c:\xml\test1.xml" se envían al servicio por omisión (DB2.DEFAULT.SERVICE) utilizando la política por omisión (DB2.DEFAULT.POLICY) sin ningún identificador de correlación.

```
Valores MQSENDXMLFILE('c:\xml\test1.xml');
```

Este ejemplo devuelve el valor '1', en caso de que sea satisfactorio

Ejemplo 2: Los documentos XML incluidos en el archivo `c:\xml\test2.xml` se envían al servicio MYSERVICE utilizando la política MYPOLICY sin ningún identificador de correlación.

```
Valores MQSENDXMLFILE('MYSERVICE', 'MYPOLICY', 'c:\xml\test2.xml');
```

Este ejemplo devuelve el valor '1', en caso de que sea satisfactorio

Ejemplo 3: Los documentos XML incluidos en el archivo "c:\xml\test3.xml" se envían al servicio MYSERVICE utilizando la política MYPOLICY con identificador de correlación "Test3".

```
Values MQSENDXML('MYSERVICE','MYPOLICY', 'c:\xml\test3.xml', 'Test3');
```

Este ejemplo devuelve el valor '1', en caso de que sea satisfactorio

Ejemplo 4: Los documentos XML incluidos en el archivo "c:\xml\test4.xml" se envían al servicio MYSERVICE utilizando la política por omisión (DB2.DEFAULT.POLICY) y sin ningún identificador de correlación.

```
Values MQSENDXMLFILE('MYSERVICE', 'c:\xml\test4.xml');
```

Este ejemplo devuelve el valor '1', en caso de que sea satisfactorio

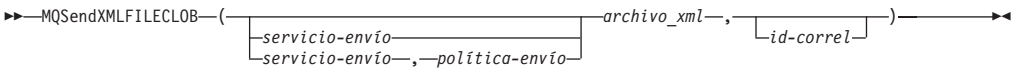
Función MQSendXMLFILECLOB

Propósito:

La función MQSendXMLFILECLOB envía los datos incluidos en *archivo_xml* a la ubicación de MQSeries especificada por servicio-envío utilizando la calidad de

política-envío. El tipo de datos que se envía es XMLCLOB. Correl-id puede especificar un identificador opcional de correlación de mensajes definido por el usuario. La función devuelve un valor de 1, en caso de que se ejecute satisfactoriamente.

Sintaxis:



Parámetros:

Tabla 74. Parámetros de MQSENDXMLFILE

Parámetro	Tipo de datos	Descripción
archivo_xml	XMLCLOB	Un nombre de archivo XML con un tamaño máximo de 80 bytes. El archivo contiene los datos que se deben enviar a través de MQSeries.
servicio-envío	VARCHAR(48)	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Si se especifica, el servicio-envío hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Si no se especifica *servicio-envío*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de servicio-envío es de 48 bytes
política-envío	VARCHAR(48)	Serie que contiene el servicio AMI de MQSeries que se debe utilizar para manejar este mensaje. Si se especifica, *política-envío* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *política-envío*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-envío* es de 48 bytes
id-correl	VARCHAR(24)	Serie que contiene un identificador de correlación opcional que se debe asociar a este mensaje. *id-correl* se suele especificar en los casos de petición/respuesta especificados para asociar peticiones a respuestas. Si no se especifica, no se listará ningún id de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

Resultados:

Si la función se ejecuta satisfactoriamente, devuelve un valor de '1'. El efecto secundario de ejecutar satisfactoriamente esta función es que se enviará un mensaje que contenga los *datos-msje* a la ubicación especificada por *servicio-envío* utilizando la política definida por *política-envío*.

Procedimientos almacenados MQSeries de XML Extender

Procedimientos almacenados de XML Extender MQSeries - Visión general

Procedimientos almacenados de composición

Utilice los procedimientos almacenados de composición, dxxmqGen(), dxxmqGenCLOB(), dxxmqRetrieve() y dxxmqRetrieveCLOB(), para generar documentos XML a partir de los datos de las tablas de bases de datos existentes y para enviar los documentos XML generados a una cola de mensajes. Los procedimientos almacenados dxxmqGen() y dxxmqGenCLOB() utilizan un archivo DAD como entrada. No necesitan tener colecciones XML habilitadas. Los procedimientos almacenados dxxmqRetrieve y dxxmqRetrieveCLOB utilizan nombres de colección como entrada de datos.

Procedimientos almacenados de descomposición

Los procedimientos almacenados de descomposición dxxmqInsert(), dxxmqInsertAll(), dxxInsertCLOB(), dxxmqShred(), dxxmqShredCLOB y dxxmqShredAll() se utilizan para fragmentar o desglosar documentos XML de entrada de una cola de mensajes y almacenar los datos en tablas de bases de datos nuevas o existentes.

Los procedimientos almacenados dxxmqInsert(), dxxmqInsertAll(), dxxmqInsertAllCLOB() y dxxInsertCLOB() utilizan como entrada un nombre de colección XML habilitado.

Los procedimientos almacenados dxxmqShred(), dxxmqShredAll() dxxmqShredCLOB y dxxmqShredAllCLOB utilizan un archivo DAD como entrada de datos. No necesitan tener una colección XML habilitada.

En la tabla siguiente se resumen los diferentes procedimientos almacenados y se explican sus funciones.

Tabla 75. Los procedimientos almacenados XML de MQSeries®

Función	Propósito
dxxmqGen	Invocar el procedimiento almacenado dxxmqGen para componer documentos XML utilizando un archivo DAD como parámetro de entrada. El tipo de documento resultante es XMLVARCHAR(4000).
dxxmqGenCLOB	Crea un documento XML a partir de los datos que están almacenados en las tablas de la colección XML especificadas en el archivo DAD y envía el documento XML a una cola de mensajes MQ. El tipo de documento resultante es XMLCLOB(1M).

Tabla 75. Los procedimientos almacenados XML de MQSeries® (continuación)

Función	Propósito
dxxmqRetrieve	Invocar el procedimiento almacenado dxxmqRetrieve para componer documentos XML, utilizando como parámetro de entrada un nombre de colección. El tipo de documento resultante es XMLVARCHAR(4000).
dxxmqRetrieveCLOB	Invocar el procedimiento almacenado dxxmqRetrieve para componer documentos XML, utilizando como parámetro de entrada un nombre de colección. El tipo de documento resultante es XMLCLOB(1M).
dxxmqShred	Invocar el procedimiento almacenado dxxmqShred para descomponer un documento XML utilizando como parámetro de entrada un archivo DAD. El tipo de documento resultante es XMLVARCHAR(4000).
dxxmqShredAll	Invocar el procedimiento almacenado dxxmqShredAll para descomponer varios documentos XML utilizando como parámetro de entrada un archivo DAD. El tipo de documento resultante es XMLVARCHAR(4000).
dxxmqShredCLOB	Descompone un documento XML de entrada desde una cola de mensajes, basándose en una correlación de archivo DAD, y almacena el contenido de los elementos y atributos XML en las tablas de DB2® UDB especificadas. El tipo de documento resultante es XMLCLOB(1M).
dxxmqShredAllCLOB	Descompone un documento XML de entrada de una cola de mensajes, basándose en una correlación de archivo DAD, y almacena el contenido de los elementos y atributos XML en tablas de DB2 UDB especificadas. El tipo de documento resultante es XMLCLOB(1M).
dxxmqInsert	Invocar el procedimiento almacenado para descomponer un documento XML utilizando un nombre de colección como parámetro de entrada. El tipo de documento resultante es XMLVARCHAR(4000).
dxxmqInsertAll	Invocar el procedimiento almacenado dxxmqInsertAll para descomponer varios documentos XML utilizando como parámetro de entrada un nombre de colección. El tipo de documento resultante es XMLVARCHAR(4000).
dxxmqInsertCLOB	Fragmenta o desglosa un documento XML de entrada de una cola de mensajes y almacena los datos en tablas de bases de datos nuevas o existentes. El tipo de documento resultante es XMLCLOB(1M).

Tabla 75. Los procedimientos almacenados XML de MQSeries® (continuación)

Función	Propósito
dxxmqInsertAllCLOB	Fragmenta o desglosa todos los documentos XML de entrada de una cola de mensajes y almacena los datos en tablas de bases de datos nuevas o existentes. El procedimiento almacenado dxxmqInsertAllCLOB utiliza un nombre de colección, en lugar de un nombre de archivo DAD, para determinar cómo almacenar los datos. El tipo de documento resultante es XMLCLOB(1M).

Información relacionada:

- "Procedimiento almacenado dxxmqGenCLOB" en la página 253
- "Procedimiento almacenado dxxmqRetrieve" en la página 255
- "Procedimiento almacenado dxxmqRetrieveCLOB" en la página 258
- "Procedimiento almacenado dxxmqShred" en la página 260
- "Procedimiento almacenado dxxmqShredAll" en la página 262
- "Procedimiento almacenado dxxmqShredCLOB" en la página 263
- "Procedimiento almacenado dxxmqInsert" en la página 265
- "Procedimiento almacenado dxxmqInsertAll" en la página 269
- "Procedimiento almacenado dxxmqInsertCLOB" en la página 267
- "Procedimiento almacenado dxxmqGen()" en la página 250
- "Procedimiento almacenado dxxmqShredAllCLOB" en la página 264
- "Procedimiento almacenado dxxmqInsertAllCLOB" en la página 271

Procedimiento almacenado dxxmqGen()

Propósito:

Crea un documento XML a partir de los datos que están almacenados en las tablas de la colección XML especificadas en el archivo DAD y envía el documento XML a una cola de mensajes MQ. El procedimiento almacenado devuelve una serie para indicar el estado del procedimiento almacenado.

Para dar soporte a la consulta dinámica, dxxmqGen() admite un parámetro de entrada, *override*. Basándose en el valor de *overrideType*, la aplicación puede alterar temporalmente la sentencia de SQL_stmt para la correlación de SQL o las condiciones de RDB_node para la correlación de RDB_node del archivo DAD. El parámetro de entrada *overrideType* se utiliza para clarificar el tipo de *override (alteración temporal)*.

Sintaxis:

```
dxxmqGen(varchar(48)    serviceName,     /*entrada*/
         varchar(48)    policyName,      /*entrada*/
         varchar(80)    dadFileName,     /*entrada*/
         integer        overrideType,    /*entrada*/
         varchar(1024)  override,        /*entrada*/
         integer        maxRows,         /*entrada*/
         integer        numRows,         /*salida*/
         char(20)       status)          /*salida*/
```

Parámetros:

Tabla 76. Parámetros de dxxmqGen()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. DB2.DEFAULT.SERIVCE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
dadFileName	Nombre del archivo DAD.	ENTRADA
overrideType	Distintivo para indicar el tipo del siguiente parámetro *override*: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal mediante una sentencia de SQL_stmt. • **XML_OVERRIDE**: Alteración temporal mediante una condición basada en XPath.	ENTRADA
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar dicha correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de 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 utiliza este parámetro con *overrideType*, deberá utilizar la correlación de RDB_node en el archivo DAD.	ENTRADA
maxRows	Número máximo de mensajes que se generan en la cola de mensajes.	ENTRADA
numRows	Número real de filas generadas en la cola de mensajes.	SALIDA

Tabla 76. Parámetros de dxxmqGen() (continuación)

Parámetro	Descripción	Parámetro de E/S
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

El siguiente ejemplo de fragmento genera un documento XML y lo envía a la cola. Se supone que se han definido un servicio MQ/AMI, *myService* y una política, *myPolicy*, en el archivo de depósito. Este archivo guarda las definiciones de depósito en formato XML.

```
#include "dxx.h"
#include "dxxrc.h"
        EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
        char           serviceName[48];    /* nombre del servicio MQ/AMI*/
        char           policyName[48];     /* nombre de la política MQ/AMI*/
        char           dadFileName[80];    /* nombre del archivo DAD */
        char           override[2];        /* alt. temp; se establecerá en NULL*/
        short          overrideType;       /* definido en dxx.h */
        short          max_row;            /* número máximo de filas */
        short          num_row;            /* número real de filas */
        char           status[20]          /* código de estado o mensaje */
        short          ovtype_ind;
        short          ov_ind;
        short          maxrow_ind;
        short          numrow_ind;
        short          dadFileName_ind;
        short          serviceName_ind;
        short          policyName_ind;
        short          status_ind;

     EXEC SQL END DECLARE SECTION;
               strcpy(dadFileName,"c:\dxx\dad\litem3.dad");
          strcpy(serviceName,"myService");
          strcpy(policyName,"myPolicy");
          override[0] = '\0';
          overrideType = NO_OVERRIDE;
          max_row = 500;
          num_row = 0;
              status[0] = '\0';
          dadFileName_ind = 0;
          serviceName_ind = 0;
          policyName_ind = 0;
          maxrow_ind = 0;
          numrow_ind = -1;
               ovtype_ind=0;
               ov_ind=-1;
          status_ind = -1;

         /* Llamar al procedimiento almacenado */
              EXEC SQL CALL dxxmqGen(:serviceName:serviceName_ind,
                :policyName:policyName_ind,
                :dadFileName:dadFileName_ind,
                :overrideType:ovtype_ind,
```

```
:override:ov_ind,
      :max_row:maxrow_ind,
      :num_row:numrow_ind,
      :status:status_ind);
```

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Tareas relacionadas:

- "Invocación de procedimientos almacenados de XML Extender" en la página 199

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqGenCLOB

Propósito:

Crea un documento XML a partir de los datos que están almacenados en las tablas de la colección XML especificadas en el archivo DAD y envía el documento XML a una cola de mensajes MQ. El tipo de documento es XMLCLOB. El procedimiento almacenado devuelve una serie para indicar el estado del procedimiento almacenado. Este procedimiento almacenado no está soportado en Enterprise Server Edition (ESE).

Para dar soporte a la consulta dinámica, dxxmqGenCLOB admite un parámetro de entrada, *override*. Basándose en el valor de *overrideType*, la aplicación puede alterar temporalmente la sentencia de SQL_stmt para la correlación de SQL o las condiciones de RDB_node para la correlación de RDB_node del archivo DAD. El parámetro de entrada *overrideType* se utiliza para clarificar el tipo de *override (alteración temporal)*.

Sintaxis:

```
dxxmqGenCLOB(varchar(48)    serviceName,     /*entrada*/
             varchar(48)    policyName,      /*entrada*/
             varchar(80)    dadFileName,     /*entrada*/
             integer        overrideType,    /*entrada*/
             varchar(1024)  override,        /*entrada*/
             integer        maxRows,         /*entrada*/
             integer        numRows,         /*salida*/
             char(20)       status)          /*salida*/
```

Parámetros:

Tabla 77. Parámetros de dxxmqGenCLOB

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. DB2.DEFAULT.SERIVCE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, el parámetro *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
dadFileName	Nombre del archivo DAD.	ENTRADA
overrideType	Distintivo para indicar el tipo del siguiente parámetro *override*: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal mediante una sentencia de SQL_stmt. • **XML_OVERRIDE**: Alteración temporal mediante una condición basada en XPath.	ENTRADA
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar dicha correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de 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 utiliza este parámetro *overrideType*, deberá utilizar dicha correlación de RDB_node en el archivo DAD.	ENTRADA
maxRows	Número máximo de mensajes que se generan en la cola de mensajes.	ENTRADA
numRows	Número real de filas generadas en la cola de mensajes.	SALIDA

Tabla 77. Parámetros de dxxmqGenCLOB (continuación)

Parámetro	Descripción	Parámetro de E/S
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqRetrieve

Propósito:

El procedimiento almacenado dxxmqRetrieve() sirve como medio para recuperar documentos XML descompuestos. Como entrada, dxxmqRetrieve() admite un almacenamiento intermedio que contiene el nombre de la colección XML habilitada y los nombres de servicio y política MQ/AMI. Envía el documento XML compuesto a una cola MQ; devuelve el número de filas enviadas a la cola y un mensaje de estado. El procedimiento almacenado dxxmqRetrieve permite utilizar el mismo archivo DAD tanto para la composición como para la descomposición.

Para dar soporte a la consulta dinámica, dxxmqRetrieve() admite un parámetro de entrada, *override*. Basándose en el valor de *overrideType*, la aplicación puede alterar temporalmente la sentencia de SQL_stmt, para la correlación de SQL, o las condiciones del RDB_node, para la correlación de RDB_node, en el archivo DAD. El parámetro de entrada *overrideType* se utiliza para clarificar el tipo de *override (alteración temporal)*.

Los requisitos del archivo DAD para dxxmqRetrieve() son los mismos que los requisitos para dxxmqGen(). La única diferencia es que el archivo DAD no es un parámetro de entrada para dxxmqRetrieve(); en su lugar, el parámetro necesario es el nombre de una colección XML habilitada.

Sintaxis:

```
dxxmqRetrieve(varchar(48)    serviceName,        /*entrada*/
              varchar(48)    policyName,         /*entrada*/
              varchar(80)    collectionName,     /*entrada*/
              integer        overrideType,       /*entrada*/
              varchar(1024)  override,           /*entrada*/
              integer        maxrows,            /*entrada*/
              integer        numrows,            /*salida*/
              char(20)       status)             /*salida*/
```

Parámetros:

Tabla 78. Parámetros de dxxmqRetrieve()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
collectionName	Nombre de una colección habilitada.	ENTRADA
overrideType	Distintivo para indicar el tipo del siguiente parámetro *override*: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal mediante una sentencia de SQL_stmt. • **XML_OVERRIDE**: Alteración temporal mediante una condición basada en XPath.	ENTRADA
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar la correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de SQL_stmt especificada en el archivo DAD. • **XML_OVERRIDE**: Serie de caracteres que contiene una o más expresiones entre comillas dobles, separadas por "AND". La longitud máxima es de 1024 bytes. La serie de caracteres *overrideType* requiere que se utilice la correlación de RDB_node en el archivo DAD.	ENTRADA
maxRows	Es el número máximo de filas de la tabla resultante.	ENTRADA

Tabla 78. Parámetros de dxxmqRetrieve() (continuación)

Parámetro	Descripción	Parámetro de E/S
numRows	Es el número real de filas creadas de la tabla resultante.	SALIDA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

El siguiente fragmento es un ejemplo de llamada a dxxmqRetrieve().

```
#include "dxx.h"
#include "dxxrc.h"
     EXEC SQL INCLUDE SQLCA;
     EXEC SQL BEGIN DECLARE SECTION;
     char           serviceName[48];      /* nombre del servicio MQ/AMI*/
     char           policyName[48];       /* nombre de la política MQ/AMI*/
     char           collection[32];       /* nombre de la colección XML */
     char           override[2];          /* alter. temp.; se establecerá en NULL*/
     short          overrideType;         /* definido en dxx.h */
     short          max_row;              /* número máximo de filas */
     short          num_row;              /* número real de filas */
              char        status[20];         /* código de estado o mensaje */
     short          ovtype_ind;
         short   ov_ind;
     short          maxrow_ind;
     short          numrow_ind;
     short          collection_ind;
     short          serviceName_ind;
     short          policyName_ind;
     short          status_ind;

     EXEC SQL END DECLARE SECTION;

     /* inicializar variable del lenguaje principal e indicadores */
     strcpy(collection,"sales_ord");
     strcpy(serviceName,"myService");
     strcpy(policyName,"myPolicy");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
     max_row = 500;
     num_row = 0;
     status[0] = '\0';
     serviceName_ind = 0;
     policyName_ind = 0;
     collection_ind = 0;
     maxrow_ind = 0;
     numrow_ind = -1;
               ovtype_ind=0;
               ov_ind=-1;
     status_ind = -1;

     /* Llamar al procedimiento almacenado */
              EXEC SQL CALL dxxmqRetrieve(:serviceName:serviceName_ind,
          :policyName:policyName_ind,
          :collection:collection_ind,
          :overrideType:ovtype_ind,
```

```
:override:ov_ind,
:max_row:maxrow_ind,
:num_row:numrow_ind,
      :status:status_ind);
```

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqRetrieveCLOB

Propósito:

El procedimiento almacenado dxxmqRetrieveCLOB sirve como medio para recuperar documentos XML descompuestos. Como entrada, dxxmqRetrieveCLOB admite un almacenamiento intermedio que contiene el nombre de la colección XML habilitada y los nombres de servicio y política MQ/AMI. Envía el documento XML compuesto a una cola MQ; y devuelve el número de filas enviadas a la cola y un mensaje de estado. El procedimiento almacenado dxxmqRetrieveCLOB permite utilizar el mismo archivo DAD tanto para la composición como para la descomposición. Este procedimiento almacenado no está soportado en Enterprise Server Edition (ESE).

Para dar soporte a la consulta dinámica, dxxmqRetrieveCLOB admite un parámetro de entrada, *override*. Basándose en el valor de *overrideType*, la aplicación puede alterar temporalmente la sentencia de SQL_stmt, para la correlación de SQL, o las condiciones del RDB_node, para la correlación de RDB_node, en el archivo DAD. El parámetro de entrada *overrideType* se utiliza para clarificar el tipo de *override (alteración temporal)*.

Los requisitos del archivo DAD para dxxmqRetrieveCLOB son los mismos que para dxxmqGenCLOB. La única diferencia es que el archivo DAD no es un parámetro de entrada para dxxmqRetrieveCLOB; en su lugar el parámetro necesario es el nombre de una colección XML habilitada.

Sintaxis:

```
dxxmqRetrieveCLOB(varchar(48)    serviceName,        /*entrada*/
                  varchar(48)    policyName,         /*entrada*/
                  varchar(80)    collectionName,     /*entrada*/
                  integer        overrideType,       /*entrada*/
                  varchar(1024)  override,           /*entrada*/
                  integer        maxrows,            /*entrada*/
                  integer        numrows,            /*salida*/
                  char(20)       status)             /*salida*/
```

Parámetros:

Tabla 79. Parámetros de dxxmqRetrieveCLOB

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
collectionName	Nombre de una colección habilitada.	ENTRADA
overrideType	Distintivo para indicar el tipo del siguiente parámetro *override*: • **NO_OVERRIDE**: Sin alteración temporal. • **SQL_OVERRIDE**: Alteración temporal mediante una sentencia de SQL_stmt. • **XML_OVERRIDE**: Alteración temporal mediante una condición basada en XPath.	ENTRADA
override	Altera temporalmente la condición especificada en el archivo DAD. El valor de entrada se basa en el parámetro *overrideType*. • **NO_OVERRIDE**: Es una serie nula. • **SQL_OVERRIDE**: Es una sentencia de SQL válida. Si utiliza este parámetro con *overrideType*, deberá utilizar la correlación de SQL en el archivo DAD. La sentencia de SQL de entrada altera temporalmente la sentencia de SQL_stmt especificada en el archivo DAD. • **XML_OVERRIDE**: Serie de caracteres que contiene una o más expresiones entre comillas dobles, separadas por "AND". El tamaño máximo es de 1024 bytes. La serie de caracteres *overrideType* requiere que se utilice la correlación de RDB_node en el archivo DAD.	ENTRADA
maxRows	Es el número máximo de filas de la tabla resultante.	ENTRADA
numRows	Es el número real de filas creadas de la tabla resultante.	SALIDA

Tabla 79. Parámetros de dxxmqRetrieveCLOB (continuación)

Parámetro	Descripción	Parámetro de E/S
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqShred

Propósito:

Descompone un documento XML de entrada de una cola de mensajes, basándose en una correlación de archivo DAD, y almacena el contenido de los elementos y atributos XML en tablas de DB2 UDB especificadas.

Para que dxxmqShred() funcione, todas las tablas especificadas en el archivo DAD deben existir y todas las columnas y tipos de datos especificados en el archivo DAD deben ser coherentes con las tablas existentes. El procedimiento almacenado requiere que las columnas especificadas en la condición de unión, en DAD, correspondan a las relaciones de clave primaria-clave foránea en las tablas existentes. Las columnas de la condición de unión que se han especificado en el RDB_node del element_node (nodo de elemento) raíz deben existir en las tablas.

Sintaxis:

```
dxxmqShred(varchar(48)   serviceName,    /* entrada */
           varchar(48)   policyName,     /* entrada */
           varchar(80)   dadFileName,    /* entrada */
           varchar(10)   status)         /* salida */
```

Parámetros:

Tabla 80. Parámetros de dxxmqShred()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA

Tabla 80. Parámetros de dxxmqShred() (continuación)

Parámetro	Descripción	Parámetro de E/S
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
dadFileName	Nombre del archivo DAD. El tamaño máximo es de 80 bytes.	ENTRADA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

El siguiente fragmento es un ejemplo de llamada a dxxmqShred().

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

EXEC SQL INCLUDE SQLCA;
          EXEC SQL BEGIN DECLARE SECTION;
              char        serviceName[48];    /* nombre del servicio MQ/AMI */
              char        policyName[48];     /* nombre de la política MQ/AMI */
              char        dadFileName[80];    /* nombre del archivo DAD */
              char        status[20];         /* código de estado o mensaje */
              short       serviceName_ind;
              short       policyName_ind;
              short       dadFileName_ind;
              short       status_ind;
           EXEC SQL END DECLARE SECTION;

              /* inicializar variable del lenguaje principal e indicadores */
              strcpy(dadFileName,"e:/dxx/samples/dad/getstart_xcollection.dad");
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
            status[0]='\0';
              serviceName_ind=0;
              policyName_ind=0;
              dadFileName_ind=0;
              status_ind=-1;

              /* Llamar al procedimiento almacenado */
              EXEC SQL CALL dxxmqShred(:serviceName:serviceName_ind,
                                       :policyName:policyName_ind,
                                       :dadFileName:dadFileName_ind,
                                       :status:status_ind);
```

Información relacionada:

- Apéndice C, “Límites de XML Extender”, en la página 321

Procedimiento almacenado dxxmqShredAll

Propósito:

Descompone todos los documentos XML de entrada de una cola de mensajes, basándose en una correlación de archivo DAD. El contenido de los elementos y atributos XML se almacena en las tablas de DB2 UDB especificadas.

Para que dxxmqShredAll() funcione, todas las tablas especificadas en el archivo DAD deben existir y todas las columnas y los tipos de datos que estén especificados en la DAD deben ser coherentes con las tablas existentes. El procedimiento almacenado requiere que las columnas especificadas en la condición de unión, en la DAD, correspondan a relaciones de clave primaria-foránea en las tablas existentes. Las columnas de la condición de unión que se han especificado en el RDB_node del element_node (nodo de elemento) raíz deben existir en las tablas.

Sintaxis:

```
dxxmqShredAll(varchar(48)   serviceName,     /* entrada */
              varchar(48)   policyName,     /* entrada */
              varchar(80)   dadFileName,    /* entrada */
              varchar(20)   status)         /* salida */
```

Parámetros:

Tabla 81. Parámetros de dxxmqShredAll()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
dadFileName	Nombre del archivo DAD. El tamaño máximo es de 80 bytes.	ENTRADA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

El siguiente fragmento es un ejemplo de una llamada a dxxmqShredAll().

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

EXEC SQL INCLUDE SQLCA;
          EXEC SQL BEGIN DECLARE SECTION;
              char        serviceName[48];    /* nombre del servicio MQ/AMI */
              char        policyName[48];     /* nombre de la política MQ/AMI */
              char        dadFileName[80];    /* nombre del archivo DAD */
              char        status[20];         /* código de estado o mensaje */
              short       serviceName_ind;
              short       policyName_ind;
              short       dadFileName_ind;
              short       status_ind;
           EXEC SQL END DECLARE SECTION;

          /* inicializar variable del lenguaje principal e indicadores */
              strcpy(dadFileName,"e:/dxx/samples/dad/getstart_xcollection.dad");
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
        status[0]=\0;
              serviceName_ind=0;
              policyName_ind=0;
              dadFileName_ind=0;
              status_ind=-1;

              /* Llamar al procedimiento almacenado */
        EXEC SQL CALL dxxmqShredAll(:serviceName:serviceName_ind,
                                    :policyName:policyName_ind,
                                    :dadFileName:dadFileName_ind,
                                    :status:status_ind);
```

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqShredCLOB

Propósito:

Descompone un documento XML de entrada de una cola de mensajes, basándose en una correlación de archivo DAD, y almacena el contenido de los elementos y atributos XML en tablas de DB2 UDB especificadas. El tipo de documento de entrada es XMLCLOB.

Para dxxmqShredCLOB, todas las tablas especificadas en el archivo DAD deben existir y todas las columnas y tipos de datos que estén especificados en la DAD deben ser coherentes con las tablas existentes. Este procedimiento almacenado requiere que las columnas especificadas en la condición de unión de la DAD correspondan a las relaciones de clave primaria-foránea de las tablas existentes. Las columnas de la condición de unión que se hayan especificado en el RDB_node del element_node (nodo de elemento) raíz deben existir en las tablas.

Sintaxis:

```
dxxmqShredCLOB(varchar(48)   serviceName,    /* entrada */
               varchar(48)   policyName,     /* entrada */
               varchar(80)   dadFileName,    /* entrada */
               varchar(10)   status)         /* salida */
```

Parámetros:

Tabla 82. Parámetros de dxxmqShredCLOB

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
dadFileName	Nombre del archivo DAD. El tamaño máximo es de 80 bytes.	ENTRADA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqShredAllCLOB

Propósito:

Descompone un documento XML de entrada de una cola de mensajes, basándose en una correlación de archivo DAD, y almacena el contenido de los elementos y atributos XML en tablas de DB2 UDB especificadas.

Para dxxmqShredAllCLOB, todas las tablas especificadas en el archivo DAD deben existir y todas las columnas y tipos de datos que se especifiquen en el archivo DAD deben ser coherentes con las tablas existentes. Este procedimiento almacenado requiere que las columnas especificadas en la condición de unión de la DAD correspondan a las relaciones de clave primaria-foránea de las tablas existentes. Las columnas de la condición de unión que estén especificadas en el RDB_node del nodo de elemento (element_node) raíz deben existir en las tablas.

Sintaxis:

```
dxxmqShredCLOB(varchar(48)   serviceName,     /* entrada */
               varchar(48)   policyName,     /* entrada */
               varchar(80)   dadFileName,    /* entrada */
               varchar(10)   status)         /* salida */
```

Parámetros:

Tabla 83. Parámetros de dxxmqShredAllCLOB

Parámetro	Descripción	Parámetro ENTRADA/ SALIDA
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
dadFileName	Nombre del archivo DAD. El tamaño máximo es de 80 bytes.	ENTRADA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Información relacionada:

• Apéndice C, “Límites de XML Extender”, en la página 321

Procedimiento almacenado dxxmqInsert

Propósito:

Fragmenta o desglosa un documento XML de entrada de una cola de mensajes y almacena los datos en tablas de bases de datos nuevas o existentes. dxxmqInsert utiliza un nombre de colección en lugar de un nombre de archivo DAD para determinar cómo almacenar los datos.

Sintaxis:

```
dxxmqInsert(varchar(48)   serviceName,     /* entrada */
            varchar(48)   policyName,      /* entrada */
            varchar(80)   collectionName,  /* entrada */
            varchar(20)   status)          /* salida */
```

Parámetros:

Tabla 84. Parámetros de dxxmqInsert()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
collectionName	Es el nombre de una colección XML habilitada. El tamaño máximo es de 80 bytes.	ENTRADA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

En el siguiente ejemplo de fragmento, la llamada dxxmqInsert() recupera el documento XML de entrada `order1.xml` de una cola de mensajes definida por *serviceName*, descompone el documento e inserta datos en las tablas de la colección SALES_ORDER de acuerdo con la correlación que está especificada en el archivo DAD con el que se ha habilitó el documento.

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

EXEC SQL INCLUDE SQLCA;
          EXEC SQL BEGIN DECLARE SECTION;
              char          serviceName[48];
              char          policyName[48];
              char          collection[80];     /* nombre de una colección XML */
              char          status[10];

              short                 serviceName_ind;
              short                 policyName_ind;
              short                 collection_ind;
              short                 status_ind;
              EXEC SQL END DECLARE SECTION;

              /* inicializar variable del lenguaje principal e  indicadores */
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
              strcpy(collection,"sales_ord")
              status[0]=\0;
              serviceName_ind = 0;
              policyName_ind = 0;
              collection_ind = 0;
              status_ind = -1;

              /* Llamar al procedimiento almacenado */
              EXEC SQL CALL dxxmqInsert(:serviceName:serviceName_ind,
                             :policyName:policyName_ind,
                             :collection:collection_ind,
                             :status:status_ind);
```

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqInsertCLOB

Propósito:

Fragmenta o desglosa un documento XML de entrada de una cola de mensajes y almacena los datos en tablas de bases de datos nuevas o existentes. dxxmqInsertCLOB utiliza un nombre de colección, en lugar de un nombre de archivo DAD, para determinar cómo almacenar los datos. El tipo de documento de entrada es XMLCLOB.

Sintaxis:

```
dxxmqInsertCLOB(varchar(48)  serviceName,  /* entrada */
                varchar(48)   policyName,      /* entrada */
             varchar(80)   collectionName,  /* entrada */
                varchar(20)   status)          /* salida */
```

Parámetros:

Tabla 85. Parámetros de dxxmqInsertCLOB()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
collectionName	Es el nombre de una colección XML habilitada.	ENTRADA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

En el siguiente ejemplo de fragmento, la llamada dxxmqInsertCLOB() recupera el documento XML de entrada `order1.xml` de una cola de mensajes definida mediante *serviceName*, descompone el documento e inserta datos en las tablas de la colección SALES_ORDER de acuerdo con la correlación especificada en el archivo DAD con el que se ha habilitado.

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

EXEC SQL INCLUDE SQLCA;
             EXEC SQL BEGIN DECLARE SECTION;
             char          serviceName[48];
             char          policyName[48];
             char          collection[48];     /* nombre de una colección XML */
             char          status[10];

          short          serviceName_ind;
          short          policyName_ind;
              short                collection_ind;
          short          status_ind;
         EXEC SQL END DECLARE SECTION;
```

```
     /* inicializar variable del lenguaje principal e indicadores */
     strcpy(serviceName, "myService");
     strcpy(policyName, "myPolicy");
strcpy(collection,"sales_ord")
     status[0] = \0;
     serviceName_ind = 0;
     policyName_ind = 0;
collection_ind = 0;
     status_ind = -1;

     /* Llamar al procedimiento almacenado */
     EXEC SQL CALL dxxmqInsertCLOB(:serviceName:serviceName_ind;
                    :policyName:policyName_ind,
                    :collection:collection_ind,
                    :status:status_ind);
```

Información relacionada:

- Apéndice C, “Límites de XML Extender”, en la página 321

Procedimiento almacenado dxxmqInsertAll

Propósito:

Fragmenta o desglosa todos los documentos XML de entrada de una cola de mensajes y almacena los datos en tablas de bases de datos nuevas o existentes. dxxmqInsertAll utiliza un nombre de colección, en lugar de un nombre de archivo DAD para determinar cómo almacenar los datos.

Sintaxis:

```
dxxmqInsertAll(varchar(48) serviceName,    /* entrada */
               varchar(48)   policyName,      /* entrada */
               varchar(48)   collectionName,  /* entrada */
               varchar(20)   status)          /* salida */
```

Parámetros:

Tabla 86. Parámetros de dxxmqInsertAll()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA

Tabla 86. Parámetros de dxxmqInsertAll() (continuación)

Parámetro	Descripción	Parámetro de E/S
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
collectionName	Es el nombre de una colección XML habilitada. El tamaño máximo es de 80 bytes.	ENTRADA
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

En el siguiente ejemplo de fragmento, la llamada dxxmqInsertAll recupera todos los documentos XML de entrada de una cola de mensajes definida por *serviceName*, descompone los documentos e inserta datos en las tablas de la colección SALES_ORDER de acuerdo con la correlación que está especificada en el archivo DAD con el que se habilitó el documento.

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

EXEC SQL INCLUDE SQLCA;
          EXEC SQL BEGIN DECLARE SECTION;
              char          serviceName[48];
              char          policyName[48];
              char          collection[80];     /* nombre de una colección XML */
              char          status[10];

              short                 serviceName_ind;
              short                 policyName_ind;
              short                 collection_ind;
              short                 status_ind;
              EXEC SQL END DECLARE SECTION;

              /* inicializar variable del lenguaje principal e indicadores */
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
         strcpy(collection,"sales_ord");
              status[0]='\0';
              serviceName_ind = 0;
              policyName_ind = 0;
              collection_ind = 0;
              status_ind = -1;

              /* Llamar al procedimiento almacenado */
              EXEC SQL CALL dxxmqInsertAll(:serviceName:serviceName_ind,
```

```
:policyName:policyName_ind,
:collection:collection_ind,
:status:status_ind);
```

Conceptos relacionados:

- "Procedimientos almacenados y funciones de XML Extender para MQSeries - Visión general" en la página 225

Información relacionada:

- "Cómo leer los diagramas de sintaxis" en la página ix
- Apéndice C, "Límites de XML Extender", en la página 321

Procedimiento almacenado dxxmqInsertAllCLOB

Propósito:

Fragmenta o desglosa todos los documentos XML de entrada de una cola de mensajes y almacena los datos en tablas de bases de datos nuevas o existentes. El procedimiento almacenado dxxmqInsertAllCLOB utiliza un nombre de colección, en lugar de un nombre de archivo DAD, para determinar cómo almacenar los datos.

Sintaxis:

```
dxxmqInsertAllCLOB(varchar(48)  serviceName,  /* entrada */
                   varchar(48)   policyName,      /* entrada */
                   varchar(48)   collectionName,  /* entrada */
                   varchar(20)   status)          /* salida */
```

Parámetros:

Tabla 87. Parámetros de dxxmqInsertAllCLOB()

Parámetro	Descripción	Parámetro de E/S
serviceName	Serie que contiene el destino de MQSeries lógico al que se va a enviar el mensaje. Cuando *serviceName* aparece listado, hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. El valor de DB2.DEFAULT.SERVICE se utiliza cuando no se especifica *serviceName*. El tamaño máximo de *serviceName* es de 48 bytes.	ENTRADA
policyName	Serie que contiene la política de servicios AMI de MQSeries que se utiliza para manejar mensajes. Cuando se especifica, *policyName* hace referencia a una política definida en el archivo de depósito AMT.XML. Si no se especifica *policyName*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *policyName* es de 48 bytes.	ENTRADA
collectionName	Es el nombre de una colección XML habilitada.	ENTRADA

Tabla 87. Parámetros de dxxmqInsertAllCLOB() (continuación)

Parámetro	Descripción	Parámetro de E/S
status	El texto y los códigos devueltos que especifican si el procedimiento almacenado se ha ejecutado o no satisfactoriamente, los códigos de error que se generan y el número de documentos XML que se reciben o se envían a la cola de mensajes.	SALIDA

Ejemplos:

En el ejemplo de fragmento siguiente, la llamada dxxmqInsertAllCLOB recupera todos los documentos XML de entrada de una cola de mensajes definida por *serviceName*, descompone los documentos e inserta datos en las tablas de la colección SALES_ORDER de acuerdo con la correlación especificada en el archivo DAD con el que se ha habilitado.

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

EXEC SQL INCLUDE SQLCA;
       EXEC SQL BEGIN DECLARE SECTION;
              char          serviceName[48];
              char          policyName[48];
              char          collection[48];     /* nombre de una colección XML */
              char          status[10];

          short          serviceName_ind;
          short          policyName_ind;
         short                collection_ind;
          short          status_ind;
    EXEC SQL END DECLARE SECTION;

         /* inicializar variable del lenguaje principal e indicadores */
              strcpy(serviceName, "myService");
              strcpy(policyName, "myPolicy");
         strcpy(collection,"sales_ord")
              status[0] = '\0';
              serviceName_ind = 0;
              policyName_ind = 0;
         collection_ind = 0;
              status_ind = -1;

            /* Llamar al procedimiento almacenado */
              EXEC SQL CALL dxxmqInsertAllCLOB(:serviceName:serviceName_ind;
                             :policyName:policyName_ind,
                             :collection:collection_ind,
                             :status:status_ind);
```

Información relacionada:

- Apéndice C, “Límites de XML Extender”, en la página 321

Capítulo 12. Extensible stylesheet language transformation (XSLT)

Creación de un documento XML utilizando una hoja de estilo XSLT

XSLT (Extensible Stylesheet Language Transformation) consta de una serie de markups que se pueden utilizar para aplicar normas de formato a cada uno de los elementos de un documento XML. XSL funciona aplicando diversas normas de estilo al contenido de un documento XML basándose en los elementos que encuentra. Según el diseño, las hojas de estilo XSLT son documentos XML corrientes.

Creado en un principio para el diseño de páginas, XSLT se utiliza ahora de diversas formas. Por ejemplo, se puede utilizar como herramienta de conversión de ámbito general, como sistema para reorganizar el contenido del documento o como método para generar varios resultados, como por ejemplo, HTML, WAP y SVG desde una sola fuente.

XSLT es un puente muy importante entre el proceso XML y lenguajes más familiares como HTML. XSLT permite transformar una estructura XML en otros tipos de datos eliminando o sustituyendo los códigos XML. También permite cambiar el orden de la información, extraer determinada información especial o clasificarla.

Requisitos previos:

Para crear un documento HTML utilizando una hoja de estilo, necesita completar las siguientes tareas:

1. Crear un archivo XML en la tabla resultante.
2. Crear una hoja de estilo.

Puede crear un archivo HTML utilizando XSLTransformToFile o XSLTransformToClob. Este archivo de salida puede grabarse en el servidor de DB2 UDB o bien desde la línea de mandatos en un editor de texto.

Procedimiento:

Para crear el archivo HTML en el servidor DB2 UDB, escriba la siguiente sintaxis:

```
SELECT XSLTransformToFile( CAST(doc AS CLOB(4k)),
    '$instalación_dxx$\samples\db2xml\xslt\getstart.xsl',
         0,
   '$instalación_dxx$\samples\db2xml\html\getstart.html')
 FROM RESULT_TAB
```

donde *$instalación_dxx$* es el directorio donde se ha instalado DB2 XML Extender.

Para crear el archivo HTML desde la línea de mandatos, abra cualquier editor de texto y escriba el siguiente mandato:

```
getstart_xslt.cmd
```

Información relacionada:

- "Procedimiento almacenado XSLTransformToClob()" en la página 274

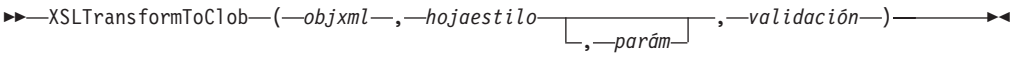
• "Procedimiento almacenado XSLTransformToFile()" en la página 275

Procedimiento almacenado XSLTransformToClob()

Propósito:

XSLTransformToClob() lee un documento XML como un localizador CLOB y una hoja de estilo como CLOB o desde un archivo, y devuelve el documento como CLOB.

Sintaxis:



Parámetros:

Parámetro	Tipo de datos	Descripción
objxml	CLOB	Es el documento XML.
hojaestilo	CLOB, VARCHAR	Es la hoja de estilo. La ubicación y el nombre del archivo de entrada de hoja de estilo.
parám	CLOB VARCHAR	Es el documento de los parámetros XML. La ubicación y el nombre del archivo de los parámetros XML.
validación	INTEGER	Habilitar (1) o inhabilitar (0) la validación del objxml.

Resultados:
XSLTransformToClob() devuelve unos datos de tipo CLOB, en caso de que sea satisfactorio.
Ejemplos:
Los siguientes ejemplos crean una tabla de ejemplo y almacenan los dos archivos de entrada en la base de datos: `getstart.xml` y `getstart.xsl`. La base de datos debe estar habilitada para XML Extender.

```
CREATE TABLE xslt_tab(xmlobj CLOB(4k), stylesheet CLOB(4k))
INSERT INTO xslt_tab(xmlobj, stylesheet) VALUES(
 DB2XML.XMLCLOBFromFile('c:\instalación_dxx\samples\db2xml\xml\getstart.xml
'),
DB2XML.XMLCLOBFromFile('c:\instalación_dxx\samples\db2xml\xslt\getstart.xsl
'))
```

Ejemplo 1: El ejemplo siguiente transforma un documento XML en un documento HTML utilizando la tabla creada:

```
SELECT XSLTransformToClob(xmlobj, stylesheet)
FROM xslt_tab
```

Ejemplo 2: Este ejemplo transforma un documento XML en un documento HTML utilizando un archivo de hoja de estilo:

```
SELECT XSLTransformToClob( xmlobj,
 c:\instalación_dxx\samples\db2xml\xslt\getstart.xsl
')
  FROM xslt_tab
```

Ejemplo 3: En este ejemplo, la salida se cambia mediante parámetros adicionales. El documento de los parámetros XML debe definir el espacio de nombres. Los parámetros se deben acomodar en el elemento <*param*>. El valor correspondiente también se puede especificar en un atributo *value* o bien en el contenido del elemento <*param*>.

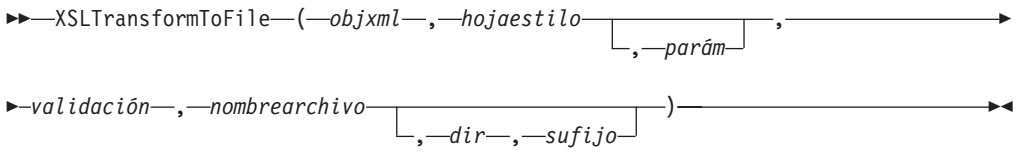
```
c:\instalación_dxx\samples\db2xml\xml\getstart_xslt_param.xml:
<?xml version="1.0"?>
<params xmlns="http://www.ibm.com.XSLtransformParameters">
   <param name="noShipments" value="true"/>
   <param name="headline">The customers...</param>
</params>

SELECT XSLTranfsormToClob( xmlobj, stylesheet, param, 1)
FROM xslt_tab
```

Procedimiento almacenado XSLTransformToFile()

Propósito:

Lee un documento XML como un CLOB y una hoja de estilo como un CLOB o desde un archivo. Luego, la función definida por el usuario (UDF) XSLTransformToFile() graba los resultados de la hoja de estilo y del documento XML en un archivo. Cuando un directorio y una extensión de archivo se suministran como parámetros, la UDF creará un archivo con un nombre de archivo exclusivo en este directorio.

Sintaxis:



Parámetros:

Tabla 88. Descripciones de parámetros de XSLTransformDir()

Parámetro	Tipo de datos	Descripción
objxml	CLOB	Es el documento XML.
hojaestilo	CLOB VARCHAR	Es la hoja de estilo. La ubicación y el nombre del archivo de entrada de hoja de estilo.
parám	VARCHAR VARCH	Es el documento de los parámetros XML. La ubicación y el nombre del archivo de los parámetros XML.
validación	INTEGER	Habilitar (1) o inhabilitar (0) la validación del objxml.

Tabla 88. Descripciones de parámetros de XSLTransformDir() (continuación)

Parámetro	Tipo de datos	Descripción
nombrearchivo	VARCHAR	Es el nombre del archivo de salida.
dir	VARCHAR	Es el directorio del archivo de salida.
sufijo	VARCHAR	Es el sufijo del archivo de salida.

Resultados:
XSLTransformToFile() devuelve un VARCHAR para el nombre de archivo grabado.
Ejemplos:
En el ejemplo siguiente, se crea una tabla de ejemplo y se almacenan dos archivos en las tablas `getstart.xml` y `getstart.xsl`. Para crear la tabla de ejemplo, la base de datos DB2 UDB debe estar habilitada para XML Extender.

```
CREATE TABLE xslt_tab(xmlobj CLOB(4k), stylesheet CLOB(4k))

INSERT INTO xslt_tab(xmlobj, stylesheet) VALUES(
DB2XML.XMLCLOBFromFile('$instalación_dxx$\samples\db2xml\xml\getstart.xml
'),
DB2XML.XMLCLOBFromFile('$instalación_dxx$\samples\db2xml\xslt\getstart.xsl
'))
```

donde *$instalación_dxx$* es el directorio donde se ha instalado DB2 XML Extender.

Ejemplo 1: Este ejemplo transforma el documento XML en un documento HTML y graba el documento creado en el archivo especificado:

```
SELECT XSLTransformFile( xmlobj, stylesheet,
   '$instalación_dxx$\samples\db2xml\html\getstart.html

  FROM xslt_tab
```

donde *$instalación_dxx$* es el directorio donde se ha instalado DB2 XML Extender.

Ejemplo 2: Este ejemplo graba un documento HTML en un archivo utilizando un archivo de hoja de estilo. Se habilita la validación, pero el resultado es el mismo. Esta característica es necesaria para incluir los valores por omisión de un esquema XML en el proceso de transformación. No se especifican parámetros. La UDF genera el nombre de archivo.

```
SELECT XSLTransformToFile( xmlobj,
 '$instalación_dxx$\samples\db2xml\xslt\getstart.xsl',
 '$instalación_dxx$\samples\db2xml\html\getstart.html')
     FROM xslt_tab
```

donde *$instalación_dxx$* es el directorio donde se ha instalado DB2 XML Extender.

Ejemplo 3: En este ejemplo, la salida se cambia mediante parámetros adicionales. El documento de los parámetros XML debe definir el espacio de nombres. Los parámetros se deben acomodar en el elemento *<param>*. El valor correspondiente también se puede especificar en un atributo *value* o bien en el contenido del elemento *<param>*.

```
$instalación_dxx$\samples\db2xml\xml\getstart_xslt_param.xml:', 'html')
<?xml version="1.0"?>
<params xmlns="http://www.ibm.com.XSLtransformParameters">
   <param name="noShipments" value="true"/>
   <param name="headline">The customers...</param>
</params>
```

donde *$instalación_dxx$* es el directorio donde se ha instalado DB2 XML Extender.

Ejemplo 4: Este ejemplo graba un documento HTML en un archivo utilizando un archivo de hoja de estilo y almacena el nombre de archivo para cada fila en una columna adicional de la tabla.

```
UPDATE TABLE xslt_tab ADD COLUMN filename VARCHAR(512)
UPDATE TABLE xslt_tab SET filename =
  XSLTransformToFile(xmlobj,stylesheet, param, 1,
    '$instalación_dxx$\samples\db2xml\html
          ', 'html')
  FROM xslt_tab
```

donde *$instalación_dxx$* es el directorio donde se ha instalado DB2 XML Extender.

Capítulo 13. Tablas de soporte de administración de XML Extender

Cuando se habilita una base de datos, se crea una tabla de depósito de 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.

Tabla de referencia de DTD

XML Extender también funciona como un depósito de DTD de XML. Cuando se habilita para XML una base de datos, se crea una tabla de depósito DTD denominada DTD_REF. Cada fila de esta tabla representa una DTD, junto con información adicional de metadatos. Puede 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 89.

Tabla 89. 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 DTD en el archivo DAD, el archivo DAD 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)	El autor de la DTD. Esta información es opcional.
CREATOR	VARCHAR(128)	ID de usuario que realiza la primera inserción. Esta columna es opcional.
UPDATOR	VARCHAR(128)	ID de usuario que realiza la última actualización. Esta columna es opcional.

La aplicación sólo puede modificar la DTD cuando USAGE_COUNT sea igual a cero.

Tabla de utilización de XML (XML_USAGE)

La tabla XML_USAGE contiene 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*). La tabla XML_USAGE se crea en el momento de habilitar la base de datos. Las columnas en la tabla XML_USAGE se muestran en la tabla Tabla 90 en la página 280.

Tabla 90. Tabla XML_USAGE

Nombre de columna	Descripción
esquema_tabla	Para una columna XML, es el nombre de esquema de la tabla de usuario que contiene la columna XML. Para una colección XML, es el valor de DXX_COLL como nombre de esquema por omisión.
nombre_tabla	Para una columna XML, es el nombre de la tabla de usuario que contiene una columna XML. Para una colección XML, es un valor DXX_COLLECTION, que identifica la entidad como una colección.
nombre_col	Es el nombre de la columna XML o colección XML. Forma parte de la clave compuesta junto con nombre_tabla.
DTDID	Serie que asocia una DTD insertada en DTD_REF con una DTD especificada en un archivo DAD; este valor debe coincidir con el elemento DTDID en el archivo DAD. Esta columna es una clave foránea.
DAD	El contenido del archivo DAD que se asocia con la columna XML o colección XML.
modalidad_acceso	Especifica la modalidad de acceso que se utiliza: 1 para la colección XML, 0 para la columna XML
vista_por_omisión	Almacena el nombre de la vista por omisión, si existe alguna.
sufijo_activador	No es NULL. Se utiliza para nombres de activadores exclusivos.
validación	Tiene un valor de 1 para validar o de 0 para pasar por alto la validación.

No añada, modifique o suprima entradas de la tabla XML_USAGE; sólo es para uso interno de XML Extender.

Capítulo 14. Resolución de problemas

Resolución de problemas en XML Extender

Todas las sentencias de SQL incorporado y las llamadas de la interfaz de la línea de mandatos (CLI) de DB2 UDB del programa, incluyendo aquellas que invocan las funciones definidas por el usuario (UDF) de DB2 UDB XML Extender, generan códigos que indican si la sentencia de SQL o la llamada a la CLI de DB2 se han ejecutado satisfactoriamente.

El programa puede recuperar información que sirva de suplemento a estos códigos incluidos la información y los mensajes de error de SQLSTATE. El usuario puede utilizar esta información de diagnóstico para identificar y corregir problemas en el programa.

A veces, el origen de un problema no se puede diagnosticar con facilidad. En estos casos, es posible que necesite proporcionar esta información al soporte de software de IBM para poder aislar y resolver el problema. XML Extender incluye un recurso de rastreo que registra la actividad de XML Extender. La información de rastreo puede constituir una información valiosa para el soporte de software de IBM. Sólo debe utilizar el recurso de rastreo cuando así se lo indique el soporte de software IBM.

Esta sección describe el recurso de rastreo, los códigos y mensajes de error.

Información relacionada:

- "Códigos SQLSTATE y números de mensaje asociados para XML Extender" en la página 283
- "Mensajes de XML Extender" en la página 288
- "Detención del rastreo" en la página 282
- "Inicio del rastreo de XML Extender" en la página 281

Inicio del rastreo de XML Extender

Propósito:

Registra la actividad del servidor de XML Extender. Para iniciar el rastreo, aplique la opción `on` a **dxxtrc**, junto con el nombre de un directorio existente 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 UDB tiene su propio archivo de anotaciones. El archivo de rastreo no tiene límites de tamaño.

Sintaxis:

Inicio del rastreo:

Parámetros:

Tabla 91. Parámetros de rastreo

Parámetro	Descripción
directorio_rastreo	Es el nombre de una vía de acceso y directorio existente donde se coloca dxx*INSTANCE*.trc. Es obligatorio; no es el valor por omisión.

Ejemplos:
El siguiente ejemplo muestra el inicio del rastreo para la instancia db2inst1. El archivo de rastreo, dxxdb2inst1.trc, está situado en el directorio /home/db2inst1/instalación_dxx/log.

```
dxxtrc on
/home/db2inst1/instalación_dxx/log
```

Detención del rastreo

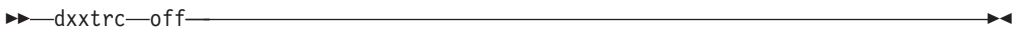
Propósito:

Desactiva el rastreo. Deja de anotarse información de rastreo.

Recomendación: Debido a que el archivo de registro de rastreo no está limitado y puede afectar al rendimiento, es aconsejable desactivar el rastreo al trabajar en un entorno de producción.

Sintaxis:

Detención del rastreo:



Ejemplos:
Este ejemplo muestra que el recurso de rastreo se desactiva.

```
dxxtrc off
```

Códigos de retorno de las UDF de XML Extender

Las sentencias de SQL incorporado devuelven códigos de retorno 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, consulte 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 acerca de la recuperación de errores con la función SQLError, consulte la publicación *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). Un valor de SQLCODE 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 de XML Extender encuentra una condición de aviso o de error, transfiere la información asociada a DB2 UDB para su inclusión en el mensaje SQLCODE.

Las sentencias de SQL incorporado y las llamadas de la CLI de DB2 UDB que invocan UDF de DB2 XML Extender pueden devolver mensajes SQLCODE y valores SQLSTATE exclusivos de dichas UDF, pero DB2 UDB devuelve dichos valores de la misma manera que lo hace con otras sentencias SQL incorporadas u otras llamadas de la CLI de DB2 UDB. Por lo tanto, la manera que se accede a estos es la misma para sentencias de SQL incorporado o para llamadas de la CLI de DB2 UDB que no inician las UDF de DB2 UDB XML Extender.

Códigos de retorno de los procedimientos almacenados de XML Extender

XML Extender proporciona códigos de retorno para ayudar a resolver los problemas que se producen en los procedimientos almacenados. Cuando reciba un código de retorno de un procedimiento almacenado, el archivo que se indica a continuación, que asocia el código de retorno con un número de mensaje de error de XML Extender y la constante simbólica.

`instalación_dxx/include/dxxrc.h`

Información relacionada:

- "Códigos SQLSTATE y números de mensaje asociados para XML Extender" en la página 283

Códigos SQLSTATE y números de mensaje asociados para XML Extender

Tabla 92. Códigos de SQLSTATE y números de mensaje asociados

SQLSTATE	Número de 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	XML Extender no puede abrir el archivo especificado.
38X01	DXXA072E	XML Extender ha intentado vincular automáticamente la base de datos antes de habilitarla, pero no ha podido encontrar los archivos de vinculación.
	DXXC001E	XML Extender no pudo encontrar el archivo especificado.
38X02	DXXC002E	XML Extender no puede leer datos del archivo especificado.
38X03	DXXC003E	XML Extender no puede grabar datos en el archivo.

Tabla 92. Códigos de SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Número de mensaje	Descripción
	DXXC011E	XML Extender no puede grabar datos en el archivo de control de rastreo.
38X04	DXXC004E	XML Extender no ha podido realizar operaciones en el localizador especificado.
38X05	DXXC005E	El tamaño del archivo es mayor que el tamaño de XMLVarchar y XML Extender no puede importar todos los datos del archivo.
38X06	DXXC006E	El tamaño del archivo es mayor que el tamaño de XMLCLOB y XML Extender no puede importar todos los datos del archivo.
38X07	DXXC007E	El número de bytes en el localizador LOB no es igual que el tamaño del archivo.
38X08	DXXD001E	Una función de extracción escalar ha utilizado una vía de ubicación que aparece varias veces. Una función escalar sólo puede utilizar una vía de ubicación que no tenga apariciones múltiples.
38X09	DXXD002E	La expresión de la vía de búsqueda tiene una sintaxis incorrecta.
38X10	DXXG002E	XML Extender no ha podido asignar memoria del sistema operativo.
38X11	DXXA009E	Este procedimiento almacenado es sólo para una columna XML.
38X12	DXXA010E	Mientras intentaba habilitar una columna, XML Extender no pudo encontrar el ID de DTD, que es el identificador especificado para la DTD en el archivo DAD (definición de acceso a documento).
	DXXQ060E	XML Extender no ha podido encontrar el ID de SCHEMA mientras intentaba habilitar la columna. El ID de SCHEMA corresponde al valor del atributo de localización del código nonamespacelocation que se encuentra dentro del código schemabindings en el archivo DAD.
38X14	DXXD000E	Se ha intentado almacenar un documento no válido en una tabla. Ha fallado la validación.
38X15	DXXA056E	El elemento de validación en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente.

Tabla 92. Códigos de SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Número de mensaje	Descripción
	DXXA057E	El atributo de nombre de una tabla auxiliar contenido en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente.
	DXXA058E	El atributo de nombre de una columna contenido en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente.
	DXXA059E	El atributo de tipo de una columna contenido en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente.
	DXXA060E	El atributo de vía de acceso de una columna en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente.
	DXXA061E	El atributo multi_occurrence de una columna contenido en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente.
	DXXQ000E	Falta un elemento obligatorio en el archivo DAD (definición de acceso a documento).
	DXXQ056E	El elemento/atributo especificado no puede correlacionarse con una columna especificada como parte de una clave foránea. Los valores de datos para claves foráneas vienen determinados por los de las claves primarias; una correlación del elemento/atributo especificado en el documento XML con una tabla y columna no es necesaria.
	DXXQ057E	Los códigos schemabindings y DTD ID no pueden coexistir en el archivo DAD.
	DXXQ058E	Falta el código nonamespacelocation dentro del código schemabindings en el archivo DAD.
	DXXQ059E	No es posible encontrar el código doctype dentro del código XCollection en la DAD para la validación del esquema.
	DXXQ062E	Esta condición de error suele estar causada porque falta una especificación multi_occurrence = YES en el element_node padre del elemento o atributo proporcionado.

Tabla 92. Códigos de SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Número de mensaje	Descripción
	DXXQ063E	El valor del atributo multi_occurrence en el element_node especificado en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente. El valor debe ser 'yes' o 'no', en mayúsculas o minúsculas.
	DXXQ064E	Una columna clave especificada en la condición de unión no se ha correlacionado con ningún nodo de elemento o atributo.
38X16	DXXG004E	Se ha pasado un valor nulo a un procedimiento almacenado de XML para un parámetro obligatorio.
38X17	DXXQ001E	La sentencia de SQL en el archivo DAD (definición de acceso a documento) o la sentencia que la altera temporalmente no es válida. Es necesaria una sentencia SELECT para generar documentos XML.
38X18	DXXG001E	XML Extender ha encontrado un error interno.
	DXXG006E	XML Extender ha encontrado un error interno mientras utilizaba la CLI.
38X19	DXXQ002E	El sistema se está quedando 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 devuelve el número de documentos especificado.
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 attribute_node contenido en el archivo DAD (definición de acceso a documento) no tiene un atributo de nombre.
38X24	DXXQ007E	El elemento attribute_node contenido en el archivo DAD (definición de acceso a documento)no tiene un elemento de columna ni un RDB_node.

Tabla 92. Códigos de SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Número de mensaje	Descripción
38X25	DXXQ008E	Un elemento text_node contenido en el archivo de definición de acceso a documento (archivo DAD) no tiene un elemento de columna.
38X26	DXXQ009E	La tabla resultante especificada no se ha podido encontrar en el catálogo del sistema.
38X27	DXXQ010E DXXQ040E	El RDB_node del attribute_node o del text_node debe tener una tabla.
	DXXQ011E	El RDB_node del attribute_node o del text_node debe tener una columna.
	DXXQ017E	XML Extender ha generado un documento XML que es demasiado grande para caber en la columna de la tabla resultante.
	DXXQ040E	El nombre de elemento especificado en el archivo DAD (definición de acceso a documento) es incorrecto.
38X28	DXXQ012E	XML Extender no pudo encontrar el elemento esperado mientras procesaba la DAD.
	DXXQ016E	Todas las tablas deben estar definidas en el RDB_node del elemento superior, en el archivo DAD (definición de acceso a documento). Las tablas de subelementos deben coincidir con las tablas definidas en el elemento superior. El nombre de tabla de este RDB_node no está en el elemento superior.
38X29	DXXQ013E	El elemento de tabla o columna debe tener un nombre en el archivo DAD (definición de acceso a documento).
	DXXQ015E	La condición en el elemento de condición del archivo DAD (definición de acceso a documento) tiene un formato no válido.
	DXXQ061E	El formato de la representación de la serie no es válido. Si la serie es un valor de fecha, hora o de indicación de la hora, la sintaxis no se ajusta a su tipo de datos.
38X30	DXXQ014E	Un elemento element_node contenido en el archivo DAD (definición de acceso a documento) no tiene un atributo de nombre.

Tabla 92. Códigos de SQLSTATE y números de mensaje asociados (continuación)

SQLSTATE	Número de mensaje	Descripción
	DXXQ018E	Falta la cláusula ORDER BY en la sentencia de SQL que correlaciona SQL con XML, y que está contenida en un archivo DAD (definición de acceso a documento).
38X31	DXXQ019E	El elemento objids no tiene un elemento de columna en el archivo DAD (definición de acceso a documento) que correlaciona SQL con XML.
38x33	DXXG005E	Este parámetro no está soportado en este release. Está soportado en futuros releases.
38x34	DXXG000E	Se ha especificado un nombre de archivo no válido.
38X36	DXXA073E	La base de datos no estaba vinculada cuando se ha intentado habilitar.
38X37	DXXG007E	El entorno local del sistema operativo del servidor no es coherente con la página de códigos de DB2 UDB.
38X38	DXXG008E	No se puede encontrar el entorno local del sistema operativo del servidor en la tabla de la página de códigos.
38X41	DXXQ048E	El procesador de la hoja de estilo ha devuelto un error interno. Es posible que el documento XML o la hoja de estilo no sean válidos.
38X42	DXXQ049E	El archivo de salida especificado ya existe en este directorio.
38X43	DXXQ050E	La UDF no ha podido crear un nombre de archivo exclusivo para el documento de salida en el directorio especificado porque no puede acceder al mismo. Es posible que todos los nombres de archivo que pueden generarse ya estén en uso o que el directorio no exista.
38X44	DXXQ051E	Uno o más parámetros de entrada o de salida no tienen un valor válido.
38X45	DXXQ055E	Error ICU encontrado durante la operación de conversión.

Mensajes de XML Extender

DXXA000I **Habilitando 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 la 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 proveedor de servicios de software. Cuando informe del error, asegúrese de incluir todo el texto del mensaje, el archivo de rastreo y una explicación sobre cómo reproducir el problema.

DXXA002I **Conectándose a la base de datos** <*base de datos*>**.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA003E **No es posible conectarse a la base de datos** <*base de 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 lo está, solicite al administrador de la base de datos su recuperación a partir de una copia de seguridad.

DXXA004E **No es posible habilitar la base de datos** <*base de 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 lo está, solicite al administrador de la base de datos su recuperación a partir de una copia de seguridad.

DXXA005I **Habilitando base de datos** <*base de datos*>**. Por favor, espere.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA006I **La base de datos** <*base de datos*> **se ha habilitado satisfactoriamente.**

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA007E **No es posible inhabilitar la base de datos** <*base de datos*>**.**

Explicación: 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 **Inhabilitando 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 para la Columna XML.

Respuesta del Usuario: Asegúrese de que el código Xcolumn esté especificado correctamente en el archivo DAD.

DXXA010E **El intento de encontrar el DTDID** <*id_dtd*> **ha fallado.**

Explicación: Mientras intentaba habilitar una columna, XML Extender no pudo encontrar el ID de DTD, que es el identificador especificado para la DTD en el archivo DAD (definición de acceso a documento).

Respuesta del Usuario: Compruebe si está especificado el valor correcto para el DTDID en el archivo DAD.

DXXA011E **No se ha podido insertar un registro en la tabla de 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, 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 Ha fallado el intento de alterar la tabla <*nombre_tabla*>**.**

Explicación: Mientras intentaba habilitar una columna, XML Extender no pudo modificar la tabla especificada.

Respuesta del Usuario: Compruebe los privilegios necesarios para modificar la tabla.

DXXA014E La columna de ID raíz especificada: <*id_raíz*> **no es una clave primaria única 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 El DXXROOT_ID de columna ya existe en la tabla <*nombre_tabla*>**.**

Explicación: La columna DXXROOT_ID existe, pero no fue creada por 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: 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: 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, 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 encuentran las columnas en la tabla <*nombre_tabla*>**.**

Explicación: Mientras intentaba crear la vista por omisión, 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 por omisión <*vista_por_omisión*>**.**

Explicación: Mientras intentaba habilitar una columna, XML Extender no pudo crear la vista especificada.

Respuesta del Usuario: Compruebe si el nombre de la vista por omisión es exclusivo. Si ya existe una vista con el nombre especificado, especifique un nombre exclusivo para la vista por omisión.

DXXA022I Columna <*nombre_columna*> **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, XML Extender no pudo encontrar el archivo DAD (definición de acceso a documento).

Respuesta del Usuario: Compruebe que ha especificado nombres correctos para la base de datos, la tabla y la columna.

DXXA024E XML Extender ha encontrado un error interno al acceder a las tablas de catálogo del sistema.

Explicación: 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 descartar la vista por omisión <*vista_por_omisión*>.

Explicación: Mientras intentaba inhabilitar una columna, XML Extender no pudo eliminar la vista por omisión.

Respuesta del Usuario: Compruebe que el ID de usuario del administrador para XML Extender tiene los privilegios necesarios para eliminar la vista por omisión.

DXXA026E No se puede descartar la tabla auxiliar <*tabla_auxiliar*>.

Explicación: Mientras intentaba inhabilitar una columna, XML Extender no pudo eliminar la tabla especificada.

Respuesta del Usuario: Compruebe que el ID de usuario del administrador para 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 activador interno. Causas posibles:

- Falta memoria en el sistema.
- No existe un activador con este nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- Falta memoria en el sistema.
- No existe un activador con este nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- Falta memoria en el sistema.
- No existe un activador con este nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- Falta memoria en el sistema.
- No existe un activador con este nombre.

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 proveedor de servicios 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, 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 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, 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 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, 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 XML Extender tiene los privilegios necesarios para actualizar la tabla.

DXXA034I XML Extender ha inhabilitado satisfactoriamente la columna <*nombre_columna*>.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA035I XML Extender está inhabilitando la base de datos <*base de datos*>. Por favor, espere.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA036I XML Extender ha inhabilitado satisfactoriamente la base de datos <*base de datos*>.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA037E El nombre especificado de espacio de tabla tiene más de 18 caracteres.

Explicación: El nombre del espacio de tabla 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 por omisión tiene más de 18 caracteres.

Explicación: El nombre de la vista por omisión 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, XML Extender no pudo crear la tabla auxiliar especificada.

Respuesta del Usuario: Compruebe que el ID de usuario del administrador para 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 activador interno. Causas posibles:

- El archivo DAD tiene una sintaxis incorrecta.
- Falta memoria en el sistema.
- Existe otro activador con el mismo nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- El archivo DAD tiene una sintaxis incorrecta.
- Falta memoria en el sistema.
- Existe otro activador con el mismo nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- El archivo DAD tiene una sintaxis incorrecta.
- Falta memoria en el sistema.
- Existe otro activador con el mismo nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- El archivo DAD tiene una sintaxis incorrecta.
- Falta memoria en el sistema.
- Existe otro activador con el mismo nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- Falta memoria en el sistema.
- No existe un activador con este nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- El archivo DAD tiene una sintaxis incorrecta.
- Falta memoria en el sistema.
- Existe otro activador con el mismo nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- El archivo DAD tiene una sintaxis incorrecta.
- Falta memoria en el sistema.
- Existe otro activador con el mismo nombre.

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 proveedor de servicios 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 activador interno. Causas posibles:

- El archivo DAD tiene una sintaxis incorrecta.
- Falta memoria en el sistema.
- Existe otro activador con el mismo nombre.

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 proveedor de servicios de software y proporcione el archivo de rastreo.

DXXA056E El valor de validación <*valor_validación*> **en el archivo DAD no es válido.**

Explicación: El elemento de validación contenido en el archivo DAD (definición de acceso a documento 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*> **de DAD no es válido.**

Explicación: El atributo de nombre de una tabla auxiliar contenido en el archivo DAD (definición de acceso a documento) 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 Un nombre de columna <*nombre_columna*> **en el archivo DAD no es válido.**

Explicación: El atributo de nombre de una columna contenido en el archivo DAD (definición de acceso a documento) 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*> **en el archivo DAD no es válido.**

Explicación: El atributo de tipo de una columna contenido en el archivo DAD (definición de acceso a documento) 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 <*vía_ubicación*> **de** <*nombre_columna*> **en el 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 <*varias_apariciones*> de <*nombre_columna*> en el archivo DAD no es válido.

Explicación: El atributo multi_occurrence de una columna contenido en el archivo DAD (definición de acceso a documento) 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 ha podido recuperar el número de columna de <*nombre_columna*> en la tabla <*nombre_tabla*>.

Explicación: 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 Habilitando colección <*nombre_colección*>. Por favor, espere.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXA064I Inhabilitando 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 <*procedimiento_almacenado*>.

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 XML Extender ha inhabilitado satisfactoriamente la colección <*nombre_colección*>.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA067I XML Extender ha habilitado satisfactoriamente la colección <*nombre_colección*>.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA068I XML Extender ha activado satisfactoriamente el rastreo.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna respuesta.

DXXA069I 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 XML Extender no pudo encontrar los archivos de vinculación. Vincule la base de datos antes de habilitarla.

Explicación: XML Extender ha intentado vincular automáticamente la base de datos antes de habilitarla, pero no ha encontrado los archivos de vinculación.

Respuesta del Usuario: Vincule la base de datos antes de habilitarla.

DXXA073E La base de datos no está vinculada. Por favor, vincule la base de datos antes de habilitarla.

Explicación: La base de datos no estaba vinculada cuando el usuario intentó habilitarla.

Respuesta del Usuario: Vincule 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 iSeries válido.

DXXA077E La clave de licencia no es válida. Para obtener más información, consulte las anotaciones cronológicas de errores.

Explicación: La licencia de software ha caducado o no existe.

Respuesta del Usuario: Póngase en contacto con el proveedor de servicios para obtener una licencia de software nueva.

DXXC000E No se puede abrir el archivo especificado.

Explicación: 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: XML Extender no ha podido 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: 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 grabar en el archivo especificado.

Explicación: XML Extender no puede grabar 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 Ubicador de LOB: rc=<*ubicador_rc*>.

Explicación: XML Extender no ha podido 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 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 del límite de LOB de DB2 UDB.

Explicación: El tamaño del archivo es mayor que el tamaño de XMLCLOB y XML Extender no puede importar todos los datos del archivo.

Respuesta del Usuario: Descomponga el archivo en objetos de menor tamaño 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: 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 grabar en el archivo <*nombre_archivo*>.

Explicación: Se ha producido un error al grabar en el archivo *nombre_archivo*.

Respuesta del Usuario: Compruebe que el sistema de archivos tenga suficiente espacio para el archivo.

DXXC011E No se puede grabar en el archivo de control de rastreo.

Explicación: XML Extender no puede grabar 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.

DXXC013E Los resultados de la UDF de extracción sobrepasan el límite de tamaño para el tipo de retorno de UDF.

Explicación: Los datos devueltos por una UDF de extracción deben ajustarse al límite de tamaño del tipo devuelto de la UDF, según la definición que se encuentra en la publicación DB2 UDB XML Extender Administración y programación. Por ejemplo, los resultados de extractVarchar no deben tener más de 4000 bytes (incluido el valor NULL de terminación).

Respuesta del Usuario: Utilice una UDF de extracción que tenga un límite de tamaño más grande para el tipo de retorno: 254 bytes para extractChar(), 4 KB para extractVarchar() y 2 GB para extractClob().

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 *La <vía_ubicación>* aparece varias veces.

Explicación: Una función de extracción escalar ha utilizado una vía de ubicación que aparece varias veces. Una función escalar sólo puede utilizar una vía de ubicación que no tenga apariciones múltiples.

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*> en la vía de acceso 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: XML Extender ha encontrado un error interno.

Respuesta del Usuario: Póngase en contacto con el proveedor de servicios 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: XML Extender no ha podido 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 un futuro release.

Respuesta del Usuario: Establezca este parámetro en NULL.

DXXG006E Error interno CLISTATE=<*estado_cli*>**, RC=**<*cli_rc*>**, creación** <*ID_creación*>**, archivo** <*nombre_archivo*>**, línea** <*número_línea*> **CLIMSG=**<*msj_CLI*>**.**

Explicación: XML Extender ha encontrado un error interno mientras utilizaba la CLI.

Respuesta del Usuario: Póngase en contacto con el proveedor de servicios 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 En entorno nacional <*entorno_nacional*> **no es consistente con la página de códigos de DB2 UDB** <*página_códigos*>**.**

Explicación: El entorno nacional del sistema operativo del servidor no es consistente con la página de códigos de DB2 UDB.

Respuesta del Usuario: Corrija el entorno local del sistema operativo del servidor y reinicie DB2.

DXXG008E En entorno nacional <*entorno_nacional*> **no está soportado.**

Explicación: No se puede encontrar el entorno local del sistema operativo del servidor en la tabla de la página de códigos.

Respuesta del Usuario: Corrija el entorno local del sistema operativo del servidor y reinicie DB2.

DXXG017E Se ha excedido el límite de *constante_XML_Extender* **en la creación** *ID_creación***, archivo** *nombre_archivo* **y línea** *número_línea***.**

Explicación: Consulte la publicación XML Extender Administration and Programming Guide para ver si la aplicación ha excedido un valor en la tabla de límites. Si no se ha excedido ningún límite, póngase en contacto con el proveedor de servicios de software. Cuando informe del error, incluya todos los mensajes de salida, registros de rastreo y la información sobre cómo reproducir el problema, como archivos DAD de entrada, documentos XML y definiciones de tabla.

Respuesta del Usuario: Corrija el entorno local del sistema operativo del servidor y reinicie DB2.

DXXM001W Se ha producido un error de DB2 UDB.

Explicación: DB2 ha encontrado el error especificado.

Respuesta del Usuario: Observe los mensajes adjuntos para obtener más explicaciones y consulte los mensajes de DB2 UDB y la documentación sobre los códigos del sistema operativo.

DXXQ000E Falta <*elemento*> **en el archivo DAD.**

Explicación: Falta un elemento obligatorio en el archivo DAD (definición de acceso a documento).

Respuesta del Usuario: Añada el elemento que falta 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 altera temporalmente 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 está quedando 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. Reduzca el tamaño de cada documento eliminando algunos nodos de elementos y atributos innecesarios del archivo DAD (definición de acceso a documento).

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 DAD (definición de acceso a documento), 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.

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 dicha clave en la tabla, la consulta debe generar una para la tabla utilizando una expresión de tabla y la función incorporada de DB2 UDB generate_unique().

DXXQ006E Un elemento attribute_node no tiene nombre.

Explicación: Un elemento attribute_node contenido en el archivo DAD (definición de acceso a documento) no tiene un atributo de nombre.

Respuesta del Usuario: Asegúrese de que cada attribute_node tenga un nombre en el archivo DAD.

DXXQ007E El attribute_node <*nombre_atributo*> no tiene elemento de columna o RDB_node.

Explicación: El elemento attribute_node contenido en el archivo DAD (definición de acceso a documento)no tiene un elemento de columna ni un RDB_node.

Respuesta del Usuario: Asegúrese de que cada attribute_node tenga un elemento de columna o RDB_node en el archivo DAD.

DXXQ008E Un elemento text_node no tiene ningún elemento de columna.

Explicación: Un elemento text_node contenido en el archivo DAD (definición de acceso a documento) no tiene un elemento de columna.

Respuesta del Usuario: Asegúrese de que cada text_node tenga un elemento de columna en el archivo DAD.

DXXQ009E La tabla de resultado <*nombre_tabla*> no existe.

Explicación: La tabla resultante especificada no se ha podido encontrar en el catálogo del sistema.

Respuesta del Usuario: Cree la tabla resultante antes de llamar al procedimiento almacenado.

DXXQ010E RDB_node de <*nombre_nodo*> no tiene una tabla en el archivo DAD.

Explicación: El RDB_node del attribute_node o del text_node debe tener una tabla.

Respuesta del Usuario: Especifique la tabla de RDB_node para attribute_node o text_node en el archivo DAD (definición de acceso a documento).

DXXQ011E El elemento RDB_node de <*nombre_nodo*> no tiene una columna en el archivo DAD.

Explicación: El RDB_node del attribute_node o del text_node debe tener una columna.

Respuesta del Usuario: Especifique la columna de RDB_node para attribute_node o text_node en el archivo DAD (definición de acceso a documento).

DXXQ012E Se han producido errores en la DAD.

Explicación: 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 DAD (definición de acceso a documento).

Respuesta del Usuario: Especifique el nombre del elemento de tabla o columna en el archivo DAD.

DXXQ014E Un elemento element_node no tiene nombre.

Explicación: Un elemento element_node contenido en el archivo DAD (definición de acceso a documento) no tiene un atributo de nombre.

Respuesta del Usuario: Asegúrese de que cada elemento element_node 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 RDB_node no está definido en el elemento superior del archivo DAD.

Explicación: Todas las tablas deben estar definidas en el RDB_node del elemento superior, en el archivo DAD (definición de acceso a documento). Las tablas de subelementos deben coincidir con las tablas definidas en el elemento superior. El nombre de tabla de este RDB_node no está en el elemento superior.

Respuesta del Usuario: Asegúrese de que la tabla del RDB_node esté definida en el elemento superior del archivo DAD.

DXXQ017E La columna en la tabla de resultados <*nombre_tabla*> **es demasiado pequeña.**

Explicación: XML Extender ha generado un documento XML que es demasiado grande para caber en la columna de la tabla resultante.

Respuesta del Usuario: Elimine la tabla resultante. Cree otra tabla resultante con una columna mayor. Ejecute otra vez el procedimiento almacenado.

DXXQ018E Falta la cláusula ORDER BY de 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 DAD (definición de acceso a documento).

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 (definición de acceso a documento) que correlaciona SQL con XML.

Respuesta del Usuario: Edite el archivo DAD. Añada las columnas de clave como subelementos 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 tener el 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 más larga 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 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 encuentra 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 puede encontrar 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 override (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 Identificador no válido empezando <*identificador*>

Explicación: La serie no es un identificador de SQL válido DB2 UDB.

Respuesta del Usuario: Corrija la serie en la DAD para ajustarse a las reglas de los identificadores de SQL de DB2 UDB.

DXXQ034E Elemento de condición no válido en el RDB_node de la 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 de XML Extender para obtener la sintaxis correcta de la condición de unión en una DAD.

DXXQ035E Condición de unión no válida en el primer RDB_node de la DAD: <*condición*>

Explicación: Los nombres de columna del elemento de condición del RDB_node superior deben estar calificados con el nombre de tabla si la DAD especifica múltiples tablas.

Respuesta del Usuario: Consulte la documentación de 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 el** <*elemento*> **con varias apariciones.**

Explicación: El nodo de elemento y los descendientes no tienen ninguna correlación con la base de datos, pero el valor multi_occurrence es igual a YES.

Respuesta del Usuario: Corrija la DAD estableciendo el valor multi_occurrence en NO o bien cree un RDB_node en uno de sus hijos.

DXXQ038E **La sentencia de SQL es demasiado larga: SQL_statement**

Explicación: La sentencia de SQL especificada en el elemento <*sent_SQL*> de la DAD excede el número permitido de bytes.

Respuesta del Usuario: Reduzca la longitud de la sentencia de SQL a un valor igual o menor que 32765 bytes para Windows y UNIX o 16380 bytes para OS/390 e iSeries.

DXXQ039E **Se han especificado demasiadas columnas para una tabla en el archivo DAD.**

Explicación: Se utiliza un archivo DAD para la descomposición o la composición de RDB puede tener un máximo de 100 elementos text_node y attribute_node que especifiquen columnas exclusivas dentro de la misma tabla.

Respuesta del Usuario: Reduzca el número total de elementos text_node y attribute_node que hacen referencia a columnas exclusivas dentro de la misma tabla a 100 o menos.

DXXQ040E **El nombre de elemento** <*nombre_elemento*> **en el archivo DAD no es válido.**

Explicación: El nombre de elemento especificado en el archivo DAD (definición de acceso a documento) es incorrecto.

Respuesta del Usuario: Asegúrese de que el nombre de elemento se haya escrito correctamente en el archivo DAD. Consulte la DTD del archivo DAD.

DXXQ041W **El documento XML se ha generado satisfactoriamente. Una o más de las vías de acceso de alteración temporal que se han especificado no es válida y se ha omitido.**

Explicación: Especifique solamente una vía de acceso de alteración temporal.

Respuesta del Usuario: Asegúrese de que el nombre de elemento se haya escrito correctamente en el archivo DAD. Consulte la DTD del archivo DAD.

DXXQ043E **Atributo** <*nombre_atributo*> **no encontrado bajo del elemento** <*nombre_elemento*>**.**

Explicación: El atributo <*nombre_atributo*> no estaba presente en el elemento <*nombre_elemento*> o o en uno de sus elementos hijos.

Respuesta del Usuario: Asegúrese de que el atributo aparezca en el documento XML en los lugares que el archivo DAD necesite.

DXXQ044E **El elemento** <*nombre_elemento*> **no tiene un elemento ancestro** <*ancestro*>**.**

Explicación: De acuerdo con la DAD, <*ancestro*> es un elemento ascendiente de <*nombre_elemento*> . En el documento XML, uno o más elementos <*nombre_elemento*> no tiene dicho ancestro.

Respuesta del Usuario: Asegúrese de que la anidación de elementos del documento XML se adapta a lo que se ha especificado en el archivo DAD correspondiente.

DXXQ045E **El subárbol bajo el elemento** <*nombre_elemento*> **contiene múltiples atributos con el nombre** <*nombre_atributo*>**.**

Explicación: Un subárbol debajo de <*nombre_elemento*> en el documento XML contiene varias instancias del atributo <*nombre_atributo*> que, de acuerdo con la DAD, se debe descomponer en la misma fila. Los elementos o atributos que se van a descomponer deben tener nombres exclusivos.

Respuesta del Usuario: Asegúrese de que el elemento o el atributo del subárbol tiene un nombre exclusivo.

DXXQ046W **No se ha encontrado el DTDID en la DAD.**

Explicación: En la DAD, VALIDATION se establece en 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 alguna validación, especifique el elemento DTDID en el archivo DAD.

DXXQ047E Error del analizador en la línea <mv>número_línea</mv> columna *número_col*: *msje*

Explicación: El analizador no ha podido analizar el documento debido al error notificado.

Respuesta del Usuario: Corrija el error del documento, consultando si es necesario, las especificaciones de XML.

DXXQ048E Error interno - consulte el archivo de rastreo.

Explicación: El procesador de la hoja de estilo ha devuelto un error interno. Es posible que el documento XML de la hoja de estilo no sea válido.

Respuesta del Usuario: Asegúrese de que el documento XML y la hoja de estilo sean válidos.

DXXQ049E El archivo de salida ya existe.

Explicación: El archivo de salida especificado ya existe en este directorio.

Respuesta del Usuario: Cambie la vía de salida o el nombre de archivo del documento de salida por un nombre que sea exclusivo o suprima el archivo existente.

DXXQ050E No es posible crear un nombre de archivo exclusivo.

Explicación: La UDF no ha podido crear un nombre de archivo exclusivo para el documento de salida en el directorio especificado porque no tiene acceso, se están utilizando todos los nombres de archivo que se pueden generar o bien es posible que el directorio no exista.

Respuesta del Usuario: Asegúrese de que la UDF dispone de acceso al directorio especificado; vaya a un directorio con nombres de archivo disponibles.

DXXQ051E No hay datos de entrada o de salida.

Explicación: Uno o más parámetros de entrada o de salida no tienen un valor válido.

Respuesta del Usuario: Compruebe la sentencia para ver si faltan parámetros necesarios.

DXXQ052E Se ha producido un error al acceder a la tabla DB2XML.XML_USAGE.

Explicación: O bien la base de datos no se ha habilitado o bien se ha eliminado la tabla DB2XML.XML_USAGE.

Respuesta del Usuario: Asegúrese de que la base de datos se haya habilitado y de que la tabla DB2XML.XML_USAGE sea accesible.

DXXQ053E Ha fallado una sentencia de SQL: *mje*

Explicación: Una sentencia de SQL generada durante el proceso de XML Extender no ha conseguido ejecutarse.

Respuesta del Usuario: Para obtener más información, examine el rastreo. Si no puede corregir la condición de error, póngase en contacto con el proveedor de servicios de software. Cuando informe del error, asegúrese de incluir todos los mensajes, el archivo de rastreo y la explicación que permita reproducir el error.

DXXQ054E Parámetro de entrada no válido: *parám*

Explicación: El parámetro de entrada especificado para un procedimiento almacenado o una UDF no es válido.

Respuesta del Usuario: Compruebe la signatura del procedimiento almacenado o la UDF relevantes y asegúrese de que el parámetro de entrada real es correcto.

DXXQ055E Error ICU: *error_u*

Explicación: Se ha encontrado un error ICU durante la operación de conversión.

Respuesta del Usuario: Informe del error al proveedor de servicio de software. Incluya el archivo de rastreo, y las instrucciones para reproducir el error.

DXXQ056E El elemento/atributo *nombre_xml* **no puede correlacionarse con la columna designada como parte de la clave foránea (columna** *columna* **en la tabla** *tabla***).**

Explicación: El elemento/atributo especificado no puede correlacionarse con una columna especificada como parte de una clave foránea. Los valores de datos para claves foráneas vienen determinados por los de las claves primarias; no es necesaria una correlación del elemento/atributo especificado en el documento xml con una tabla y una columna no es necesaria.

Respuesta del Usuario: Elimine la correlación de RDB_node con la columna y tabla especificada en la DAD.

DXXQ057E Los códigos schemabindings y dtdid no pueden coexistir en el archivo DAD.

Explicación: Los códigos schemabindings y dtdid no pueden coexistir en el archivo DAD.

Respuesta del Usuario: Compruebe que los códigos schemabindings o dtdid existen en el archivo DAD, pero no ambos a la vez.

DXXQ058E Falta el código nonamespacelocation dentro del código schemabindings en el archivo DAD.

Explicación: Falta el código nonamespacelocation dentro del código schemabindings en el archivo DAD.

Respuesta del Usuario: Añada el código nonamespacelocation al código schemabindings.

DXXQ059E No es posible encontrar el código doctype dentro del código XCollection en la DAD para la validación del esquema.

Explicación: El código doctype no puede encontrarse dentro del código XCollection en la DAD para la validación de esquema.

Respuesta del Usuario: Elimine el código doctype de dentro del código Xcollection para la validación del esquema.

DXXQ060E El intento de buscar el ID de SCHEMA *id_esquema* **ha fallado.**

Explicación: XML Extender no ha podido encontrar el ID de SCHEMA mientras intentaba habilitar la columna. EL ID de SCHEMA corresponde con el valor del atributo de ubicación del código nonamespacelocation que se encuentra dentro del código schemabindings en el archivo DAD.

Respuesta del Usuario: Compruebe que se ha especificado el valor correcto para el ID de SCHEMA en el archivo DAD.

DXXQ061E El formato de la serie no es válido.

Explicación: El formato de la representación de la serie no es válido. Si la serie es un valor de fecha, hora o indicación de la hora, la sintaxis no se ajusta a su tipo de datos.

Respuesta del Usuario: Compruebe el formato del valor de la fecha, hora o indicación de la hora se ajusta al formato adecuado para su tipo de datos.

DXXQ062E No quedan filas del conjunto de resultados para *tabla* **para producir un valor XML para** *elemento***.**

Explicación: Esta condición de error suele estar causada porque falta una especificación multi_occurrence = YES en el element_node padre del elemento o atributo proporcionado.

Respuesta del Usuario: compruebe en la DAD que el valor de multi_occurrence en el element_node padre refleja correctamente la multiplicidad de element_node hijos.

DXXQ063E El valor del atributo multi_occurrence en *nombre_elemento* **en el archivo DAD no es válido.**

Explicación: El valor del atributo multi_occurrence en el element_node especificado en el archivo DAD (definición de acceso a documento) es incorrecto o está ausente. El valor debe ser 'yes' o 'no', en mayúsculas o minúsculas.

Respuesta del Usuario: Asegúrese de haber especificado el atributo multi_occurrence correctamente en el archivo DAD.

DXXQ064E No se ha encontrado la columna *columna* **en la tabla foránea** *tabla***.**

Explicación: Una columna clave especificada en la condición de unión no se ha correlacionado con ningún nodo de elemento o atributo.

Respuesta del Usuario: Compruebe que la condición de unión especificada en el archivo DAD es correcta y que todas las columnas clave están correlacionadas con nodos de elementos o atributos.

DXXQ065I Todos los activadores relacionados con las columnas habilitadas para XML se han regenerado satisfactoriamente.

Explicación: Este es un mensaje informativo.

Respuesta del Usuario: No es necesaria ninguna acción.

DXXQ066E La clave primaria de la tabla *nombre_tabla* **no existe.**

Explicación: XML Extender no ha podido determinar la clave primaria de la tabla *nombre_tabla*. Compruebe que no se haya descartado la clave primaria de la tabla después de haber habilitado la columna para XML.

Respuesta del Usuario: Altere la tabla para añadir la clave primaria como ID RAÍZ cuando se habilitó la columna para XML.

DXXQ067E Ha fallado el intento de *acción***.**

Explicación: Mientras se intentaba *acción*, se ha producido un error de SQL.

Respuesta del Usuario: Póngase en contacto con el proveedor de servicios de software. Cuando informe del error, asegúrese de incluir el archivo de rastreo de XML Extender.

DXXQ068E No es posible establecer el SQLID actual a [idusuario]. SQLCODE = [código_sql].

Explicación: Se ha producido un error de SQL mientras se intentaba establecer el sqlid actual a un ID de autorización secundario.

Respuesta del Usuario: Compruebe que está especificando un Id de autorización secundario y que tiene autorización para dicho ID.

Apéndice A. Ejemplos

Este apéndice muestra los objetos a modo de ejemplo que se utilizan en los ejemplos de este manual.

Ejemplo de DTD XML

La siguiente DTD se utiliza para el documento `getstart.xml` al cual se hace referencia en esta guía.

```
<!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 15. DTD de XML de ejemplo: `getstart.dtd`

Ejemplo de documento XML: getstart.xml

El siguiente documento XML, `getstart.xml`, es el documento XML de ejemplo utilizado en todos los ejemplos de este manual. Contiene los códigos XML para formar una orden de compra.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "instalación_dxx/samples/db2xml/dtd/getstart.dtd">
    <Order key="1">
    <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>
  <Part color="black ">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
      </Part>
  <Part color="red   ">
    <key>128</key>
    <Quantity>28</Quantity>
    <ExtendedPrice>38000.00</ExtendedPrice>
    <Tax>7.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-12-30</ShipDate>
      <ShipMode>TRUCK </ShipMode>
    </Shipment>
      </Part>
</Order>
```

Figura 16. Documento XML de ejemplo: getstart.xml

Archivos de definición de acceso a documento

Las siguientes secciones contienen archivos DAD (definición de acceso a documentos) que correlacionan los datos XML con las tablas relacionales de DB2 UDB, utilizando las modalidades de acceso de columna XML o de colección XML.

- "Archivo DAD de ejemplo: Columna XML"
- "Archivo DAD de ejemplo: Colección XML: Correlación SQL" en la página 307 muestra un archivo DAD para una colección XML que hace uso de la correlación de SQL.
- "Archivo DAD de ejemplo: XML: RDB_node mapping" en la página 308 muestra un archivo DAD para una colección XML que utiliza la correlación de RDB_node.

Archivo DAD de ejemplo: Columna XML

Este archivo DAD contiene la correlación de una columna XML y define la tabla, las tablas auxiliares y columnas que contendrán los datos XML.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM  "instalación_dxx/samples/db2xml/dtd/dad.dtd">
<DAD>
   <dtdid>"instalación_dxx/samples/db2xml/dtd/getstart.dtd"
   </dtdid>
   <validation>YES</validation>

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

   </Xcolumn>

</DAD>
```

Figura 17. Archivo DAD de ejemplo para una columna XML: getstart_xcolumn.dad

Archivo DAD de ejemplo: Colección XML: Correlación SQL

Este archivo DAD contiene una sentencia de SQL que especifica las tablas, columnas y condiciones de DB2 UDB que contendrán los datos XML.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "instalación_dxx/samples/db2xml/dtd/dad.dtd"><DAD>
<validation>NO</validation>   <Xcollection>
<SQL_stmt>SELECT o.order_key, customer_name, customer_email, p.part_key, color,
   quantity, price, tax, ship_id, date, mode from order_tab o, part_tab p,
   table(select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab)  s
              p.price > 20000 and
              p.order_key = o.order_key and
              s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id</SQL_stmt>
     <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM
"instalación_dxx/samples/db2xml/dtd/getstart.dtd
"</doctype>
```

Figura 18. El archivo DAD de ejemplo para una colección XML utilizando la correlación de SQL: order_sql.dad (Parte 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_occurrence="YES">
        <element_node name="ShipDate">
        <text_node><column name="date"/></text_node>
           </element_node>
      <element_node name="ShipMode">
        <text_node><column name="mode"/></text_node>
           </element_node>
           </element_node>
           </element_node>
           </element_node>
</root_node> </Xcollection>
</DAD>
```

Figura 18. El archivo DAD de ejemplo para una colección XML utilizando la correlación de SQL: order_sql.dad (Parte 2 de 2)

Archivo DAD de ejemplo: XML: RDB_node mapping

Este archivo DAD utiliza elementos <RDB_node> para definir las tablas, columnas y condiciones de DB2 UDB que contendrán los datos XML.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "SQLLIB/samples/db2xml/dtd/dad.dtd>
<DAD>
 <dtdid>E:\dtd\lineItem.dtd</dtdid>
   <validation>YES</validation>
  <Xcollection>
     <prolog>?xml version="1.0"?</prolog>
 <doctype>!DOCTYPE Order SYSTEM
                   "SQLLIB/samples/db2xml/dtd/getstart.dtd"</doctype>
      <root_node>
         <element_node name="Order">
                    <RDB_node>
                    <table name="order_tab"/>
                    <table name="part_tab"/>
```

```
      <table name="ship_tab"/>
<condition>order_tab.order_key=part_tab.order_key AND
            part_tab.part_key=ship_tab.part_key </condition>
                  </RDB_node>
        <attribute_node name="Key">
                  <RDB_node>
                  <table name="order_tab"/>
               <column name="order_key"/>
                  </RDB_node>
  </attribute_node>
 <element_node name="Customer">

  <element_node name="Name">
 <text_node>
                  <RDB_node>
                  <table name="order_tab"/>
                <column name="customer_name"/>
                  </RDB_node>
    </text_node>
          </element_node>
   <element_node name="Email">
 <text_node>
                  <RDB_node>
                  <table name="order_tab"/>
                 <column name="customer_email"/>
                  </RDB_node>
    </text_node>
          </element_node>
          </element_node>
 <element_node name="Part">
        <attribute_node name="Key">
                  <RDB_node>
                  <table name="part_tab"/>
                  <column name="part_key"/>
                  </RDB_node>
  </attribute_node>
   <element_node name="ExtendedPrice">
 <text_node>
                  <RDB_node>
                  <table name="part_tab"/>
                     <column name="price"/>
                    <condition>price > 2500.00</condition>
                  </RDB_node>
    </text_node>
          </element_node>
  <element_node name="Tax">
 <text_node>
                  <RDB_node>
                  <table name="part_tab"/>
       <column name="tax"/>
                  </RDB_node>
    </text_node>
          </element_node>

   <element_node name="Quantity">
 <text_node>
                  <RDB_node>
                  <table name="part_tab"/>
                     <column name="qty"/>
                  </RDB_node>
    </text_node>
          </element_node>
  <element_node name="Shipment" multi_occurrence="YES">
     <element_node name="ShipDate">
 <text_node>
                  <RDB_node>
     <table name="ship_tab"/>
```

```
                        <column name="date"/>
                        <condition>date > '1966-01-01'</condition>
                    </RDB_node>
      </text_node>
            </element_node>
      <element_node name="ShipMode">
   <text_node>
                    <RDB_node>
        <table name="ship_tab"/>
                        <column name="mode"/>
                    </RDB_node>
      </text_node>
            </element_node>
              <element_node name="Comment">
   <text_node>
                    <RDB_node>
        <table name="ship_tab"/>
                       <column name="comment"/>
                    </RDB_node>
      </text_node>
            </element_node>
       </element_node> <!-- end of element Shipment-->
   </element_node> <!-- end of element Part -->
    </element_node> <!-- end of element Order -->
</root_node>
</Xcollection>

</DAD>
```

Apéndice B. Consideraciones sobre la página de códigos

Los documentos XML y otros archivos asociados deben estar codificados correctamente para cada cliente o servidor que acceda a los archivos. XML Extender da por supuesto que cuando se procesa un archivo; es necesario que el usuario conozca cómo XML Extender maneja las conversiones de páginas de códigos. Las consideraciones principales son las siguientes:

- Comprobar que la página de códigos real del cliente que recupera un documento XML desde DB2 UDB coincide con la codificación del documento.
- Comprobar que, cuando el documento lo procesa un analizador sintáctico XML, la declaración de codificación del documento XML también es coherente con el código real del documento.

Las siguientes secciones describen las cuestiones relacionadas con estas consideraciones, cómo prepararse para posibles problemas y cómo XML Extender y DB2 UDB dan soporte a las páginas de códigos al pasar los documentos desde el cliente al servidor y a la base de datos.

Terminología de las páginas de códigos XML

En esta sección se utilizan los siguientes términos acerca de las páginas de códigos XML:

codificación del documento
Es la página de códigos de un documento XML.

declaración de codificación del documento
Nombre de la página de códigos especificada en la declaración XML. Por ejemplo, la siguiente declaración de código especifica `ibm-1047`:

```
<?xml version="1.0" encoding="ibm-1047"?>
```

documento coherente
Es un documento en el que la página de códigos coincide con la declaración de código.

documento incoherente
Es un documento en el que la página de códigos no coincide con la declaración de código.

variable (de entorno) de registro DB2CODEPAGE
Especifica la página de códigos de los datos presentados a DB2 UDB desde una aplicación cliente de base de datos. DB2 UDB obtiene la página de códigos del cliente del entorno local sistema operativo, a no ser 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
Es la 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
En entorno local del sistema operativo en el que DB2 UDB se encuentra instalado.

página de códigos de la base de datos
Son los códigos los datos almacenados, que se determinan durante la creación de la base de datos. Si no se especifica explícitamente utilizando la cláusula USING CODESET, este valor es igual al entorno local de sistema operativo del servidor.

Premisas sobre la página de códigos de DB2 y XML

Cuando DB2 UDB envía o recibe 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 distintas, DB2 UDB 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 el fragmento de documento, a una tabla de base de datos.
- La base de datos, cuando se descompone un documento en una o más tablas de base de datos.
- El cliente, cuando se exporta el documento desde 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.

Premisas para importar un documento XML

Cuando se importa un documento XML en la base de datos, se importa generalmente como un documento XML para almacenarlo en una columna XML o bien para descomponerlo para una colección XML, donde el contenido del elemento y atributo se guardarán como datos DB2 UDB. Cuando se importa un documento DB2 UDB convierte la codificación del documento a la de la base de datos. DB2 UDB asume que el documento está en la página de códigos especificada en la columna "Página de código inicial" en la siguiente tabla. La Tabla 93 resume las conversiones que DB2 UDB realiza al importar un documento XML.

Tabla 93. Utilización de las UDF y los procedimientos almacenados cuando el archivo XML se importa a la base de datos.

Tarea	Página de códigos inicial de la conversión	Página de códigos final de 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 una columna o colección mediante procedimientos almacenados o utilizar mandatos de administración para importar archivos DAD	Página de códigos del cliente (la página de códigos utilizada para vincular DXXADMIN durante la instalación),	Página de códigos de base de datos	

Tabla 93. Utilización de las UDF y los procedimientos almacenados cuando el archivo XML se importa a la base de datos. (continuación)

Tarea	Página de códigos inicial de la conversión	Página de códigos final de la conversión	Comentarios
Utilizar funciones definidas por el usuario: • XMLVarchar FromFile() • XMLCLOB FromFile() • Content(): recuperación de XMLFILE y almacenamiento en CLOB	Página de códigos del servidor	Página de códigos de base de datos	La página de códigos de la base de datos se convierte en la página de códigos del cliente cuando los datos se entregan al cliente
Utilizar procedimientos almacenados para la descomposición	Página de códigos del cliente	Página de códigos de base de datos	• Se supone que el documento que se va a descomponer es la página de códigos del cliente. Los datos resultantes de la descomposición se almacenan en tablas utilizando la página de códigos de la base de datos.

Premisas para exportar un documento XML

Cuando se exporta un documento XML desde la base de datos, se exporta de acuerdo con una petición de un cliente y se entrega uno de estos objetos:

- Un documento XML procedente de una columna XML
- Los resultados de una consulta de documentos XML presentados en una columna XML
- Un documento XML compuesto resultante de una colección XML

Cuando se exporta un documento, DB2 UDB convierte la codificación del documento a la del cliente o servidor, dependiendo de dónde se haya originado la petición y dónde se vayan a presentar los datos. La Tabla 94 resume las conversiones que DB2 UDB realiza al exportar un documento XML.

Tabla 94. Utilización de las UDF y los procedimientos almacenados cuando el archivo XML se exporta desde la base de datos

Tarea	Conversión realizada por DB2	Comentarios
Utilizar funciones definidas por el usuario: • XMLFileFromVarchar() • XMLFileFromCLOB() • Content(): recuperación de XMLVARCHAR y almacenamiento en un archivo de servidor externo	De la página de códigos de la base de datos en la página de códigos del servidor	

Tabla 94. Utilización de las UDF y los procedimientos almacenados cuando el archivo XML se exporta desde la base de datos (continuación)

Tarea	Conversión realizada por DB2	Comentarios
Componer documentos XML mediante un procedimiento almacenado y guardarlos en una tabla resultante, que se puede consultar y exportar.	Convertir de la página de códigos de la base de datos al la página de códigos del cliente cuando el conjunto de resultados se presenta al cliente	• Cuando se componen documentos, XML Extender copia la declaración de codificación especificada por el código de la DAD al documento recién creado. La declaración de código debe coincidir con la página de códigos del cliente en el momento de su presentación.

Consideraciones de la declaración de codificación para XML Extender

La *declaración de codificación* especifica la página de códigos del documento XML y aparece en la sentencia de declaración de XML. Cuando se utiliza XML Extender, es importante asegurarse de que la codificación del documento coincide con la página de códigos del cliente o del servidor, dependiendo de dónde se encuentre el archivo.

Declaraciones de codificación permitidas

Puede utilizar cualquier declaración de codificación en documentos XML, conforme a algunas directrices. En esta sección se definen estas directrices junto con las declaraciones de codificación soportadas.

Los códigos transferibles recomendados para datos XML son UTF-8 y UTF-16, de acuerdo con la especificación XML. La utilización de estos códigos permite que la aplicación pueda trabajar conjuntamente con entornos diferentes. Si utiliza los códigos mostrados en la Tabla 95, podrá transferir la aplicación entre sistemas operativos de IBM. Si utiliza otros códigos, es menos probable que los datos se puedan transferir.

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.
- **Categoría** muestra el sistema operativo en el que DB2 UDB 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 95. Declaraciones de codificación soportadas por XML Extender

Categoría	Codificación	Página de códigos
Unicode	UTF-8	1208
	UTF-16	1200

Tabla 95. Declaraciones de codificación soportadas por XML Extender (continuación)

Categoría	Codificación	Página de códigos
ASCII	iso-8859-1	819
	ibm-1252	1252
	iso-8859-2	912
	iso-8859-5	915
	iso-8859-6	1089
	iso-8859-7	813
	iso-8859-8	916
	iso-8859-9	920
MBCS	gb2312	1386
	ibm-932, shift_jis78	932
	Shift_JIS	943
	IBM-eucCN	1383
	ibm-1388	1388
	IBM-eucJP, EUC-JP	954, 33722
	ibm-930	930
	ibm-939	939
	ibm-1390	1390
	ibm-1399	1399
	ibm-5026	5026
	ibm-5035	5035
	euc-tw, IBM-eucTW	964
	ibm-937	937
	euc-kr, IBM-eucKR	970
	big5	950

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 la sección "Declaraciones de codificación y codificaciones coherentes" para conocer las consecuencias de codificaciones incompatibles. Consulte la siguiente dirección Web para obtener una lista de páginas de código soportadas por el analizador XML utilizado 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 incluye una declaración de codificación que no es la que se indica en la declaración.

La Figura 19 muestra que los clientes tienen páginas de códigos coherentes con el código del documento y el código declarado.

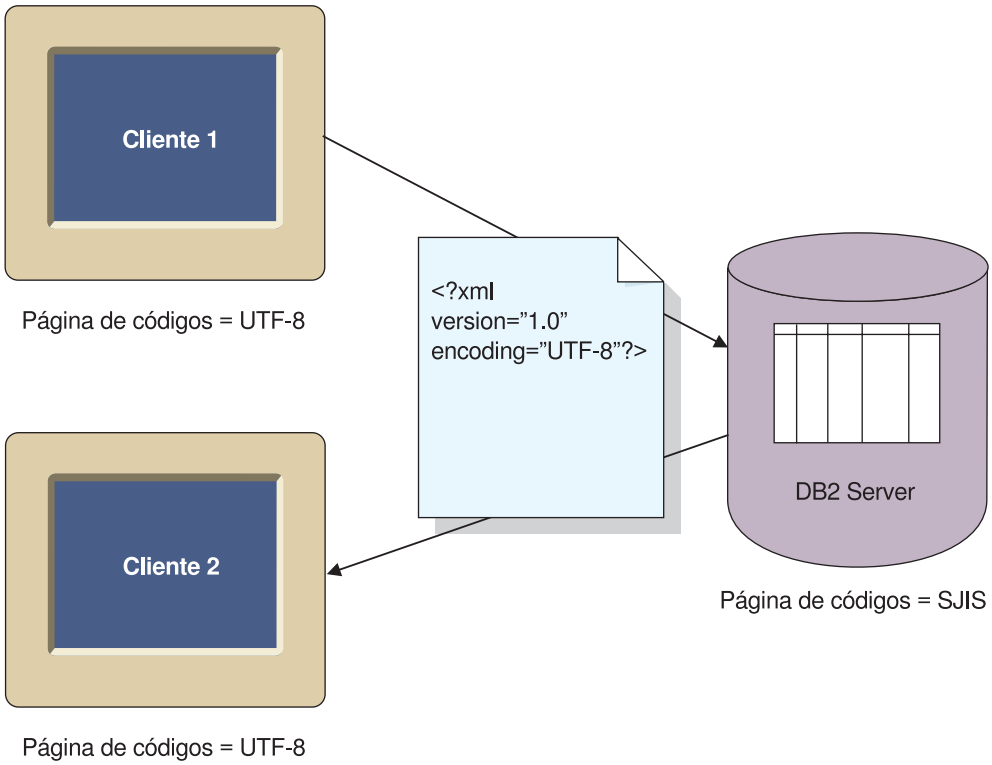


Figura 19. Clientes con páginas de códigos que coinciden

Las consecuencias de tener páginas de códigos diferentes son las siguientes situaciones posibles:

- Una conversión con una posible pérdida de datos, especialmente si la página de códigos original es Unicode y la página de códigos del sistema destino no es Unicode. Unicode contiene el juego completo de caracteres. Si un archivo se convierte desde UTF-8 a una página de códigos que no da soporte a todos los caracteres utilizados en el documento, se pueden 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 en la página 317 muestra un entorno en el que las páginas de códigos de los clientes no son coherentes.

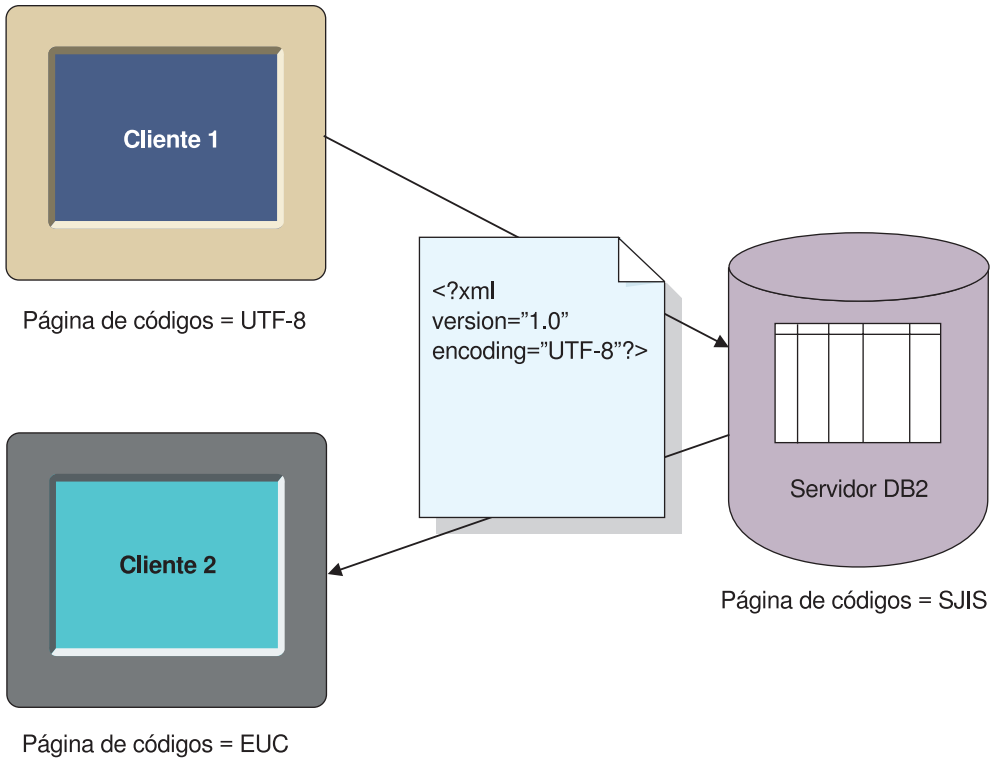


Figura 20. Clientes con páginas de códigos que no coinciden

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 código:

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" ?>
```

Casos de 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 se transfieren 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 casos que se describen a continuación.

Los siguientes casos describen situaciones de conversiones comunes que se pueden producir:

Caso 1: Este caso presenta una configuración con codificaciones coherentes, sin conversión por DB2 UDB y un documento importado desde el servidor. La declaración de codificación del documento es UTF-8, la del servidor es UTF-8 y la de la base de datos es UTF-8. DB2 UDB no necesita convertir el documento porque la página de códigos del servidor y la página de códigos de la base de datos son idénticas. La codificación y la declaración son coherentes.

1. El documento se importa en DB2 UDB utilizando la UDF XMLClobfromFile.
2. El documento se extrae en el servidor.

Caso 2: Este caso presenta una configuración con codificaciones coherentes, conversión de DB2 UDB y un documento importado desde el servidor y exportado al cliente. La codificación y declaración de código del documento es SJIS, las páginas de códigos del cliente y del servidor son SJIS y las páginas de códigos de la base de datos son UTF-8.

1. El documento se importa en DB2 UDB utilizando la UDF XMLClobfromfile del servidor. DB2 convierte el documento desde el código SJIS y lo guarda en código UTF-8. La declaración y la codificación no son coherentes en la base de datos.
2. Un cliente que utiliza SJIS solicita un documento para su presentación en el navegador Web. DB2 UDB convierte el documento a SJIS, la página de códigos del cliente. Ahora, el código y la declaración de código del documento son coherentes en el cliente.

Caso 3: Este caso presenta una configuración con codificaciones incoherentes, conversión de DB2 UDB y un documento importado desde el servidor y exportado a un cliente. La declaración de codificación del documento es SJIS para el documento de entrada. La página de códigos del servidor es SJISibm-1047 y el cliente y la base de datos del cliente utilizan UTF-8.

1. El documento se importa a la base de datos utilizando una UDF de almacenamiento. DB2 UDB convierte el documento a UTF-8 desde SJIS. El código y la declaración no son coherentes.
2. Un cliente cuya página de códigos es UTF-8 solicita un documento para su presentación en un navegador Web. DB2 no realiza la conversión porque las páginas de códigos del cliente y de la base de datos son las mismas. El código y la declaración de código del documento no son coherentes, pues la declaración es SJIS y el código es UTF-8. Un analizador XML u otras herramientas de proceso de XML no pueden procesar el documento.

Caso 4: Este caso presenta una configuración con pérdida de datos, conversión de DB2 UDB y un documento importado desde un servidor UTF-8. La declaración de código del documento es UTF-8, el servidor utiliza UTF-8 y la base de datos utiliza SJIS.

El documento se importa en DB2 UDB utilizando la UDF XMLClobfromFile. DB2 UDB 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.

Caso 5: Este caso presenta una configuración en la que existe una limitación de Windows NT. En Windows NT, los entornos locales del sistema operativo no pueden establecerse en UTF-8, no obstante, DB2 UDB permite que el cliente

establezca la página de códigos a UTF-8 mediante `db2set DB2CODEPAGE=1208`. En este caso, 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; su 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 UDB utilizando la UDF Content() que devuelve un archivo XML. DB2 UDB convierte el documento desde UTF-8 a 1252, incluso aunque el cliente espera 1208 porque el cliente se encuentra en el mismo sistema que el servidor y tiene el valor de 1208.

Recomendaciones para evitar documentos XML no coherentes

En las secciones 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 incoherentes pueden causar la pérdida de 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:

- Al exportar un documento desde la base de datos utilizando las UDF de XML Extender, pruebe uno de los siguientes métodos (asumiendo que XML Extender ha exportado el archivo, en la página de códigos del servidor, al sistema de archivos en el 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 de la base de datos utilizando los procedimientos almacenados de XML Extender, pruebe uno de los métodos siguientes (se supone que el cliente consulta la tabla resultante, donde se guarda 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 un cliente de Windows NT: En Windows NT, el entorno local de un sistema operativo no se puede establecer en 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 de DB2XML.DTD_REF
- Habilite una columna o una colección
- Descomponga procedimientos almacenados

- Cuando se utilizan las UDF Content() o XMLFromFile para importar documentos XML, los documentos deben estar codificados en la página de códigos del entorno local del sistema operativo del servidor, que no puede ser UTF-8.
- Al exportar el archivo XML de la base de datos, defina la página de códigos del cliente con el siguiente mandato para hacer que DB2 UDB codifique los datos resultantes como 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 C. Límites de XML Extender

Esta sección describe los límites existentes para:

- Objetos de XML Extender
- Valores devueltos por funciones definidas por el usuario
- Parámetros de procedimientos almacenados
- Columnas de tablas de soporte de administración
- Composición y descomposición

La siguiente tabla describe los límites de los objetos de XML Extender.

Tabla 96. Límites para objetos de XML Extender

Objeto	Límite
Número máximo de filas en una tabla de una colección XML de descomposición	10240 filas de cada documento XML descompuesto
Número máximo de bytes en el nombre de vía de acceso de archivo XML especificado como valor de un parámetro	512 bytes
Longitud del elemento sql_stmt en un archivo DAD para la composición de SQL	Sistemas operativos Windows y UNIX: 32,765 bytes. Sistemas operativos OS/390 e iSeries: 16,380 bytes.
Número máximo de columnas para una tabla, especificadas para una tabla en el archivo DAD para la descomposición RDB_node	Se especifican 500 columnas (columnas para una tabla) en los elementos text_node y attribute_node en un archivo DAD.

La tabla siguiente describe los límites de los valores devueltos por las funciones definidas por el usuario de XML Extender.

Tabla 97. Límites de los valores de las funciones definidas por el usuario

Valores devueltos por funciones definidas por el usuario	Límite
Número máximo de bytes devueltos por una UDF extractCHAR	254 bytes
Número máximo de bytes devueltos por una UDF extractCLOB	2 gigabytes
Número máximo de bytes devueltos por una UDF extractVARCHAR	4 kilobytes

La tabla siguiente describe los límites para los parámetros de los procedimientos almacenados de XML Extender.

Tabla 98. Límites de los parámetros de procedimientos almacenados

Parámetros del procedimiento almacenado	Límite
Tamaño máximo de un CLOB[1] de un documento XML	1 MB

Tabla 98. Límites de los parámetros de procedimientos almacenados (continuación)

Parámetros del procedimiento almacenado	Límite
Tamaño máximo de un CLOB[1] de DAD (Definición de acceso a documento)	100 KB
Tamaño máximo de *collectionName*	30 bytes
Tamaño máximo de *colName*	30 bytes
Tamaño máximo de *dbName*	8 bytes
Tamaño máximo de *defaultView*	128 bytes
Tamaño máximo de *rootID*	30 bytes
Tamaño máximo de *resultTabName*	18 bytes
Tamaño máximo de *tablespace*	8 bytes
Tamaño máximo de *tbName*[2]	18 bytes
Tamaño máximo de *resultColumn*	30 bytes
Tamaño máximo de *validColumn*	30 bytes
Tamaño máximo de *varchar_value*	32672 bytes

Notas:

1. Este parámetro se puede modificar para dxxGenXMLClob y dxxRetrieveXMLCLOB.
2. Si el valor del parámetro *tbName* está calificado por un nombre de esquema, el nombre completo (incluido el carácter separador) no debe tener una longitud mayor que 128 bytes.

La tabla siguiente describe los límites de la tabla DB2XML.DTD_REF.

Tabla 99. Límites de XML Extender

Columnas de la tabla DB2XML.DTD_REF	Límite
Tamaño de la columna AUTHOR	128 bytes
Tamaño de la columna CREATOR	128 bytes
Tamaño de la columna UPDATOR	128 bytes
Tamaño de la columna DTDID	128 bytes
Tamaño de la columna CLOB	100 KB

Los nombres pueden aumentar su longitud cuando DB2 UDB los convierte de la página de códigos cliente a la página de códigos de la base de datos. Un nombre puede ajustarse al límite de tamaño establecido en el cliente, pero puede exceder el límite cuando el procedimiento almacenado obtiene el nombre convertido.

La tabla siguiente describe los límites para la composición y descomposición.

Tabla 100. Límites para la composición y descomposición de XML Extender

Objeto	Límite
Número máximo de filas insertadas en una tabla en una colección XML de descomposición	10240 filas de cada documento XML descompuesto
Longitud máxima del nombre de atributo en elements_node o attribute_node dentro de una DAD	63 bytes

Tabla 100. Límites para la composición y descomposición de XML Extender (continuación)

Objeto	Límite
Número máximo de bytes en el nombre de vía de acceso XMLFile especificado como valor de parámetro	512 bytes

Variable de entorno DB2DXX_MIN_TMPFILE_SIZE:

XML Extender coloca los documentos de gran tamaño en archivos temporales para evitar utilizar demasiada memoria durante el proceso. En aquellos sistemas con gran cantidad de memoria física, es posible evitar este movimiento de documentos a archivos temporales, reduciendo así la cantidad de actividad de entrada/salida. La variable de entorno DB2DXX_MIN_TMPFILE_SIZE instruye a XML Extender para que utilice el almacenamiento intermedio de la memoria, en lugar de archivos temporales para el proceso de documentos de un tamaño menor que el valor especificado. La variable sólo se aplica cuando se está en el servidor, no en un cliente. Si varios nodos físicos participan en una partición de varios nodos, es posible establecer la variable individualmente para cada nodo, reflejando de forma precisa la cantidad de memoria en cada sistema. Si no se ha establecido la variable de entorno, los documentos de más de 128KB se colocarán de forma automática en archivos temporales durante el proceso, mientras que aquellos documentos de menos de 128K se procesarán en la memoria.

Apéndice D. Nombres de UDT y UDF para XML Extender

El nombre completo de una función de DB2® es *nombre-esquema.nombre-función*, donde *nombre-esquema* es un identificador que proporciona una agrupación lógica para un conjunto de objetos de SQL. El nombre de esquema para las UDF y los UDT de XML Extender es DB2XML. En la documentación, sólo se efectúan referencias al nombre de función.

Puede especificar los UDT y las UDF sin el nombre de esquema si añade el nombre de esquema a la vía de función. La vía de función es una lista ordenada de nombres de esquema. DB2 UDB utiliza el orden de los nombres de esquemas en la 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.

Recomendación: Añada el nombre de esquema DB2XML a la vía de función. Añadiendo este nombre de esquema, puede escribir los nombres de las UDF y los UDT de XML Extender sin tener que calificarlos con DB2XML.El ejemplo siguiente muestra cómo añadir el esquema DB2XML a la vía de función:

```
SET CURRENT FUNCTION PATH = DB2XML, CURRENT FUNCTION PATH
```

Restricción: No añada DB2XML como primer esquema en la vía de función si inicia la sesión con un ID de usuario de DB2XML. DB2XML se establece automáticamente como el primer esquema cuando se conecta como DB2XML. Si añade DB2XML como el primer esquema de la vía de función, recibirá una condición de error porque la vía de función se iniciará con dos esquemas DB2XML.

Glosario de XML Extender

activador. Mecanismo que añade automáticamente, a una *tabla de anotaciones*, información sobre documentos que se deben indexar, cada vez que se añade, cambia o suprime un documento en una columna de texto.

activador. Definición de un conjunto de acciones que se deben emprender cuando se modifica una tabla. Los activadores se pueden utilizar para realizar acciones tales como validar datos de entrada, generar automáticamente un valor para una fila recién insertada, leer en otras tablas con el fin de establecer correspondencias o grabar en otras tablas con fines de auditoría. A menudo los activadores se utilizan para comprobar la integridad de los datos o para aplicar normas de negocio.

administración. Tarea consistente en preparar documentos de texto para su búsqueda, mantener índices y obtener información de estado.

ampliar. Acción de añadir términos adicionales, procedentes de un diccionario, a un término de búsqueda.

analizar. Calcular valores numéricos para las características de una imagen y añadir los valores a un catálogo QBIC.

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 (application programming interface). Véase *interfaz de programación de aplicaciones*.

archivo externo. Documento de texto en forma de archivo almacenado en el sistema de archivos del sistema operativo, en lugar de como celda de una tabla bajo el control de DB2. Archivo que existe en un sistema de archivos externo a DB2.

argumento de búsqueda. Condiciones que se especifican al hacer una búsqueda, que consisten en uno o varios términos de búsqueda, y parámetros de búsqueda.

atributo. Véase *atributo XML*.

atributo XML. Cualquier atributo especificado por la lista de atributos (ATTLIST) debajo del elemento XML en la DTD. XML Extender utiliza la vía de ubicación para identificar un atributo.

attribute_node (nodo de atributo). Representación de un atributo de un elemento.

búsqueda booleana. Búsqueda en la que se combinan uno o más términos de búsqueda utilizando operadores booleanos.

búsqueda de palabra completa. Búsqueda en documentos coreanos que respeta los límites de las palabras.

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.

carácter comodín. Véase *carácter de máscara*.

carácter de escape. Carácter que indica que el carácter que sigue a continuación no se debe interpretar como un *carácter de máscara*.

catálogo de planos. Tabla o archivo de base de datos que sirve para almacenar datos acerca de planos de imagen; por ejemplo, el número del fotograma inicial y final de un plano en un segmento de vídeo. El usuario puede acceder a una vista de la tabla mediante una consulta SQL, o puede acceder a los datos del archivo.

catálogo QBIC. Depósito que contiene datos acerca de las características visuales de imágenes.

CCSID. Coded Character Set Identifier (identificador de juego de caracteres codificados).

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 de XML Extender.

componer. Generar documentos XML a partir de datos relacionales de una colección XML.

condition (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.

Correlación de RDB_node. Ubicación del contenido de un elemento XML o del valor de un atributo XML, que están definidos por el RDB_node. XML Extender utiliza esta correlación para determinar dónde almacenar o recuperar los datos XML.

Correlación de 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. XML Extender utiliza esta definición para determinar dónde almacenar o recuperar los datos XML. La correlación de SQL se define mediante el elemento SQL_stmt en el archivo DAD.

corriente de datos. Información devuelta por una función de API, que comprende texto (un párrafo como mínimo) que incluye el término buscado, e información para resaltar el término encontrado en ese texto.

DAD. Véase *Definición de acceso a documento*.

datos de columna. Los datos almacenados dentro de una columna de DB2 UDB. 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).

DBCS. Double-byte character support (soporte 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, que tiene un formato 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. Una tabla de DB2 UDB, llamada DTD_REF, donde cada una de las filas de la tabla representa una DTD con información de metadatos adicionales.

descomponer. Separar documentos XML para obtener el conjunto de tablas relacionales de una colección XML.

documento. Véase *documento de texto*.

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). (1) . (2) Véase *Definición de tipo de documento*.

EDI. Electronic Data Interchange (intercambio electrónico de datos).

element_node (nodo de elemento). Representación de un elemento. Un nodo de elemento puede ser el elemento raíz o un elemento hijo.

element_node superior. Representación del elemento raíz del documento XML en el archivo DAD.

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. XML Extender utiliza la vía de ubicación para identificar un elemento.

espacio de tabla. Concepto abstracto consistente en una colección de contenedores donde se almacenan objetos de base de datos. El espacio de tabla proporciona un nivel de dirección interna entre una base de datos y las tablas contenidas en la base de datos. Un espacio de tabla:

- 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 tabla. 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 tabla, o se pueden desglosar individualmente en espacio de tabla separados.

esquema. Colección de objetos de base de datos, tales como tablas, vistas, índices o activadores. 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. XML Extender proporciona dos tipos de esquemas de correlación: la *correlación de SQL* y la *correlación de base de datos relacional (correlación de RDB_node)*.

examinador. Una función de Net Search Extender que permite visualizar texto en un monitor. Consulte *navegador Web*.

examinar. Ver texto visualizado en un monitor.

expresión de vía. Véase *vía de ubicación*.

Extensible 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.

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.

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. Véase *función de acceso*.

función de acceso. Una función definida por el usuario que convierte los tipos de datos de texto almacenado en una columna a un tipo de datos que pueda ser procesado por Net Search Extender.

función de conversión de datos. Función que se utiliza para convertir un tipo 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 definida por el usuario (user-defined function, UDF). Función de SQL creada por un usuario de DB2, en oposición a una función de SQL proporcionada por DB2. Net Search Extender proporciona funciones de búsqueda, como CONTAINS, en forma de UDF.

función definida por el usuario (user-defined function, UDF). Función definida por un usuario en DB2. Una vez definida, la función se puede utilizar en consultas SQL y objetos de vídeo. Por ejemplo, se pueden crear las UDF para obtener el formato de compresión de un vídeo o para devolver la frecuencia de muestreo de un segmento de audio. Esto proporciona una forma de definir el comportamiento de objetos de un tipo de determinado.

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 por omisión 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.

gigabyte (GB). Mil millones (10^9) de bytes. Cuando hace referencia a la capacidad de memoria, cantidad equivalente a 1.073.741.824 bytes.

gran objeto binario (binary large object, BLOB). Serie binaria cuya longitud puede ser de hasta 2 GB. Los objetos de imagen, de audio y de vídeo se almacenan en una base de datos DB2 en forma de grandes objetos binarios.

gran objeto de caracteres (character large object, CLOB). Serie 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 sólo byte se almacenan en una base de datos DB2 UDB como CLOB.

gran objeto de caracteres de doble byte (double-byte character large object, DBCLOB). Serie 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 incluyen caracteres de doble byte se almacenan en una base de datos DB2 UDB como 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)* o bien *gran objeto de caracteres de doble byte (DBCLOB)*.

habilitar. Para preparar una base de datos, una tabla de texto u una columna de texto para que pueda ser utilizada por XML Extender.

ID raíz. Identificador exclusivo que asocia todas las tablas auxiliares con la tabla de aplicación.

imagen. Representación electrónica de una foto.

indexación de la estructura arborescente binaria. El esquema de indexación nativo proporcionado por el motor de DB2 UDB. Crea entradas de índice en la estructura de estructura de árbol binaria. Da soporte a los tipos de datos base de DB2.

indexar, índice. Extraer términos significativos de un texto y almacenarlos en un *índice de texto*. 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 de vídeo. Archivo que el Video Extender utiliza para encontrar un determinado *plano* o fotograma en un segmento de vídeo.

índice estructural de texto. Para indexar claves de texto basadas en la estructura de árbol del documento XML, utilizando DB2 UDB Net Search Extender.

índice lingüístico. *Índice de texto* que contiene términos que se han reducido a su forma base mediante procesos lingüísticos. Por ejemplo, “ratoncillo”, estaría indexado como “ratón”. Véase también *índice exacto*, *índice Ngram* e *índice dual*.

inhabilitar. Para restaurar una base de datos, una tabla de texto o una columna de texto al estado que tenía antes de ser habilitada para XML Extender eliminando los elementos creados durante el proceso de habilitación.

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) Los extensores de DB2 UDB proporcionan las API para solicitar funciones definidas por el usuario, operaciones administrativas, operaciones de visualización y detección de cambios. DB2 Net Search Extender proporciona las API para solicitar funciones definidas por el usuario operaciones administrativas y servicios de recuperación de información. En DB2, una función 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.

kilobyte (KB). Un millar (10^3) de bytes. Cuando hace referencia a la capacidad de memoria, cantidad equivalente a 1024 bytes.

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 de LOB. Un valor pequeño (4 bytes) almacenado en una variable de sistema principal que puede utilizarse en un programa para referirse a un LOB mucho más grande contenido en una base de datos DB2 UDB. Mediante un localizador de LOB, el usuario puede manejar el LOB como si estuviera almacenado en una variable normal de lenguaje principal, sin necesidad de mover el LOB entre la aplicación, de la máquina cliente, y el servidor de bases de datos.

localizador uniforme de recursos (uniform resource locator, URL). Una dirección que nombra un servidor HTTP y opcionalmente un nombre de directorio y archivo, por ejemplo: http://www.ibm.com/software /data/db2/extenders.

megabyte (MB). Un millón (10^6) de bytes. Cuando hace referencia a la capacidad de memoria, cantidad equivalente a 1.048.576 bytes.

método de acceso y almacenamiento. Asocia los documentos XML a una base de datos DB2 UDB 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 de árbol que se utiliza 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 de base de datos relacional (RDB_node). 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 lógico. Nodo de un procesador cuando ese procesador tiene asignados varios nodos.

objeto. En la programación orientada a objetos, concepto abstracto que consta de datos y las operaciones asociadas a esos datos.

objeto de consulta. Objeto que especifica las características, valores y factores de ponderación de una consulta QBIC. Se puede dar un nombre al objeto y guardarlo para su uso futuro en una consulta QBIC. Compárese con serie de consulta.

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*.

página de códigos. Asignación de caracteres gráficos y funciones de control a todos los elementos de un código. Por ejemplo, la asignación de caracteres y funciones a los 256 elementos de un código de 8 bits.

partición de base de datos. Parte de la base de datos que consta de sus propios datos de usuario, índices, archivos de configuración y archivos de anotaciones de transacción. A veces se denomina nodo o nodo de base de datos.

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.

procesador de línea de mandatos. Programa llamado DB2TX que:

Permite entrar mandatos de Net Search Extender

Procesa los mandatos

Visualiza el resultado.

rastreo. Acción de guardar información en un archivo que posteriormente se puede utilizar para encontrar la causa de un error.

RDB_node. Relational database node (nodo de base de datos relacional).

SBCS. Single-byte character support (soporte de caracteres de un solo byte).

segmento de vídeo. Sección de material filmado o registrado en cinta de vídeo.

servidor de partición de base de datos. Gestiona una *partición de base de datos*. Un servidor de partición de base de datos consta de un gestor de bases de datos y la colección de datos y recursos del sistema que gestiona. Generalmente cada máquina tiene asignado un solo servidor de partición de base de datos.

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, una sentencia de SQL estático no cambia, aunque si pueden cambiar los valores de las variables de sistema 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 XML Extender crea para mejorar el rendimiento cuando se buscan elementos o atributos en una columna XML.

tabla de metadatos. Véase *tabla de soporte administrativo*.

Tabla de referencia de DTD (tabla DTD_REF). Tabla que contiene definiciones DTD, que 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 soporte administrativo. Una de las tablas utilizadas por el extensor de DB2 UDB para procesar las peticiones del usuario referentes a objetos de imagen, audio y vídeo. Algunas tablas de soporte administrativo definen tablas de usuario y columnas que están habilitadas para un Extender. Otras tablas de soporte administrativo contienen información sobre atributos referente a objetos contenidos en columnas habilitadas. También se denomina *tabla de metadatos*.

tabla de texto. Una tabla de DB2 UDB que contiene *columnas de texto*.

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 administrativo. Una tabla utilizada por un extensor de DB2 UDB Extender Algunas tablas de soporte administrativo definen tablas de usuario y columnas que están habilitadas para un Extender. Otras tablas de soporte administrativo 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.

terabyte. Un billón (10^{12}) de bytes. Número de bytes equivalente a diez elevado a la potencia de doce. Cuando hace referencia a la capacidad de memoria, cantidad igual a 1.099.511.627.776 bytes.

text_node (nodo de texto). Representa el texto CDATA de un elemento.

tipo de datos. Atributo de columnas y literales.

tipo definido por el usuario (user-defined type, UDT). Un tipo de datos creado por un usuario de DB2, en oposición a un tipo de datos proporcionado por DB2 UDB como, por ejemplo, LONG VARCHAR.

tipo definido por el usuario (user-defined type, UDT). Tipo de datos que un usuario define para DB2. Los UDT se utilizan para diferenciar un LOB de otro. Por ejemplo, se puede crear un UDT para objetos de imagen y otro para objetos de audio. Aunque se almacenan en forma de BLOB, los objetos de imagen y de audio se consideran como tipos diferentes que los BLOB y como diferentes entre sí.

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. Una función de DB2 UDB proporcionada por XML Extender.

UDF (user-defined function). Véase *función definida por el usuario*.

UDT de XML. Un tipo definido por el usuario de DB2 UDB proporcionado por 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.

variable del lenguaje principal. Variable de un programa de aplicación a la que se puede hacer referencia en sentencias de SQL incorporado. Las variables de lenguaje principal son el mecanismo básico para transmitir datos entre las áreas de trabajo de una base de datos y de un programa de aplicación.

variable de referencia a archivos. Variable de programación que sirve para intercambiar un LOB con un archivo de una estación de trabajo cliente.

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 XML o atributo, y una columna de DB2 UDB al definir el esquema de indexado de la columna XML. Adicionalmente, Net Search Extender utiliza la vía de ubicación para búsquedas de texto estructural.

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 (/).

vídeo. Relativo a la porción de información registrada que se puede ver.

vista de catálogo. Una vista de la tabla del sistema creada con Net Search Extender con fines de administración. Una vista de catálogo contiene información acerca de las tablas y columnas habilitadas para su uso por Net Search Extender.

vista de unión. Una vista de DB2 UDB creada por la sentencia ″CREATE VIEW″ que une una o más tablas.

vista por omisión. 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).

Avisos

Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde 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, COMERCIALIZACIÓN 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 inexactitudes 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 cambios en los productos y programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos

sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADÁ

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 este documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

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 disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse 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 objetivos.

Este manual puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con 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 asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (*nombre de la empresa*) (*año*). Partes de este código proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *_entre el o los años_*. Reservados todos los derechos.

Marcas registradas

Los términos siguientes son marcas registradas de International Business Machines Corporation en los EE.UU. y/o en otros países y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeriesLAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Los términos siguientes son marcas registradas de otras empresas y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB:

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los EE.UU. y/o en otros países.

Intel y Pentium son marcas registradas de Intel Corporation en los EE.UU. y/o en otros países.

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los EE.UU. y/o en otros países.

UNIX es marca registrada de The Open Group en los EE.UU. y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Cómo ponerse en contacto con IBM

En los EE.UU., puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (426-4968) para marketing y ventas de DB2

En Canadá, puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-800-465-9600 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (1-800-426-4968) para marketing y ventas de DB2

Para localizar una oficina de IBM en su país o región, consulte IBM Directory of Worldwide Contacts en el sitio Web http://www.ibm.com/planetwide

Información sobre productos

La información relacionada con productos DB2 Universal Database se encuentra disponible por teléfono o a través de la World Wide Web en el sitio http://www.ibm.com/software/data/db2/udb

Este sitio contiene la información más reciente sobre la biblioteca técnica, pedidos de manuales, descargas de productos, grupos de noticias, FixPaks, novedades y enlaces con recursos de la Web.

Si vive en los EE.UU., puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

Para obtener información sobre cómo ponerse en contacto con IBM desde fuera de los EE.UU., vaya a la página IBM Worldwide en el sitio www.ibm.com/planetwide

SC10-3750-01

Spine information:

IBM® DB2® Universal Database™ XML Extender Administración y programación Versión 8.2